Data Visualization I

Dav King

1-13-2021

Main Ideas

- Data visualization is an **extremely** effective way to express information and extract meaning from data.
- We can build up an effective visualization systematically layer by layer using a grammar of graphics (`ggplot2`).

Coming Up

- Lab 2 due Wednesday the 19th.
- No Lab on Monday the 17th.

"The simple graph has brought more information to the data analyst's mind than any other device" - John Tukey

Lecture Notes and Exercises

Reminder

Before we start the exercise, we need to configure git so that RStudio can communicate with GitHub. This requires two pieces of information: your email address and your GitHub username.

Configure git by running the following code in the **terminal**. Fill in your GitHub username and the email address associated with your GitHub account.

```
git config --global user.name 'username'
git config --global user.email 'useremail'
```

Next load the `tidyverse` package. Recall, a package is just a bundle of shareable code.

```
library(tidyverse)
```

Exploratory data analysis (EDA) is an approach to analyzing datasets in order to summarize the main characteristics, often with visual representations of the data (today). We can also calculate summary statistics and perform data wrangling, manipulation, and transformation (next week).

We will use `ggplot2` to construct visualizations. The gg in `ggplot2` stands for "grammar of graphics", a system or framework that allows us to describe the components of a graphic, building up an effective visualization layer by later.

Minneapolis Housing Data

We will introduce visualization using data on single-family homes sold in Minneapolis, Minnesota between 2005 and 2015.

Question: What happens when you click the green arrow in the code chunk below? What changes in the "Environment" pane?

The dataset "mn_homes" appears in the Environment, alongside a brief description of how many observations and variables the dataset has. The function itself reads the dataset from a csv file into R.

```
mn_homes <- read_csv("mn_homes.csv")
```

```
glimpse(mn_homes)
```

```
## Rows: 495
## Columns: 13
## $ saleyear      <dbl> 2012, 2014, 2005, 2010, 2010, 2013, 2011, 2007, 2013, 20~
## $ salemonth     <dbl> 6, 7, 7, 6, 2, 9, 1, 9, 10, 6, 7, 8, 5, 2, 7, 6, 10, 6, ~
## $ salesprice    <dbl> 690467.0, 235571.7, 272507.7, 277767.5, 148324.1, 242871~
## $ area          <dbl> 3937, 1440, 1835, 2016, 2004, 2822, 2882, 1979, 3140, 35~
## $ beds          <dbl> 5, 2, 2, 3, 3, 3, 4, 3, 4, 3, 3, 3, 2, 3, 3, 6, 2, 3, 2,~
## $ baths         <dbl> 4, 1, 1, 2, 1, 3, 3, 2, 2, 2, 2, 2, 2, 1, 2, 2, 2, 2, 1,~
## $ stories       <dbl> 2.5, 1.7, 1.7, 2.5, 1.0, 2.0, 1.7, 1.5, 1.5, 2.5, 1.0, 2~
## $ yearbuilt     <dbl> 1907, 1919, 1913, 1910, 1956, 1934, 1951, 1929, 1940, 19~
## $ neighborhood  <chr> "Lowry Hill", "Cooper", "Hiawatha", "King Field", "Shing~
## $ community     <chr> "Calhoun-Isles", "Longfellow", "Longfellow", "Southwest"~
## $ lotsize       <dbl> 6192, 5160, 5040, 4875, 5060, 6307, 6500, 5600, 6350, 75~
## $ numfireplaces <dbl> 0, 0, 0, 0, 0, 2, 2, 0, 1, 0, 0, 0, 0, 1, 1, 1, 0, 1, 0,~
## $ fireplace     <lgl> FALSE, FALSE, FALSE, FALSE, FALSE, TRUE, TRUE, FALSE, TR~
```

Question: What does each row represent? Each column?

Each row represents individual houses, and their statistics across a number of different variables. Each column represents an individual variable on which different houses can be compared, such as sales price, number of bedrooms, or whether they have a fireplace.

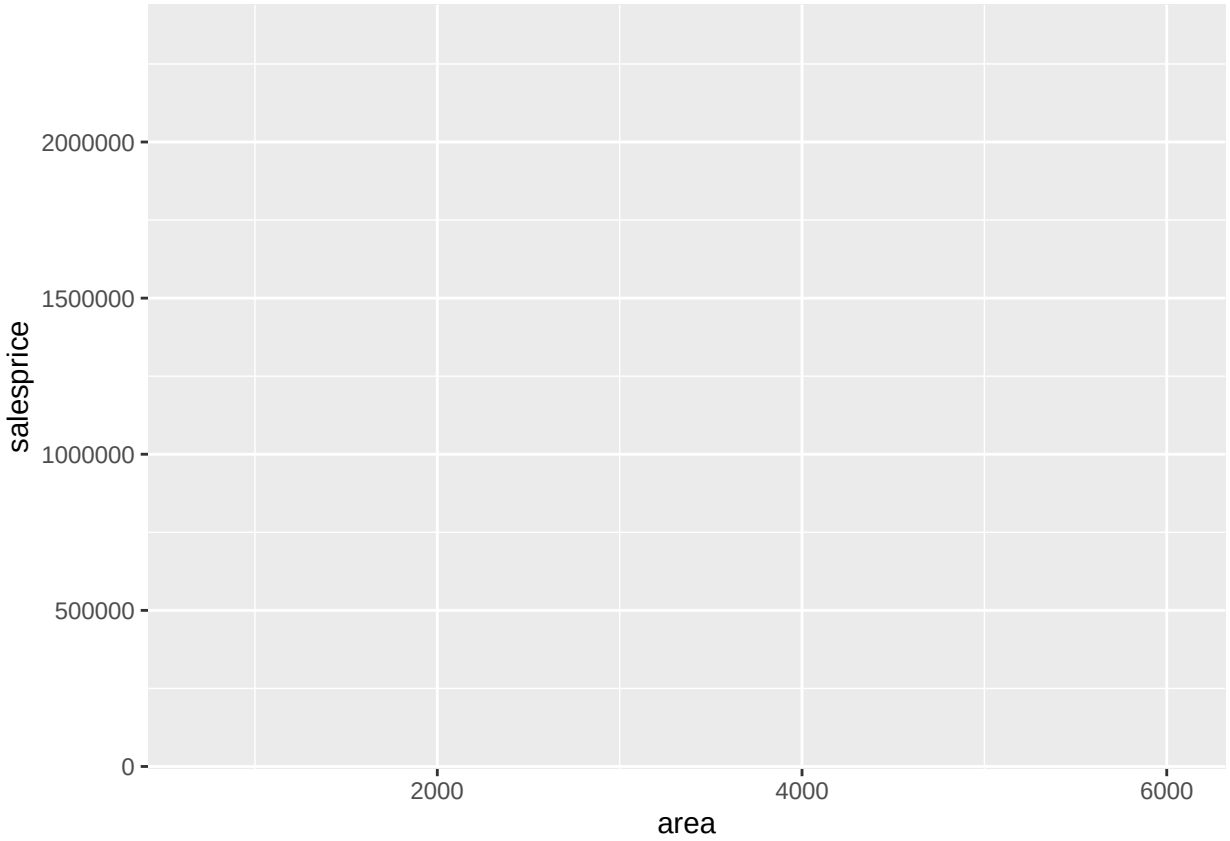
First Visualization

`ggplot` creates the initial base coordinate system that we will add layers to. We first specify the dataset we will use with `data = mn_homes`. The `mapping` argument is paired with an aesthetic (`aes`), which tells us how the variables in our dataset should be mapped to the visual properties of the graph.

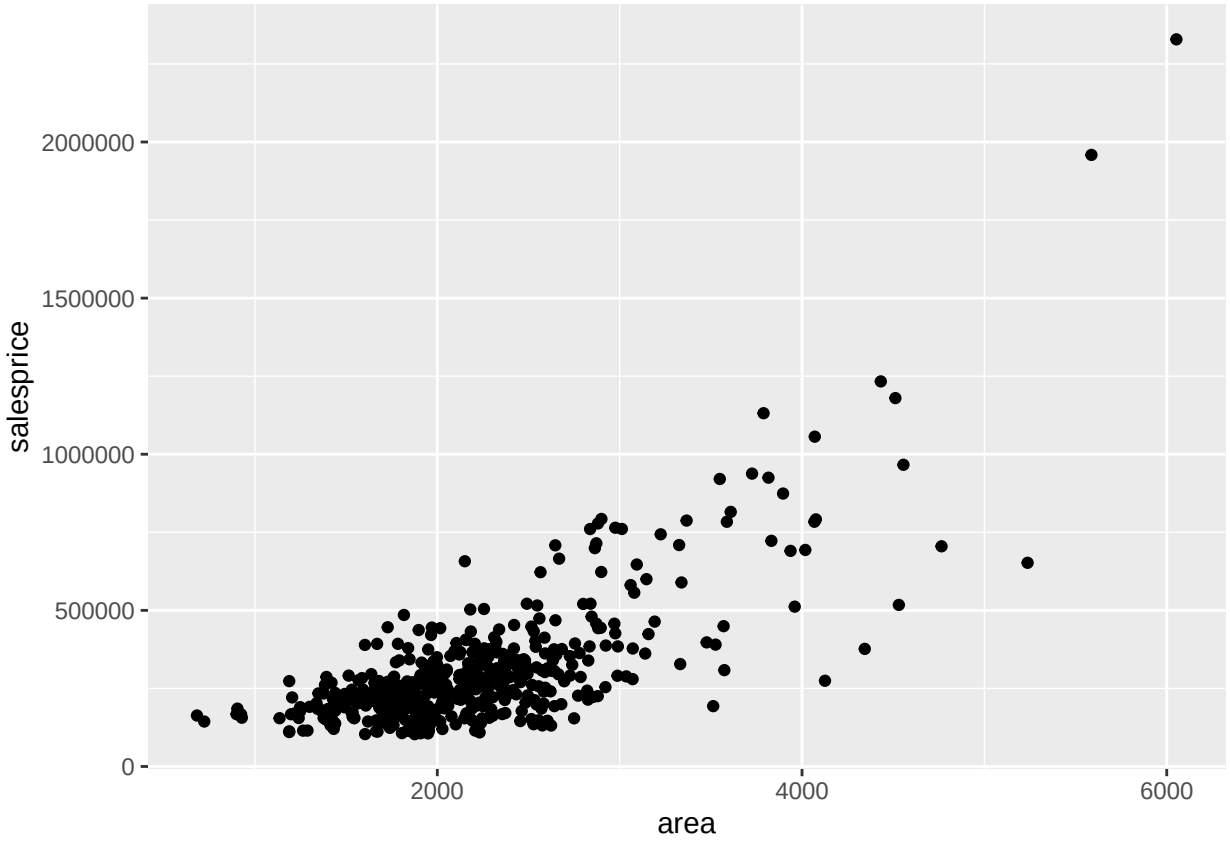
Question: What does the code chunk below do?

Iteratively, these three code chunks slowly build our graph into its relatively final state. The first chunk just greats the ggplot grid, telling it which dataset to read from and which variables to map onto the x and y axes. The second code chunk takes this, and adds the actual x-y scatterplot of the dataset. The third code chunk takes all of this and fits a loess regression line to the data.

```
ggplot(data = mn_homes,
       mapping = aes(x = area, y = salesprice))
```

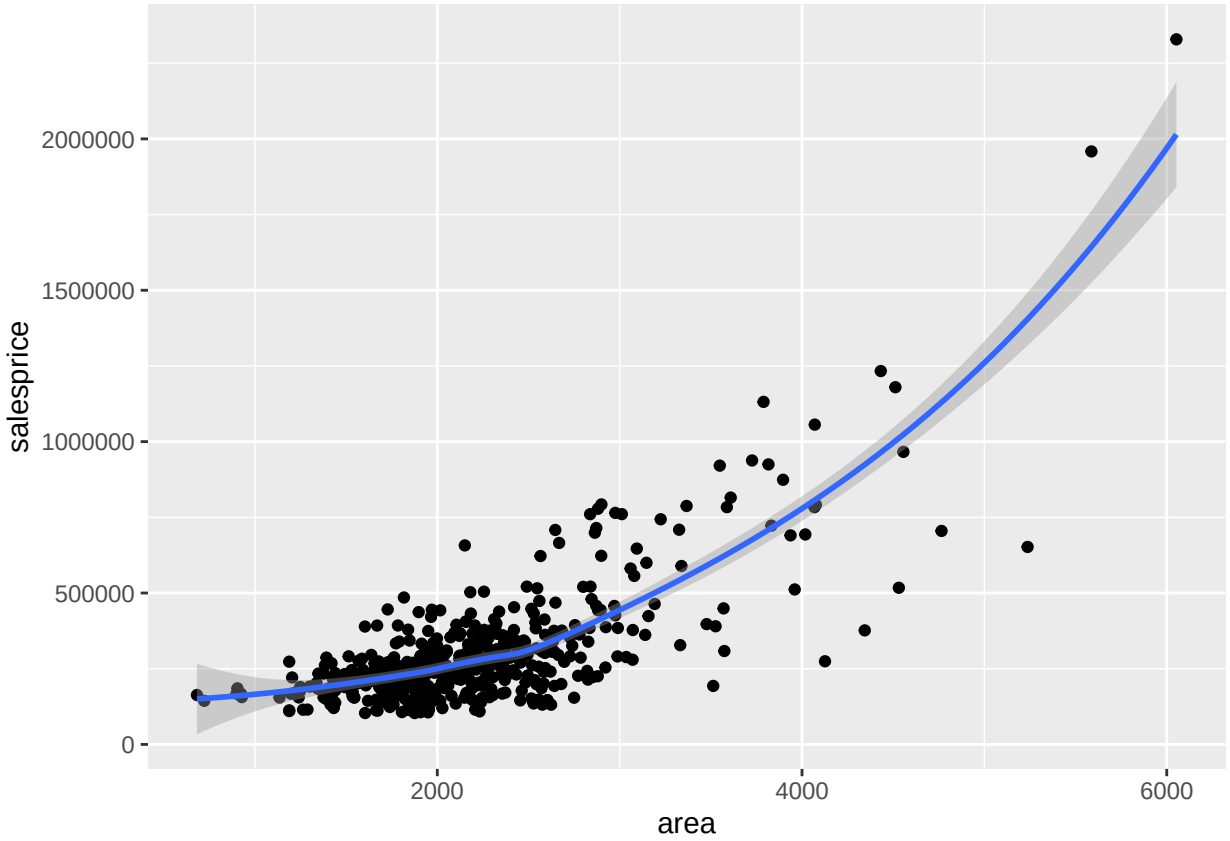



```
ggplot(data = mn_homes,
       mapping = aes(x = area, y = salesprice)) +
  geom_point()
```




```
ggplot(data = mn_homes,
       mapping = aes(x = area, y = salesprice)) +
   geom_point() +
   geom_smooth()
```

```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```

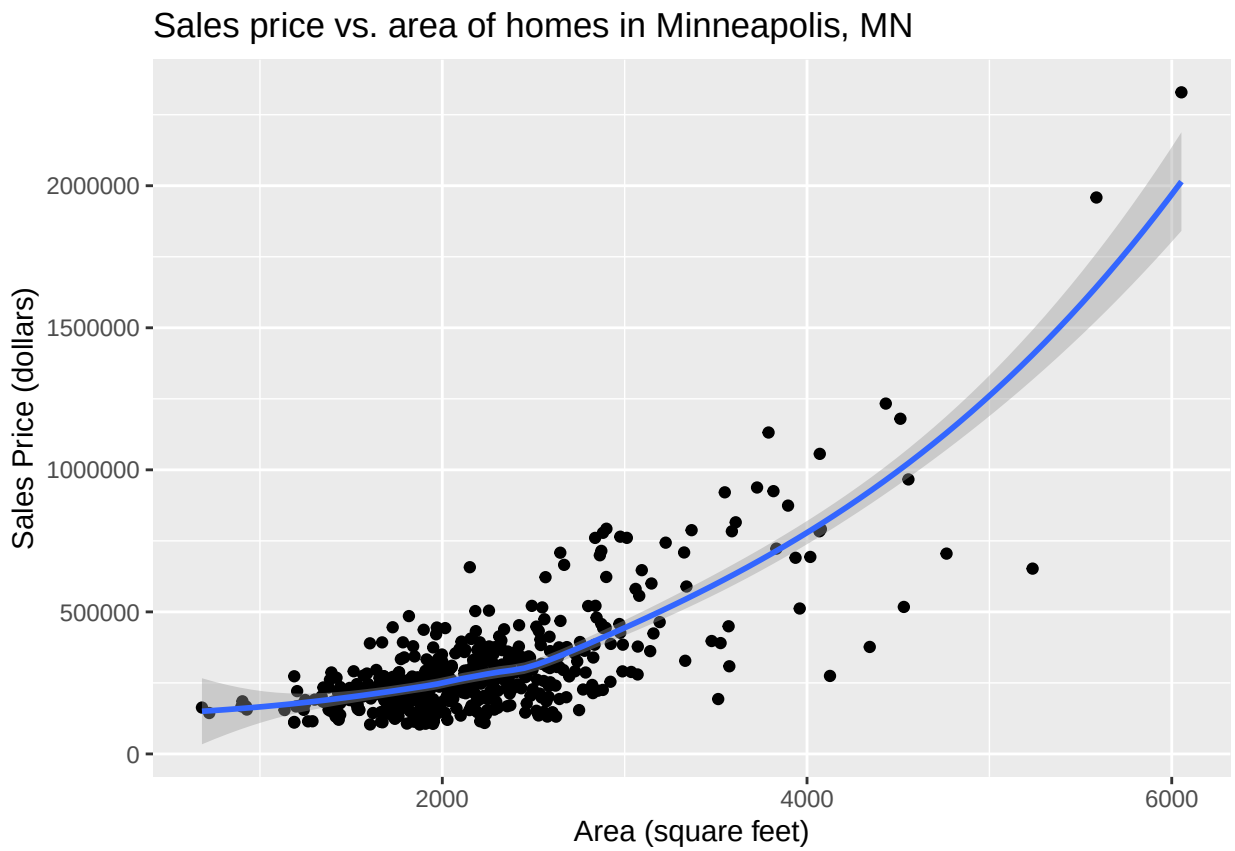


Run `?geom_smooth` in the console. What does this function do?

This fits a loess regression line (moving regression) to the data.

```
ggplot(data = mn_homes,
       mapping = aes(x = area, y = salesprice)) +
   geom_point() +
   geom_smooth() +
   labs(title = "Sales price vs. area of homes in Minneapolis, MN",
        x = "Area (square feet)", y = "Sales Price (dollars)")
```

```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```



The procedure used to construct plots can be summarized using the code below.

```
ggplot(data = [dataset],
       mapping = aes(x = [x-variable], y = [y-variable])) +
   geom_xxx() +
   geom_xxx() +
  other options
```

Question: What do you think `eval = FALSE` is doing in the code chunk above?

`eval = FALSE` is telling that code chunk not to run, as it is filled with placeholder variables that will not actually do anything and thus evaluating it will generate nothing but error messages.

Aesthetics

An aesthetic is a visual property of one of the objects in your plot.

- shape
- color
- size
- alpha (transparency)

We can map a variable in our dataset to a color, a size, a transparency, and so on. The aesthetics that can be used with each `geom_` can be found in the documentation.

Question: What will the visualization look like below? Write your answer down before running the code.

This is going to create a similar data plot to what we had before. It will build a scatterplot with area on the x axis and sales price on the y axis. It will add a title and labels for the axes, but it will nto create the loess regression line for the data. However, unlike before, this plot will have color. The data points will be colored based on whether the home had a fireplace or not, and it will use the viridis color scheme that is friendly to color-blind viewers.

Here we are going to use the viridis package, which has more color-blind accessible colors. `scale_color_viridis` specifies which colors you want to use. You can learn more about the options here.

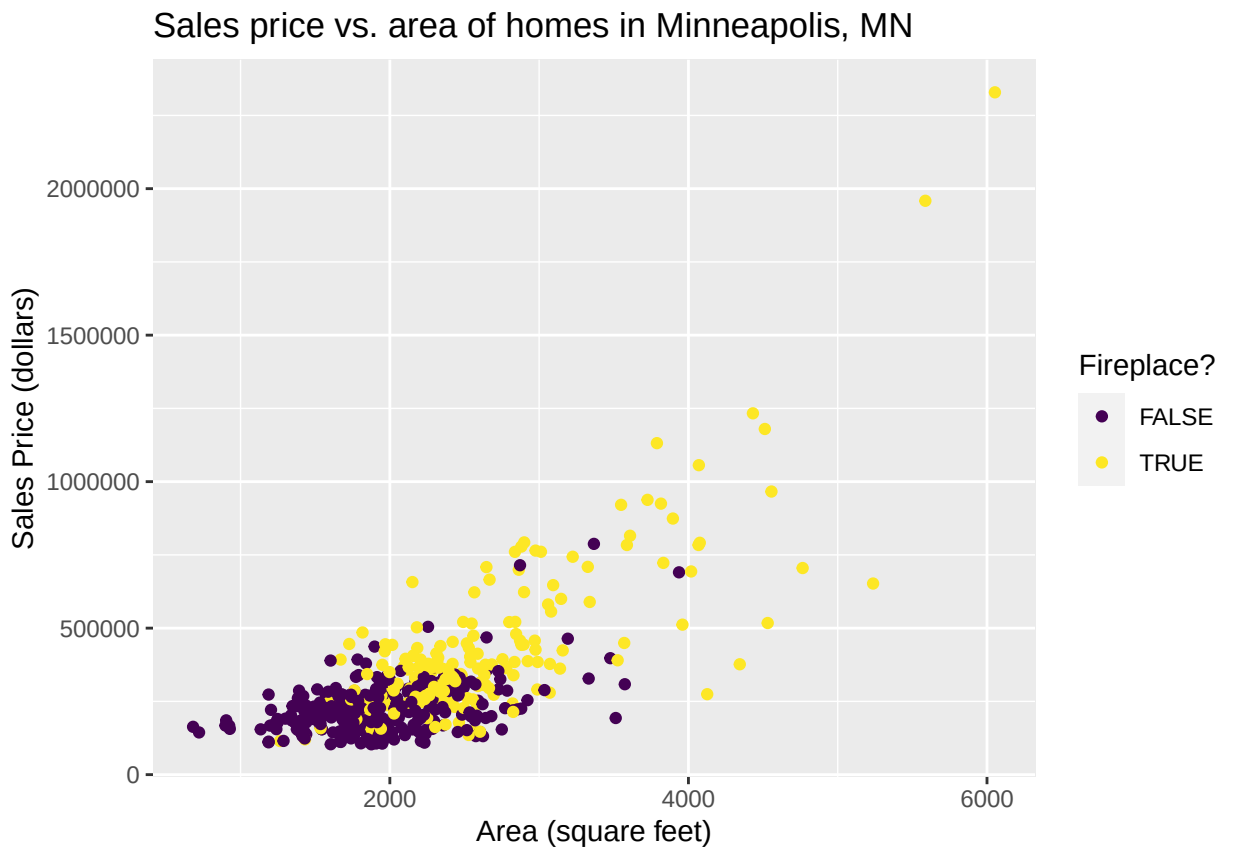
Other sources that can be helpful in devising accessible color schemes include Color Brewer, the Wes Anderson package, and the cividis package.

This visualization shows a scatterplot of area (x variable) and sales price (y variable). Using the viridis function, we make points for houses with a fireplace yellow and those without purple. We also add axis and an overall label.

```
library(viridis)
```

```
## Loading required package: viridisLite
```

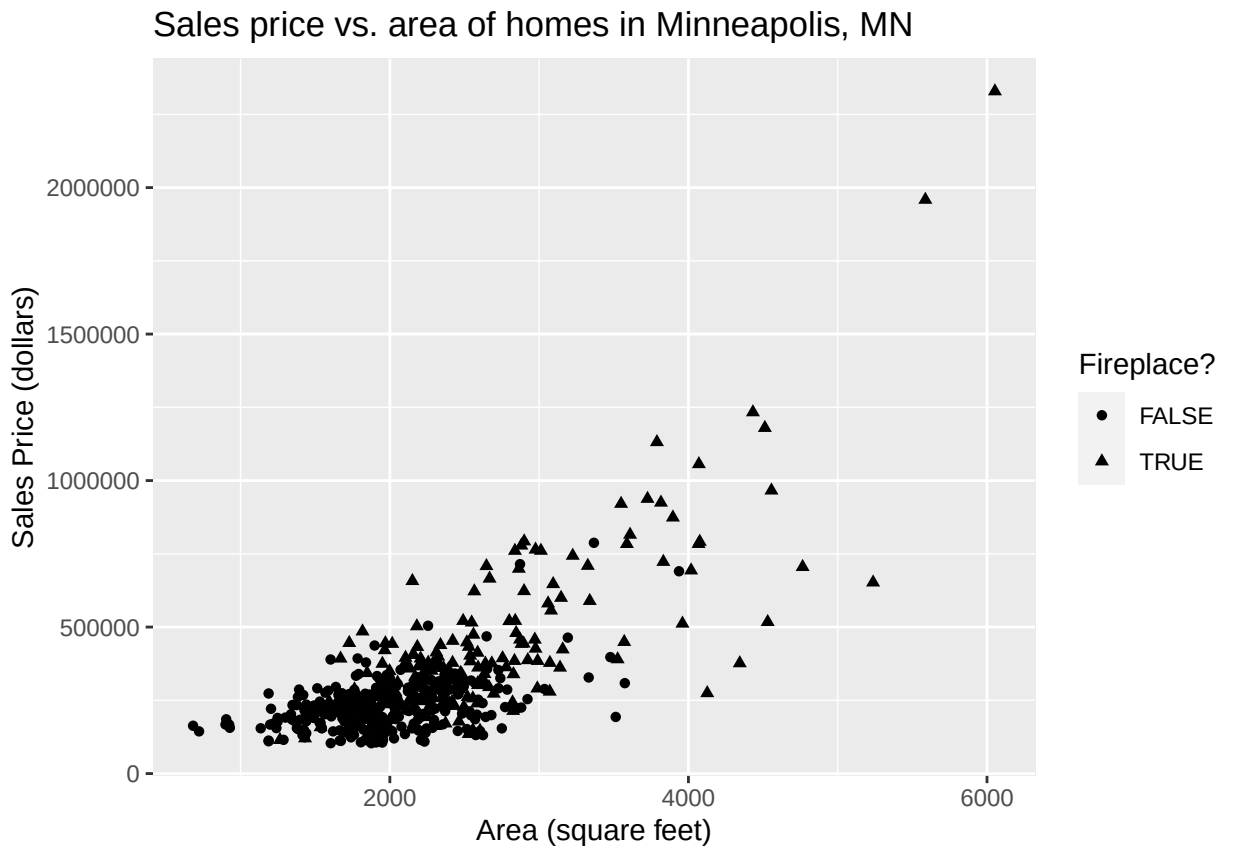
```
ggplot(data = mn_homes,
       mapping = aes(x = area, y = salesprice,
                     color = fireplace)) +
   geom_point() +
   labs(title = "Sales price vs. area of homes in Minneapolis, MN",
        x = "Area (square feet)", y = "Sales Price (dollars)") +
        scale_color_viridis(discrete=TRUE, option = "D", name="Fireplace?")
```



Question: What about this one?

This will do almost the exact same thing as the graph before it did. However, instead of changing the color of the points based on whether the homes had a fireplace or not, it will leave all points black and instead change the *shape* of the points on that basis instead.

```
ggplot(data = mn_homes,
       mapping = aes(x = area, y = salesprice,
                     shape = fireplace)) +
   geom_point() +
   labs(title = "Sales price vs. area of homes in Minneapolis, MN",
        x = "Area (square feet)", y = "Sales Price (dollars)",
        shape="Fireplace?")
```

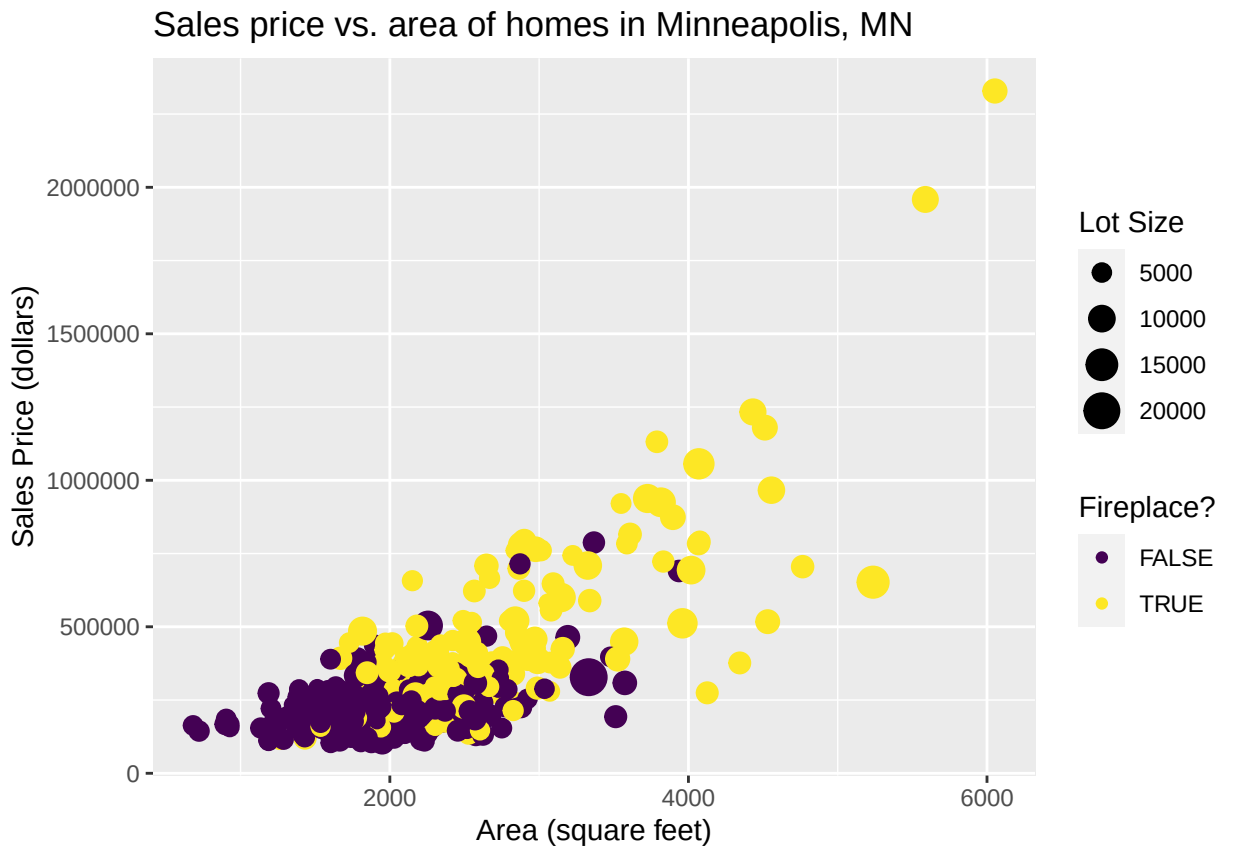


Question: This one?

Creating the same plot as the previous two examples, this plot will then color (using the viridis palate) by whether the homes had a fireplace or not, and scale the points by size according to how big the lot of the home was.

```
ggplot(data = mn_homes,
       mapping = aes(x = area, y = salesprice,
                     color = fireplace,
                     size = lotsize)) +
   geom_point() +
   labs(title = "Sales price vs. area of homes in Minneapolis, MN",
        x = "Area (square feet)", y = "Sales Price (dollars)",
```

```
       size = "Lot Size") +
  scale_color_viridis(discrete=TRUE, option = "D",name="Fireplace?")
```



Question: Are the above visualizations effective? Why or why not? How might you improve them?

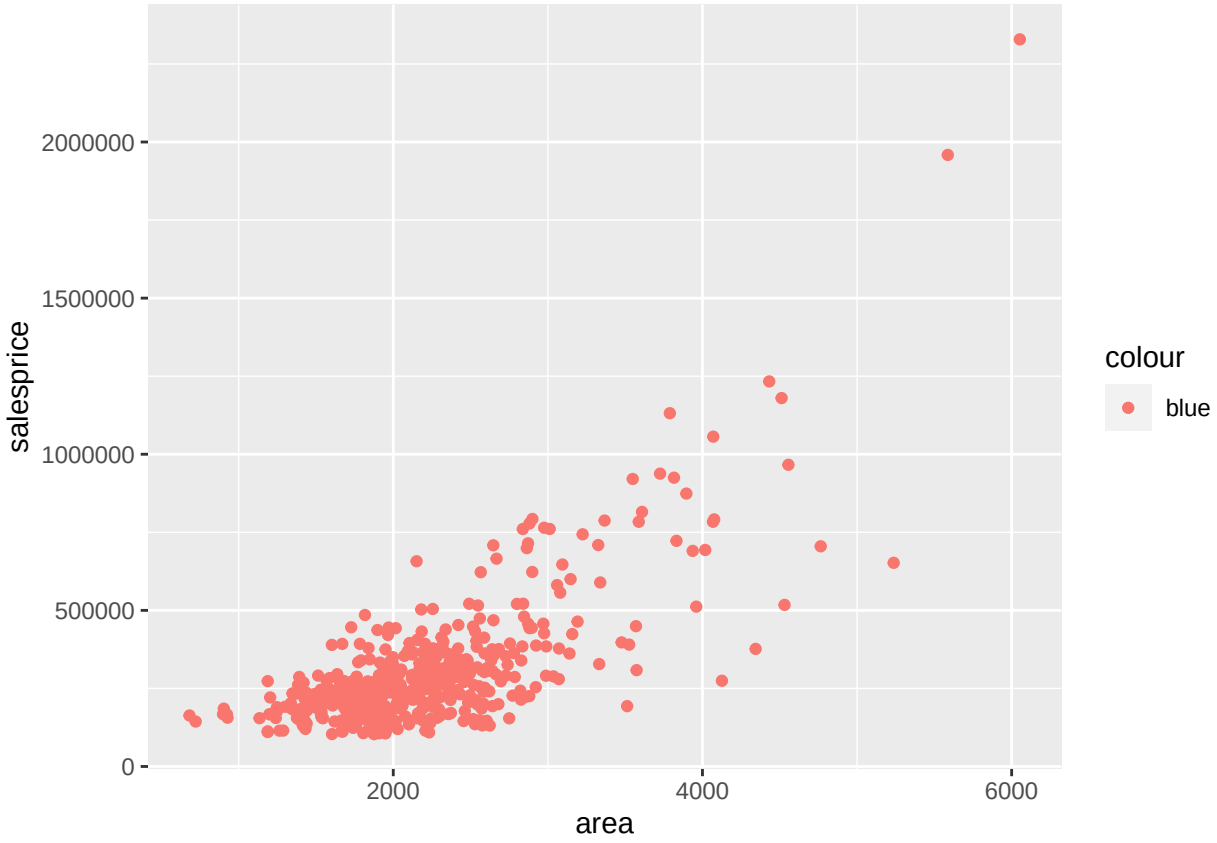
While all are effective in some manner, some are better than others. The most effective visualization is the first one, which colors the points - it clearly shows which ones fall into the category of which variable. The second graph does not match this, as it's hard to see visually the difference between shapes. Similarly, the third one is difficult to read because adding the scale by lot size does not leave points that are particularly discernible from one another, adding clutter that does not add additional information to the graph.

However, the first graph still has its limitations - even if viridis is built for colorblind readers, it still isn't necessarily readable to all of them. Additionally, if graphs were printed in black and white, the color would not help at all and would leave an uninterpretable graph. In order to make the most legible graph with those variables, instead of coloring according to fireplace, I would facet the grid into two columns along that variable instead - leaving two entirely separate, side-by-side plots of the data.

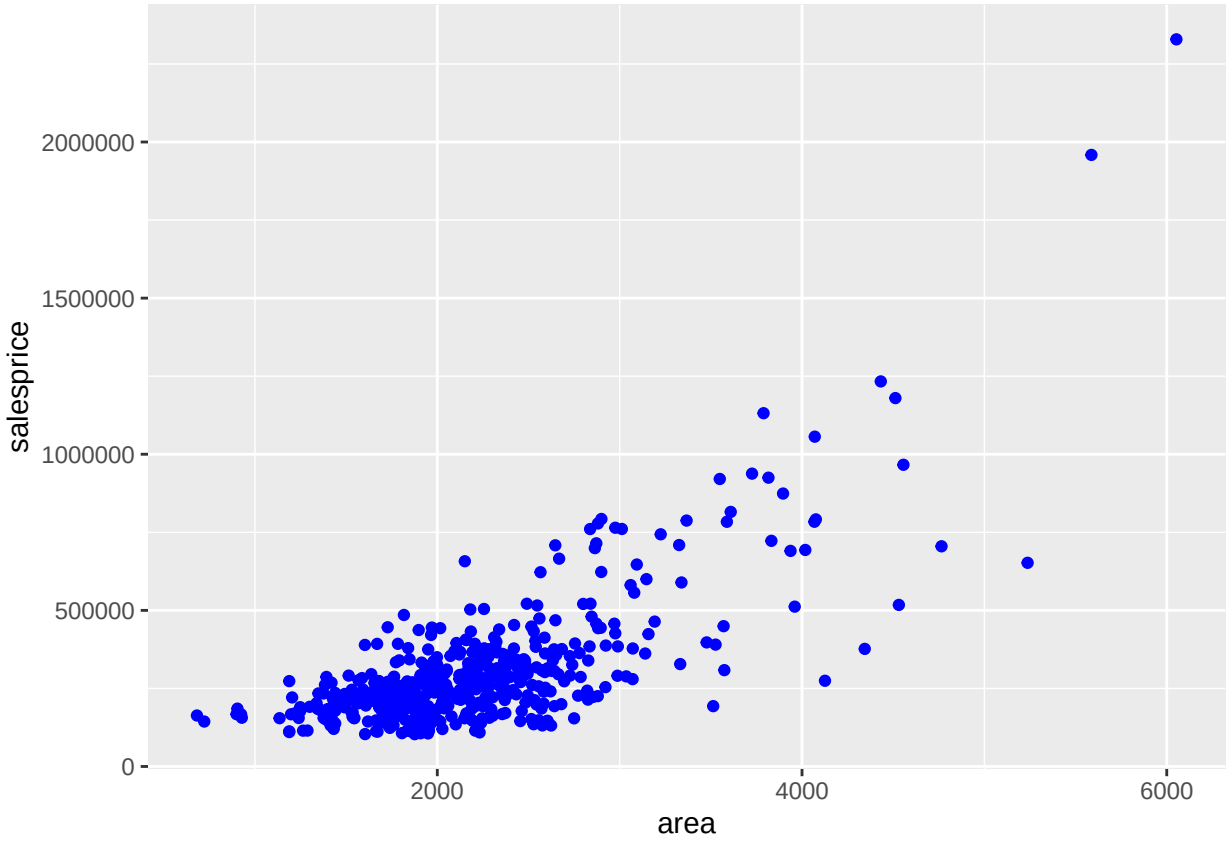
Question: What is the difference between the two plots below?

In the first plot, `color = "blue"` is included in the `aes` function, which means that the data points get colored in whatever the default color of the ggplot is, and that a legend is created which is titled "blue". In the second, `color = "blue"` is instead included in the `geom_point` function, which actually colors the points themselves instead of simply labeling a legend.

```
ggplot(data = mn_homes) +
  geom_point(mapping = aes(x = area, y = salesprice, color = "blue"))
```




```
ggplot(data = mn_homes) +
  geom_point(mapping = aes(x = area, y = salesprice), color = "blue")
```

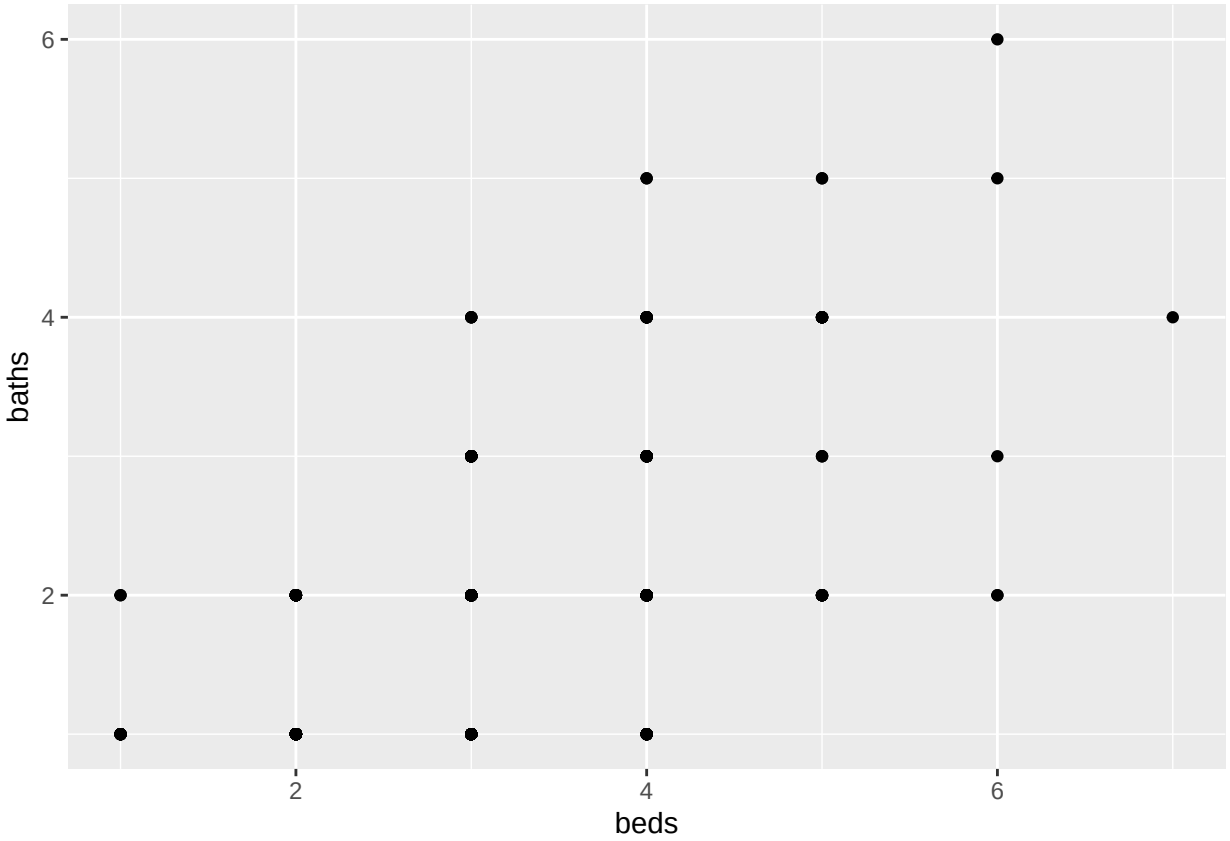


Use `aes` to map variables to plot features, use arguments in `geom_xxx` for customization not mapped to a variable.

Mapping in the `ggplot` function is global, meaning they apply to every layer we add. Mapping in a particular `geom_xxx` function treats the mappings as local.

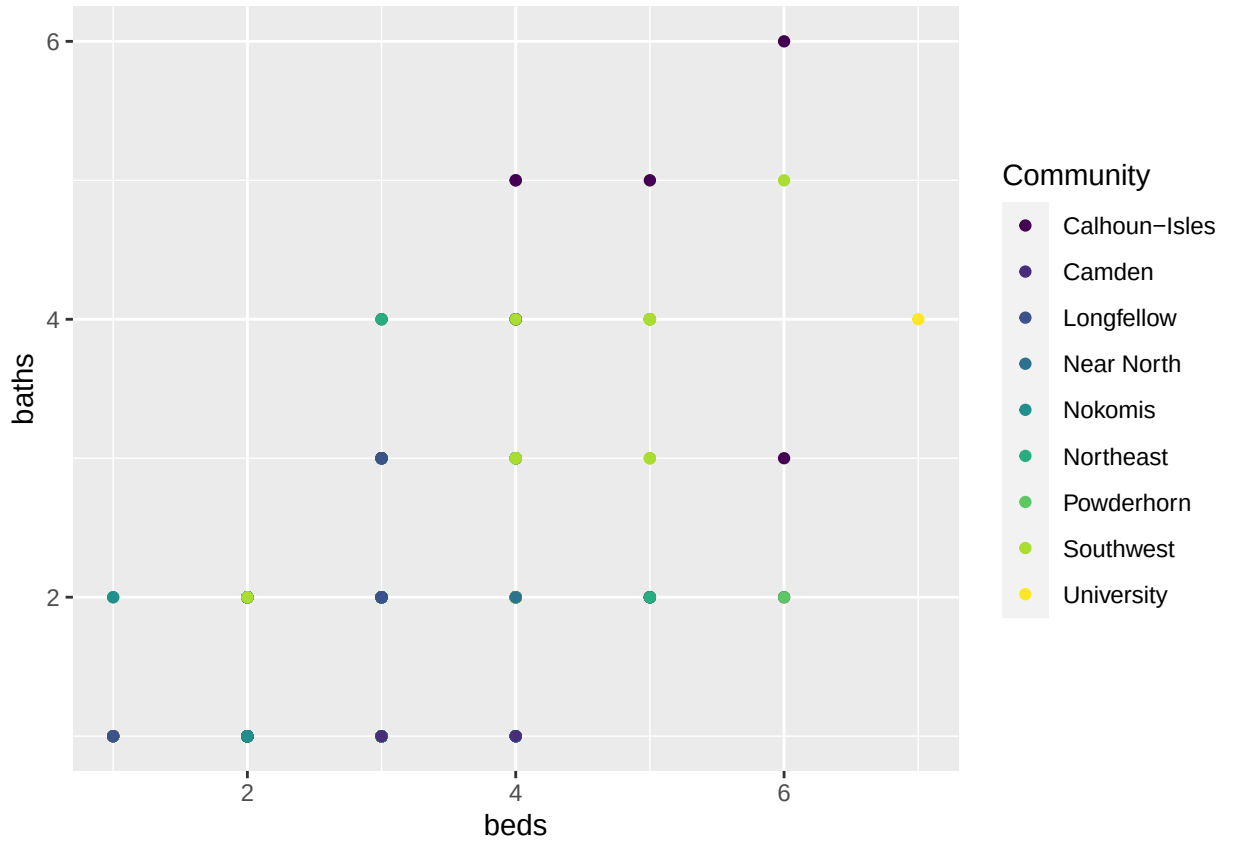
Question: Create a scatterplot using variables of your choosing using the `mn_homes` data.

```
ggplot(data = mn_homes,
       mapping = aes(x = beds, y = baths)) +
  geom_point()
```



Question: Modify your scatterplot above by coloring the points for each community.

```
ggplot(data = mn_homes,
       mapping = aes(x = beds, y = baths,
       color = community)) +
   geom_point() +
  scale_color_viridis(discrete = TRUE, option = "D", name = "Community")
```

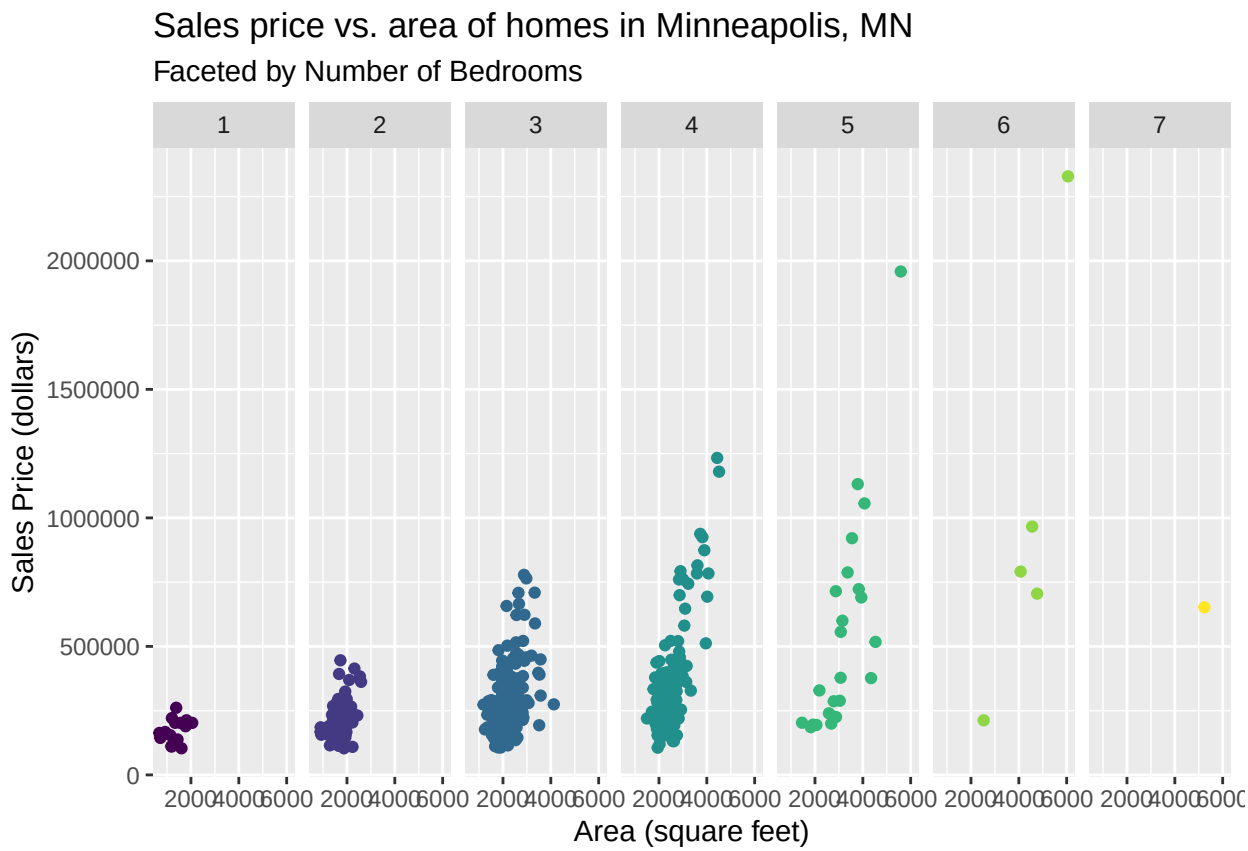


Faceting

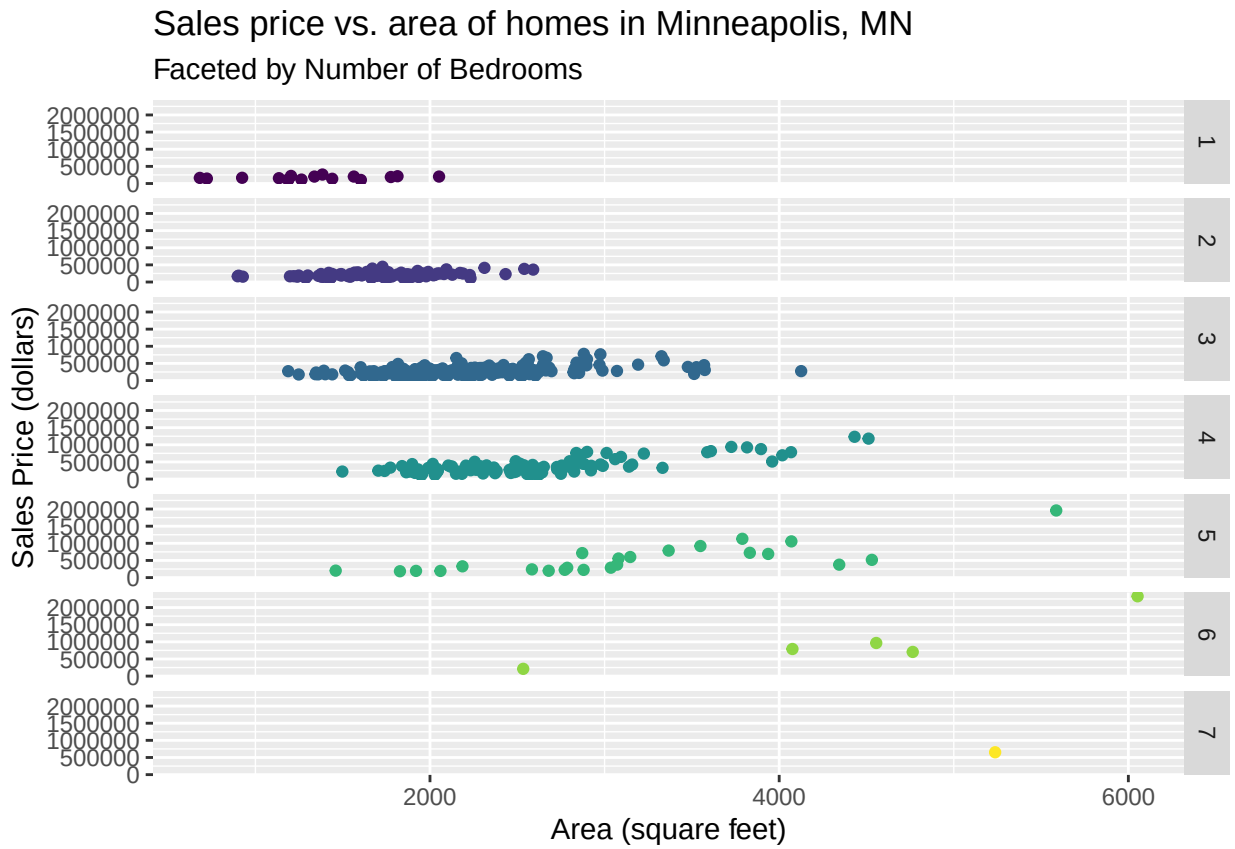
We can use smaller plots to display different subsets of the data using faceting. This is helpful to examine conditional relationships.

Let's try a few simple examples of faceting. Note that these plots should be improved by careful consideration of labels, aesthetics, etc.

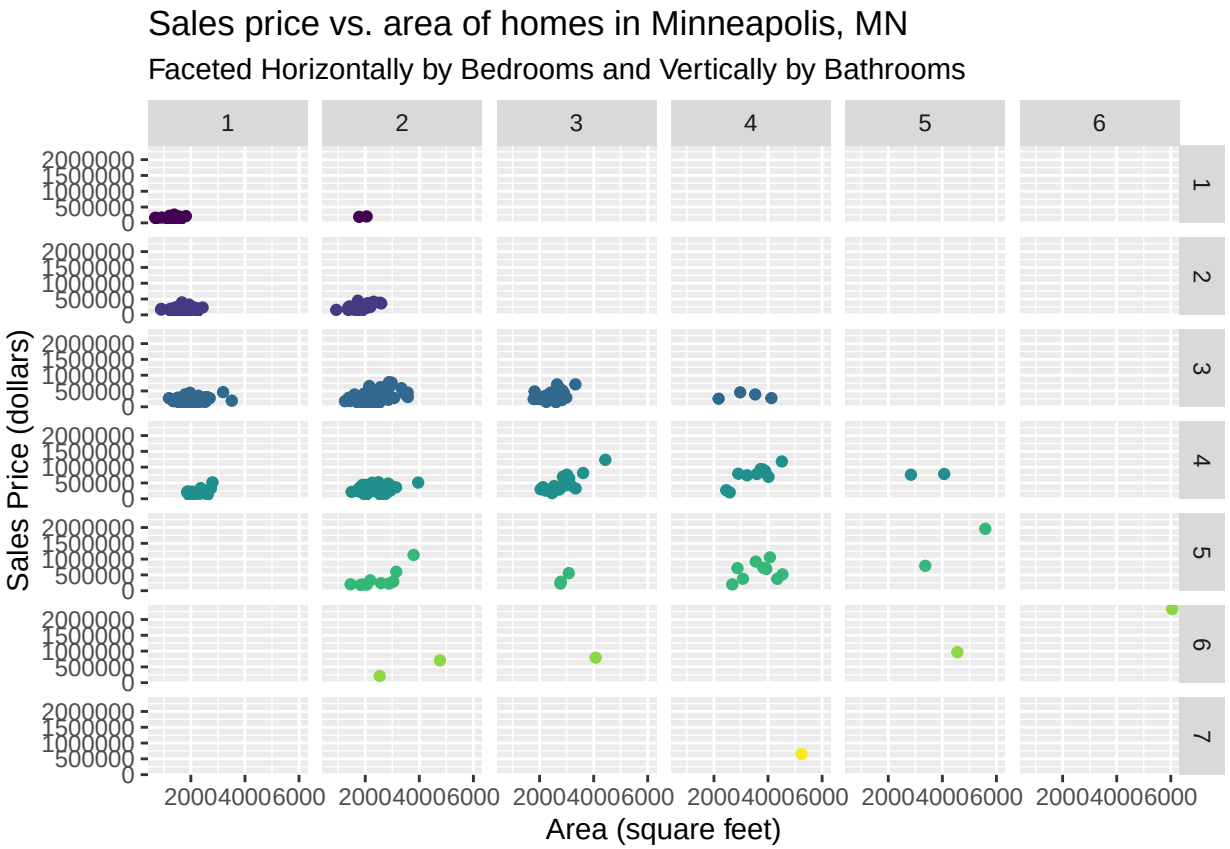
```
ggplot(data = mn_homes,
       mapping = aes(x = area, y = salesprice, color = beds)) +
   geom_point() +
   labs(title = "Sales price vs. area of homes in Minneapolis, MN",
        x = "Area (square feet)", y = "Sales Price (dollars)",
        subtitle = "Faceted by Number of Bedrooms") +
   facet_grid(. ~ beds) +
  scale_color_viridis() +
  guides(color = "none")
```



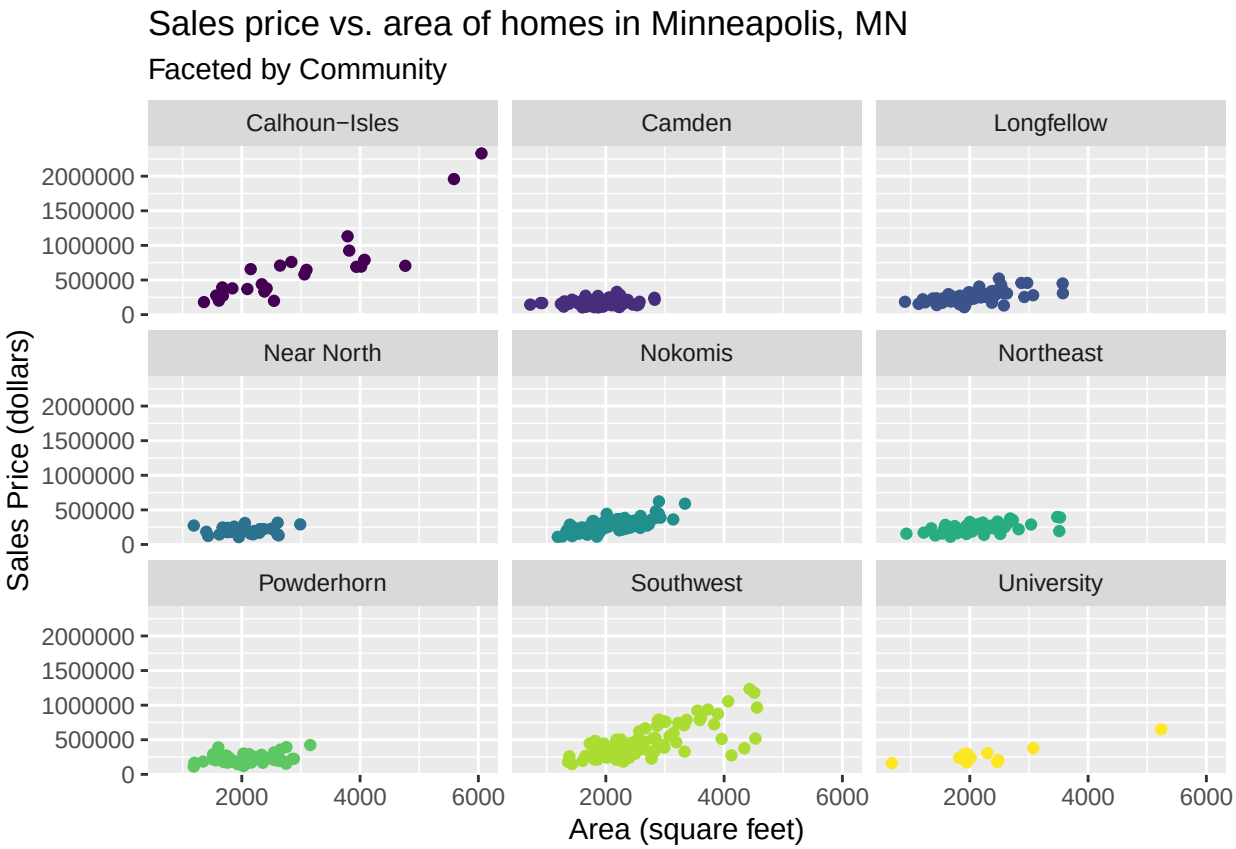

```
ggplot(data = mn_homes,
       mapping = aes(x = area, y = salesprice, color = beds)) +
  geom_point() +
  labs(title = "Sales price vs. area of homes in Minneapolis, MN",
       x = "Area (square feet)", y = "Sales Price (dollars)",
       subtitle = "Faceted by Number of Bedrooms") +
  facet_grid(beds ~ .) +
 scale_color_viridis() +
 guides(color = "none")
```




```
ggplot(data = mn_homes,
       mapping = aes(x = area, y = salesprice, color = beds)) +
  geom_point() +
  labs(title = "Sales price vs. area of homes in Minneapolis, MN",
       x = "Area (square feet)", y = "Sales Price (dollars)",
       subtitle = "Faceted Horizontally by Bedrooms and Vertically by Bathrooms") +
  facet_grid(beds ~ baths) +
 scale_color_viridis() +
 guides(color = "none")
```




```
ggplot(data = mn_homes,
       mapping = aes(x = area, y = salesprice, color = community)) +
  geom_point() +
  labs(title = "Sales price vs. area of homes in Minneapolis, MN",
       x = "Area (square feet)", y = "Sales Price (dollars)",
       subtitle = "Faceted by Community") +
  facet_wrap(~ community) +
 scale_color_viridis(discrete = TRUE) +
 guides(color = "none")
```



`facet_grid()`

- 2d grid
- rows ~ cols
- use . for no plot

`facet_wrap()`

- 1d ribbon wrapped into 2d

Practice

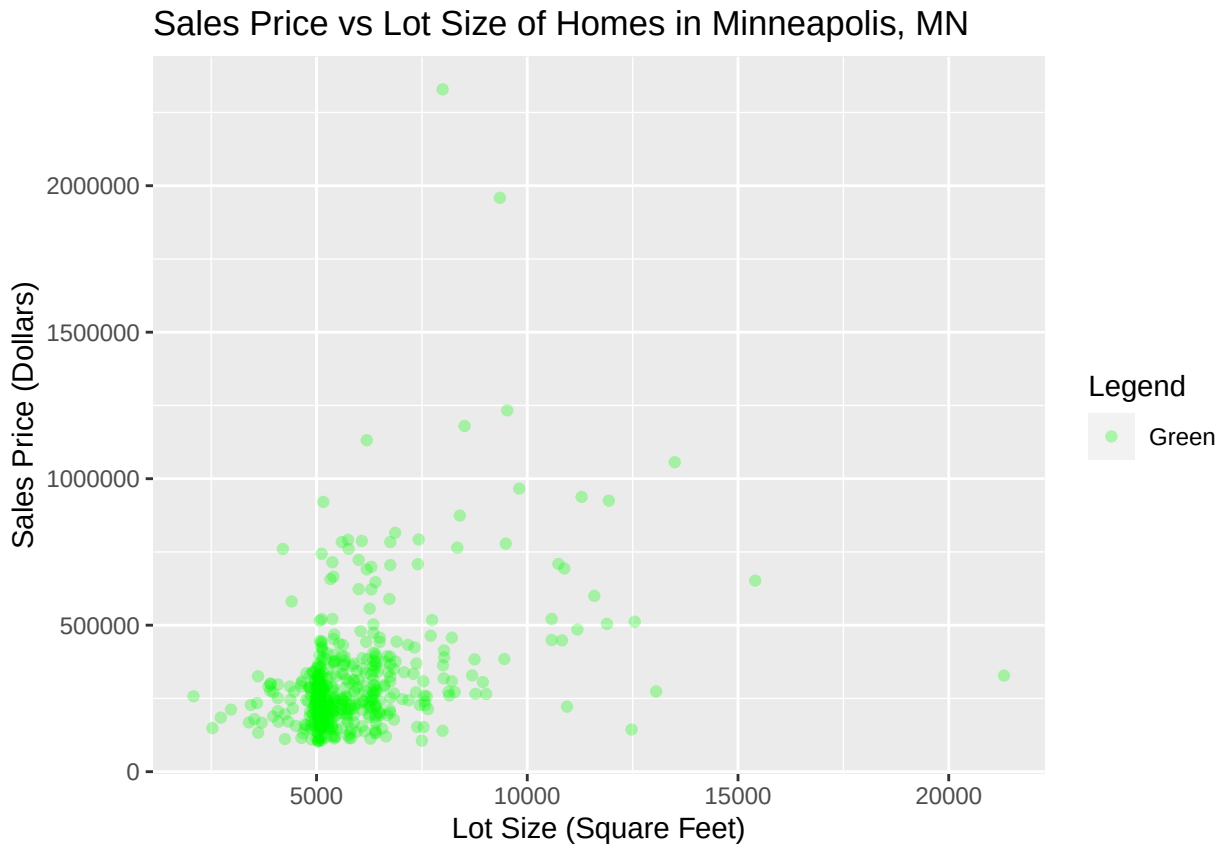
(1) Modify the code outline to make the changes described below.

- Change the color of the points to green.
- Add `alpha` to make the points more transparent.
- Add labels for the x axis, y axis, and the color of the points.
- Add an informative title.
- Consider using the `viridis` palette. (Note, you can't do all of these things at once in terms of color, these are just suggestions.)

When you are finished, remove `eval = FALSE` and knit the file to see the changes.

Here is some starter code:

```
ggplot(data = mn_homes,
       mapping = aes(x = lotsize, y = salesprice)) +
   geom_point(aes(color = "Green"), alpha = 0.3) +
   labs(title = "Sales Price vs Lot Size of Homes in Minneapolis, MN",
        x = "Lot Size (Square Feet)", y = "Sales Price (Dollars)",
        color = "Legend") +
  scale_color_identity(guide = "legend")
```



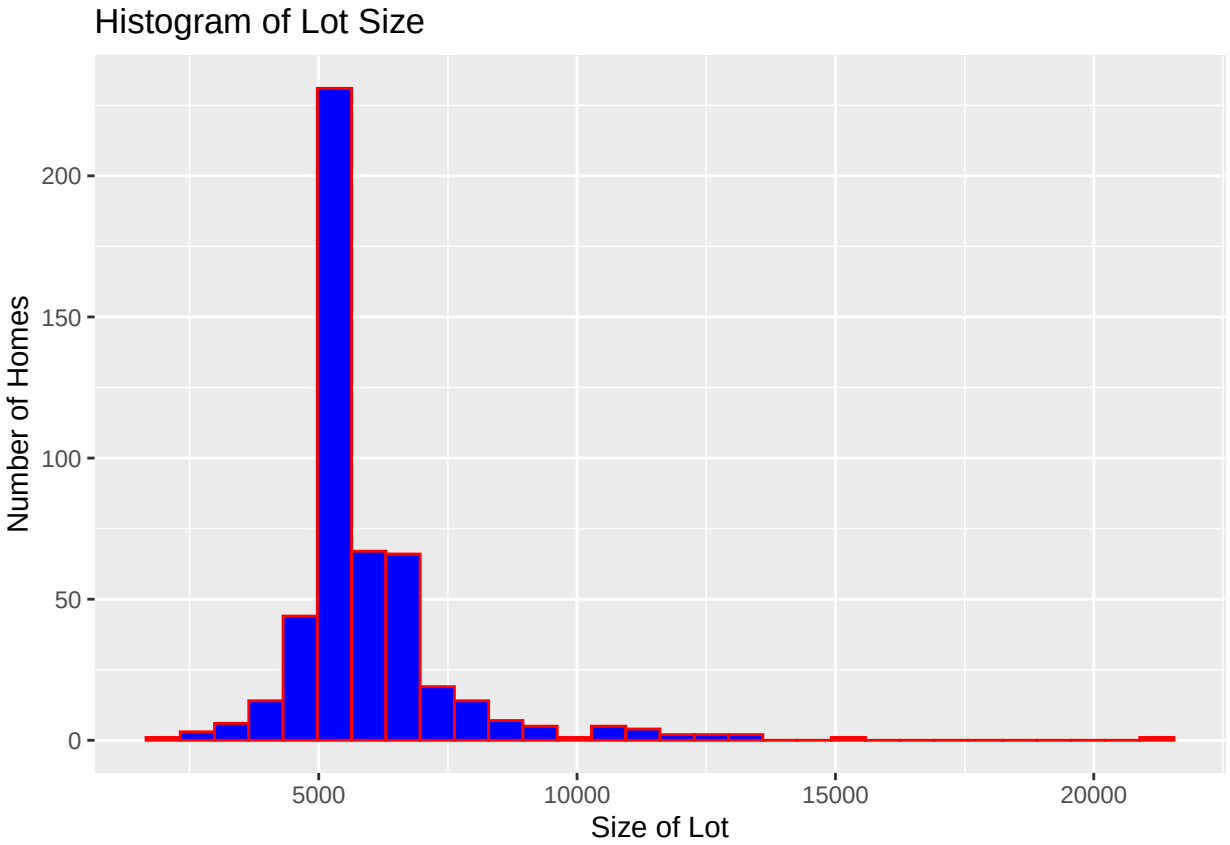
(2) Modify the code outline to make the changes described below.

- Create a histogram of `lotsize`.
- Modify the histogram by adding `fill = "blue"` inside the `geom_histogram()` function.
- Modify the histogram by adding `color = "red"` inside the `geom_histogram()` function.

When you are finished, remove `eval = FALSE` and knit the file to see the changes.

```
ggplot(data = mn_homes,
       mapping = aes(x = lotsize)) +
  geom_histogram(fill = "blue", color = "red") +
  labs(title = "Histogram of Lot Size" , x = "Size of Lot", y = "Number of Homes")
```

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



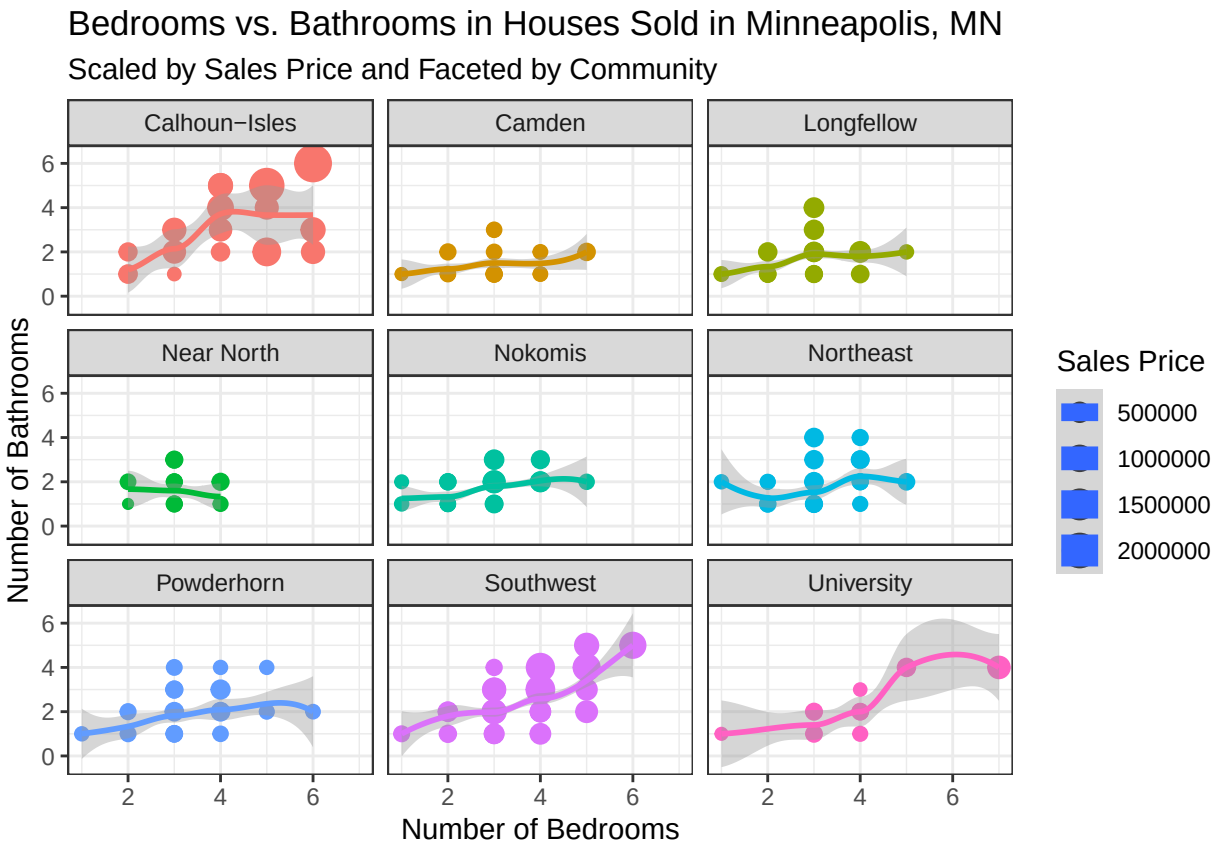
Question: What is the difference between the `color` and `fill` arguments?

Th `fill` argument provides a color that outlines the bars of the histogram, while the `color` argument fills the bars of the histogram.

(3) Develop an effective visualization on your own using the code chunk provided below. Use three variables and at least one aesthetic mapping.

```
ggplot(data = mn_homes,
       mapping = aes(x = beds, y = baths,
                     size = salesprice, color = community)) +
  geom_point() +
  geom_smooth() +
  facet_wrap(~ community) +
  guides(color = "none") +
  theme_bw() +
  labs(title = "Bedrooms vs. Bathrooms in Houses Sold in Minneapolis, MN",
       subtitle = "Scaled by Sales Price and Faceted by Community",
       x = "Number of Bedrooms", y = "Number of Bathrooms",
       size = "Sales Price")
```

```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```



Additional Resources

- https://ggplot2.tidyverse.org/
- https://raw.githubusercontent.com/rstudio/cheatsheets/master/data-visualization-2.1.pdf
- http://r-statistics.co/Top50-Ggplot2-Visualizations-MasterList-R-Code.html
- https://medium.com/bbc-visual-and-data-journalism/how-the-bbc-visual-and-data-journalism-team-works-with-graphics-in-r-ed0b35693535
- https://ggplot2-book.org/
- https://ggplot2.tidyverse.org/reference/geom_histogram.html
- https://rstudio.com/wp-content/uploads/2015/03/ggplot2-cheatsheet.pdf
- https://github.com/GraphicsPrinciples/CheatSheet/blob/master/NVSCheatSheet.pdf