Data Visualization II

STA199

1-18-2022

Main Ideas

- There are different types of variables.
- Visualizations and summaries of variables must be consistent with the variable type.

Coming Up

- Lab 2 due tomorrow.
- HW 1 goes out on Thursday.

Lecture Notes and Exercises

You will probably not need to do this any more at this stage, but if you do, please 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)
```

There are two types of variables **numeric** and **categorical**.

Types of variables

Numerical variables can be classified as either **continuous** or **discrete**. Continuous numeric variables have an infinite number of values between any two values. Discrete numeric variables have a countable number of values.

- height
- number of siblings

Categorical variables can be classified as either **nominal** or **ordinal**. Ordinal variables have a natural ordering.

- hair color
- education

Numeric Variables

To describe the distribution of a numeric we will use the properties below.

- shape
  - skewness: right-skewed, left-skewed, symmetric
  - modality: unimodal, bimodal, multimodal, uniform
- center: mean (`mean`), median (`median`)
- spread: range (`range`), standard deviation (`sd`), interquartile range (`IQR`)
- outliers: observations outside the pattern of the data

We will continue our investigation of home prices in Minneapolis, Minnesota.

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

Add a `glimpse` to the code chunk below and identify the following variables as numeric continuous, numeric discrete, categorical ordinal, or categorical nominal.

- area: numeric continuous
- beds: numeric discrete
- community: categorical nominal

```
glimpse(mn_homes$community)
```

```
##  chr [1:495] "Calhoun-Isles" "Longfellow" "Longfellow" "Southwest" "Camden" ...
```

```
glimpse(mn_homes$area)
```

```
##  num [1:495] 3937 1440 1835 2016 2004 ...
```

```
glimpse(mn_homes$beds)
```

```
##  num [1:495] 5 2 2 3 3 3 4 3 4 3 ...
```

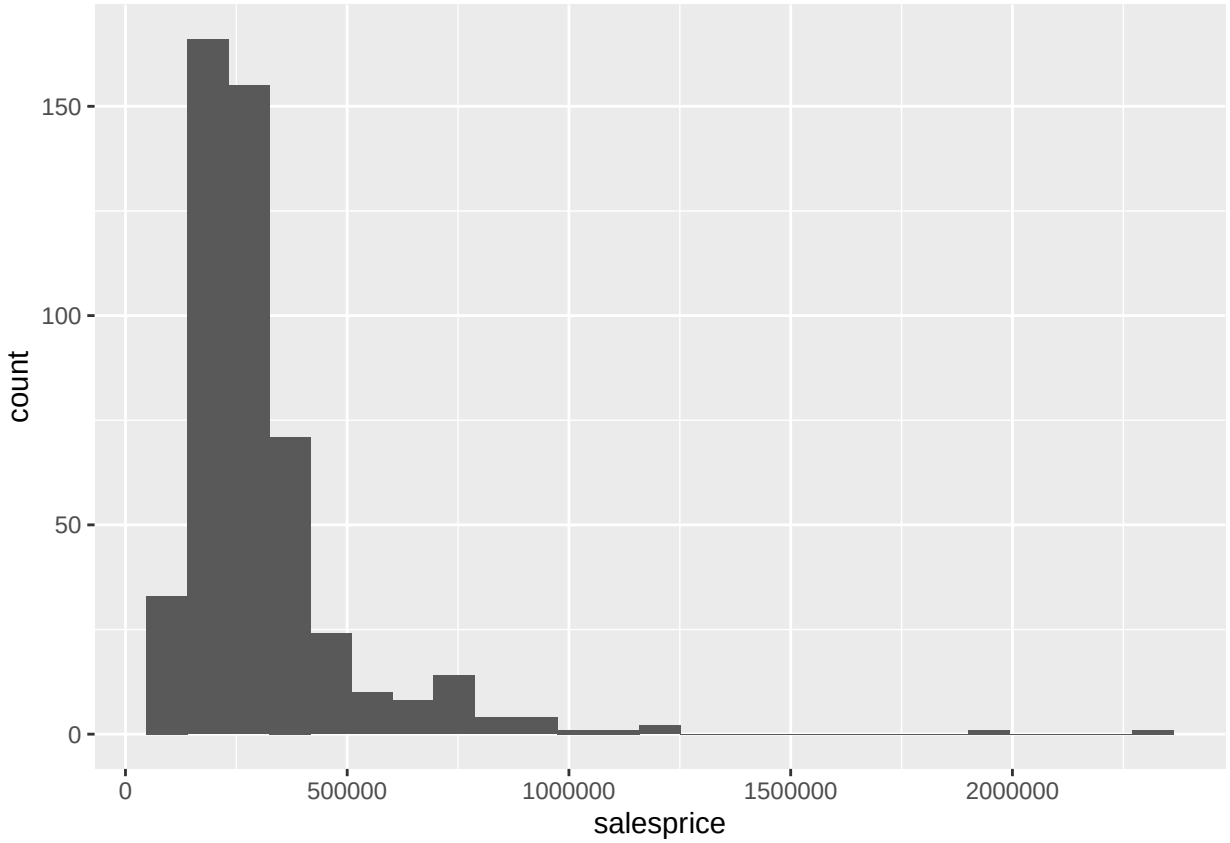
The `summary` command is also useful in looking at numerical variables. Use this command to look at the numeric variables from the previous chunk.

```
summary(mn_homes$beds)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   1.000   3.000   3.000   3.087   4.000   7.000
```

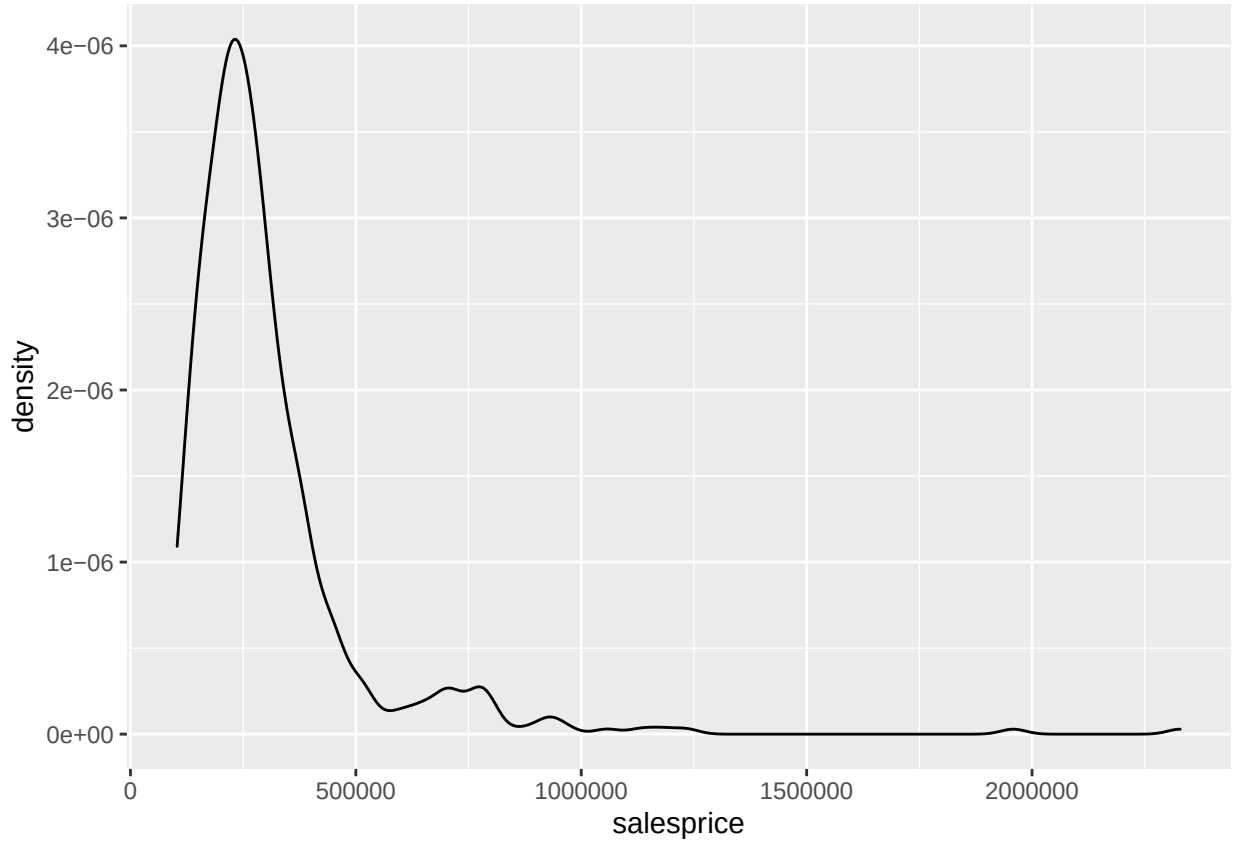
We can use a **histogram** to summarize a numeric variable.

```
ggplot(data = mn_homes,
       mapping = aes(x = salesprice)) +
   geom_histogram(bins = 25)
```



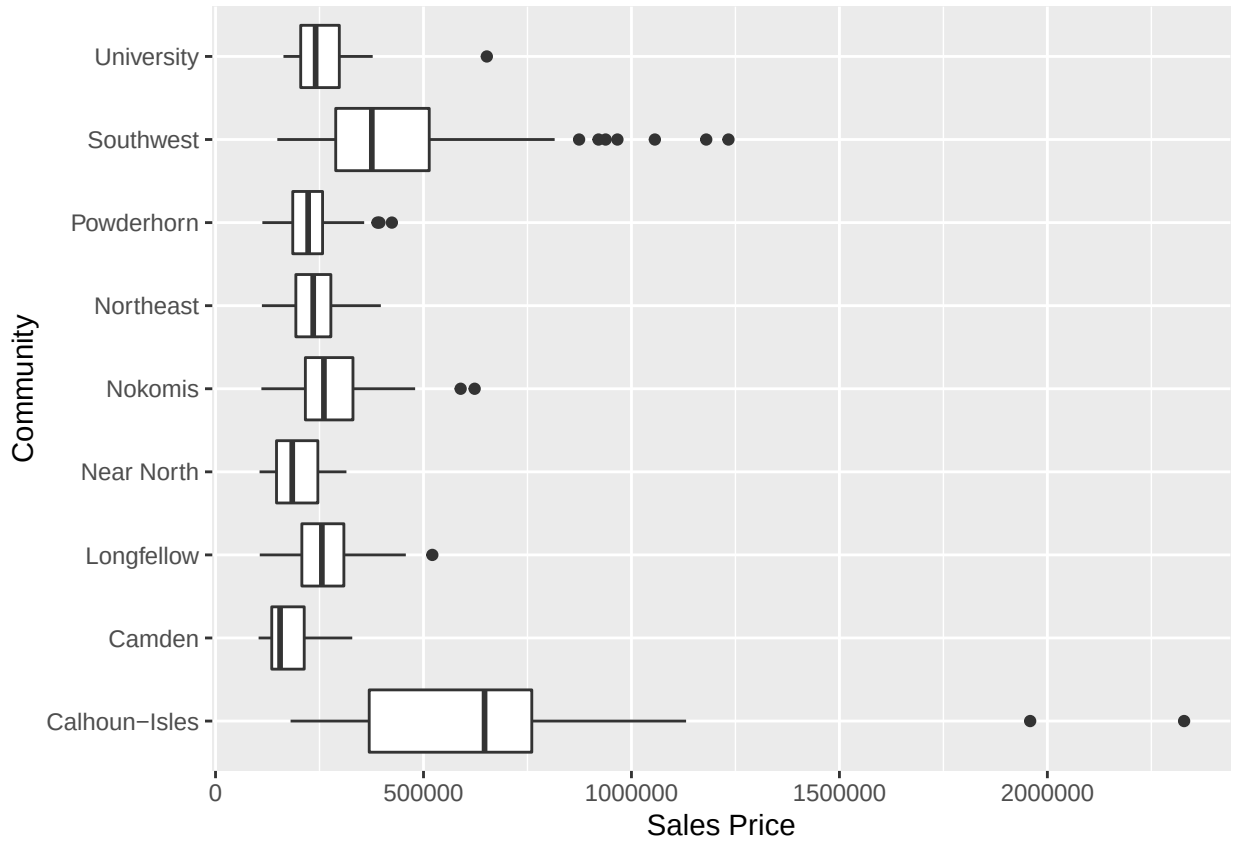
A **density plot** is another option. We just connect the boxes in a histogram with a smooth curve.

```
ggplot(data = mn_homes,
       mapping = aes(x = salesprice)) +
   geom_density()
```



Side-by-side **boxplots** are helpful to visualize the distribution of a numeric variable across the levels of a categorical variable.

```
ggplot(data = mn_homes,
       mapping = aes(x = community, y = salesprice)) +
       geom_boxplot() + coord_flip() +
       labs(main= "Sales Price by Community", x= "Community", y="Sales Price")
```

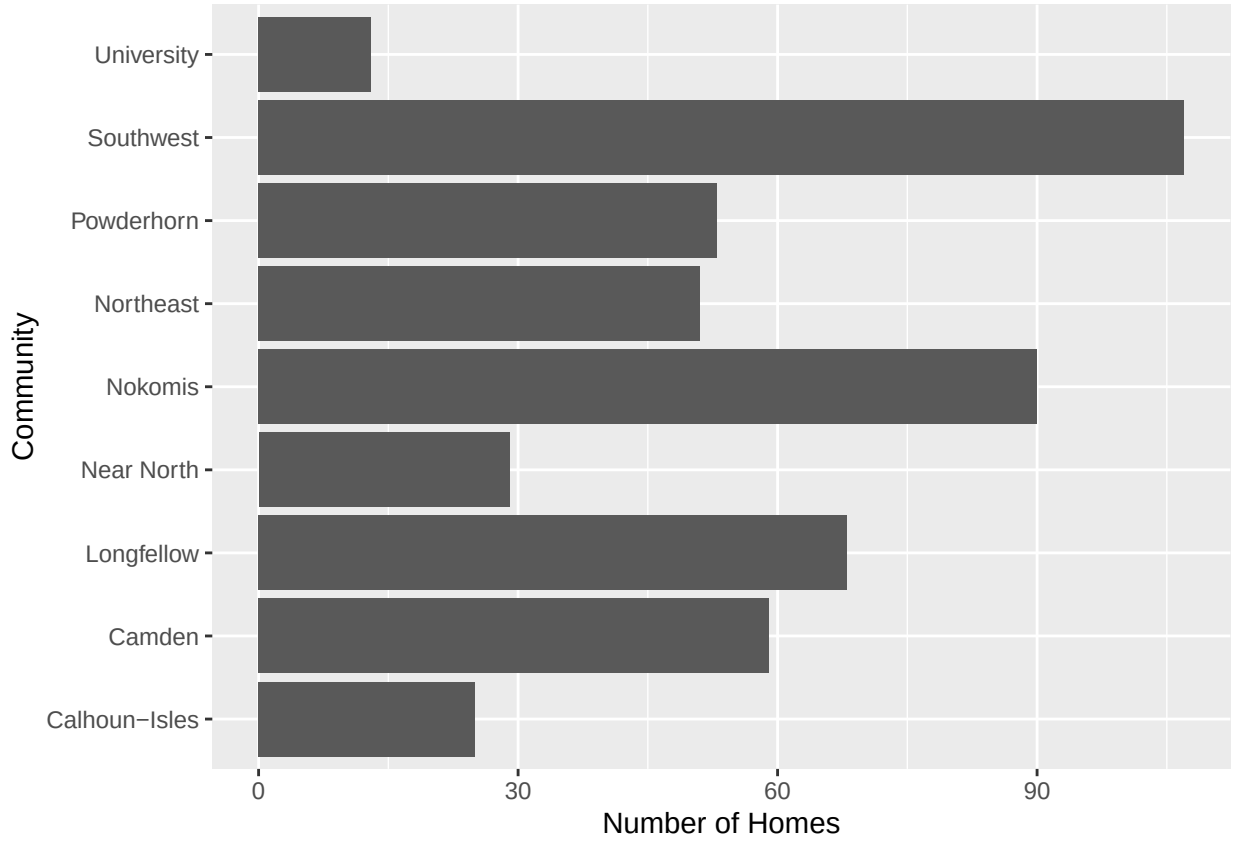


Question: What is `coord_flip()` doing in the code chunk above? Try removing it to see. `coord_flip()` ensures that the box plots are laid out horizontally, so they can be compared to a variable on the x-axis and stacked on top of one another. If it were absent, the box plots would instead be laid out vertically, side-by-side with one another and compared to a continuous variable on the y-axis.

Categorical Variables

Bar plots allow us to visualize categorical variables.

```
ggplot(data = mn_homes) +
  geom_bar(mapping = aes(x = community)) + coord_flip() +
  labs(main= "Homes by Community", x= "Community", y="Number of Homes")
```

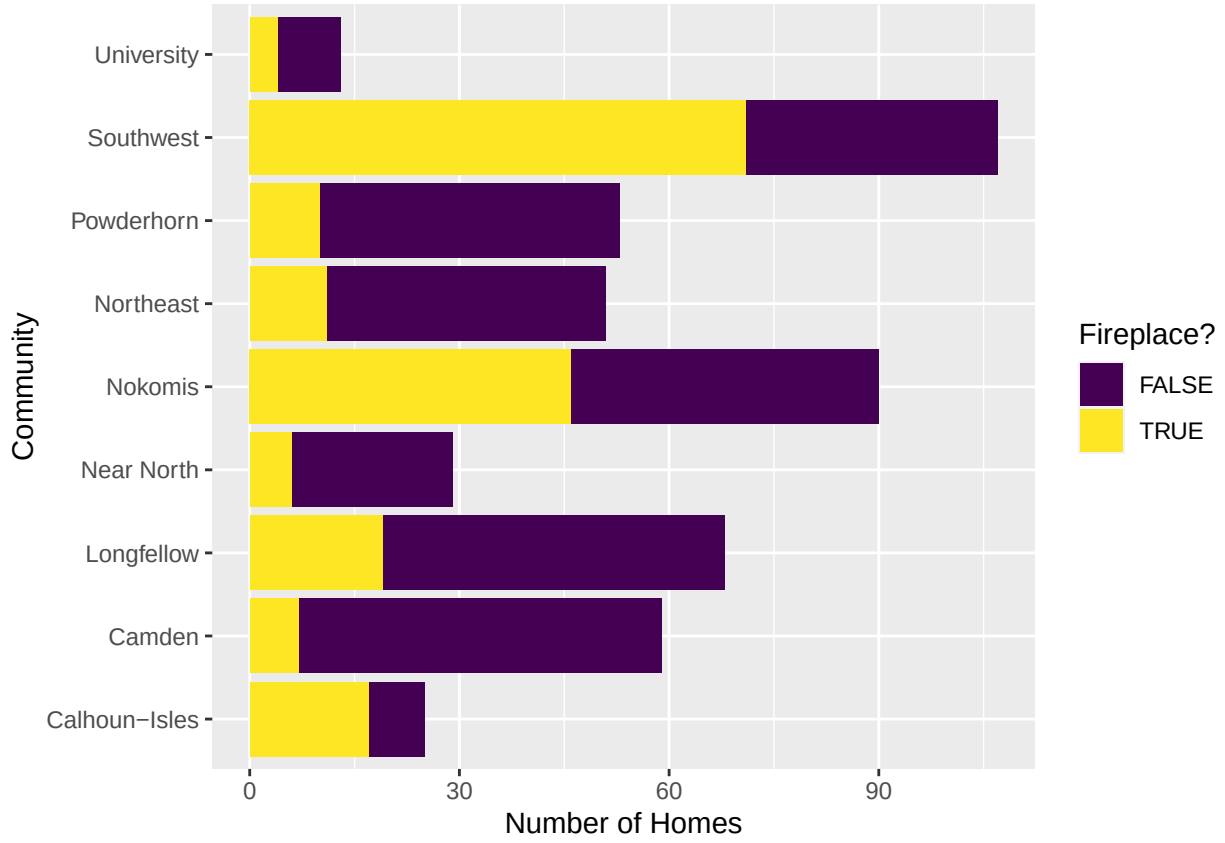


Segmented bar plots can be used to visualize two categorical variables.

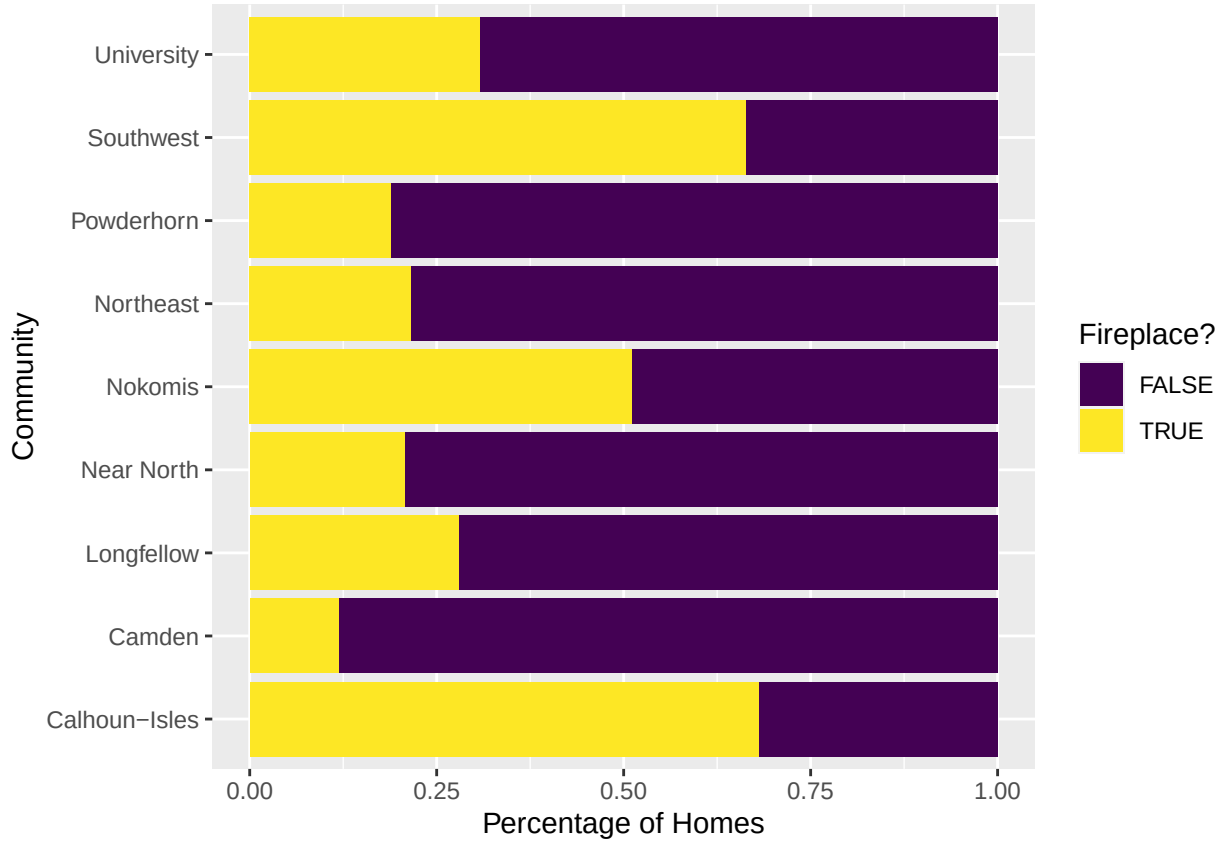
```
library(viridis)
```

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

```
ggplot(data = mn_homes, mapping = aes(x = community, fill = fireplace)) +
  geom_bar() +
  coord_flip() +
  scale_fill_viridis(discrete=TRUE, option = "D", name="Fireplace?") +
  labs(main= "Fireplaces by Community", x= "Community", y="Number of Homes")
```




```
ggplot(data = mn_homes, mapping = aes(x = community, fill = fireplace)) +
  geom_bar(position = "fill") + coord_flip() +
  scale_fill_viridis(discrete=TRUE, option = "D", name="Fireplace?") +
  labs(main= "Percentage of Homes with a Fireplace by Community", x=
  "Community", y="Percentage of Homes")
```



Question: Which of the above two visualizations do you prefer? Why? Is this answer always the same? I'm not sure if I could say I "prefer" one over the other - they show two quite different things. In the first graph, you get an understanding of how many homes there are in each community, and within that, how many of them have fireplaces. In the second graph, your point of comparison is not the number of homes in each community - which you do not know - but the relative proportion of homes in each neighborhood that has a fireplace. The second graph conveys less information, but it does enable much clearer cross-neighborhood comparison on the basis of whether or not homes have fireplaces - a worthy tradeoff, if that is your variable of interest.

There is something wrong with each of the plots below. Run the code for each plot, read the error, then identify and fix the problem.

```
ggplot(mn_homes, aes(x = lotsize, y = salesprice)) +
  geom_point(shape = 21, size = .85)
ggplot(data = mn_homes, mapping = aes(x = lotsize, y = area)) +
  geom_point(shape = 21, size = .85)
ggplot(data = mn_homes, mapping = aes(x = lotsize, y = area, color=community)) +
  geom_point(size = 0.85)
ggplot(data = mn_homes, mapping = aes(x = lotsize, y = area)) +
  geom_point()
```

General principles for effective data visualization

- keep it simple
- use color effectively
- tell a story

Why is data visualization important? We will illustrate using the `datasaurus_dozen` data from the `datasauRus` package.

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

```
glimpse(datasaurus_dozen)
```

```
## Rows: 1,846
## Columns: 3
## $ dataset <chr> "dino", "dino", "dino", "dino", "dino", "dino", "dino", "dino"~
## $ x       <dbl> 55.3846, 51.5385, 46.1538, 42.8205, 40.7692, 38.7179, 35.6410,~
## $ y       <dbl> 97.1795, 96.0256, 94.4872, 91.4103, 88.3333, 84.8718, 79.8718,~
```

The code below calculates the correlation, mean of y, mean of x, standard deviation of y, and standard deviation of x for each of the 13 datasets.

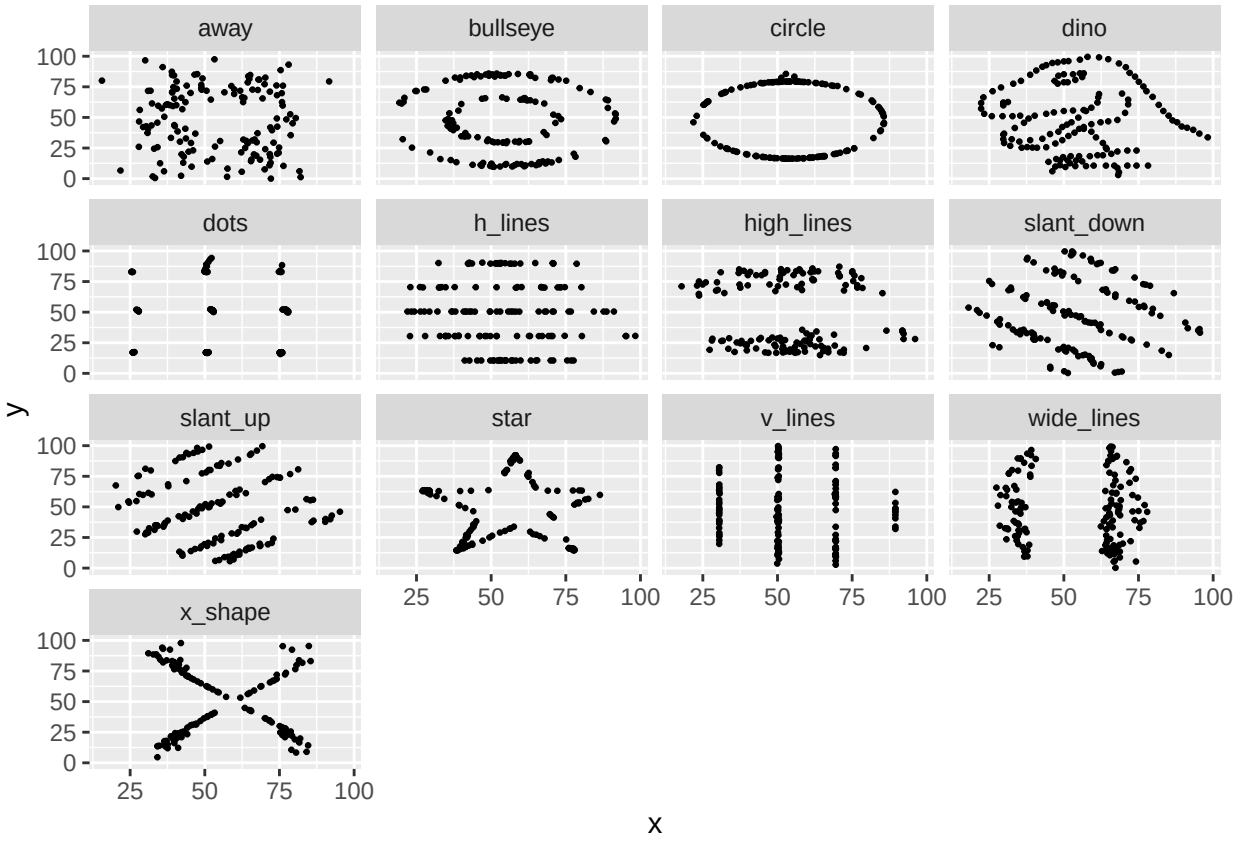
Question: What do you notice? Sneakily, they all have the same summary statistics - even when they (spoiler alert) aren't necessarily the same data at all.

```
datasaurus_dozen %>%
   group_by(dataset) %>%
   summarize(r = cor(x, y),
             mean_y = mean(y),
             mean_x = mean(x),
             sd_x = sd(x),
             sd_y = sd(y))
```

```
## # A tibble: 13 x 6
##    dataset          r mean_y mean_x  sd_x  sd_y
##    <chr>        <dbl>  <dbl>  <dbl> <dbl> <dbl>
##  1 away       -0.0641   47.8   54.3  16.8  26.9
##  2 bullseye   -0.0686   47.8   54.3  16.8  26.9
##  3 circle     -0.0683   47.8   54.3  16.8  26.9
##  4 dino       -0.0645   47.8   54.3  16.8  26.9
##  5 dots       -0.0603   47.8   54.3  16.8  26.9
##  6 h_lines    -0.0617   47.8   54.3  16.8  26.9
##  7 high_lines -0.0685   47.8   54.3  16.8  26.9
##  8 slant_down -0.0690   47.8   54.3  16.8  26.9
##  9 slant_up   -0.0686   47.8   54.3  16.8  26.9
## 10 star       -0.0630   47.8   54.3  16.8  26.9
## 11 v_lines    -0.0694   47.8   54.3  16.8  26.9
## 12 wide_lines -0.0666   47.8   54.3  16.8  26.9
## 13 x_shape    -0.0656   47.8   54.3  16.8  26.9
```

Let's visualize the relationships

```
ggplot(data = datasaurus_dozen,
       mapping = aes(x = x, y = y)) +
   geom_point(size = .5) +
   facet_wrap( ~ dataset)
```



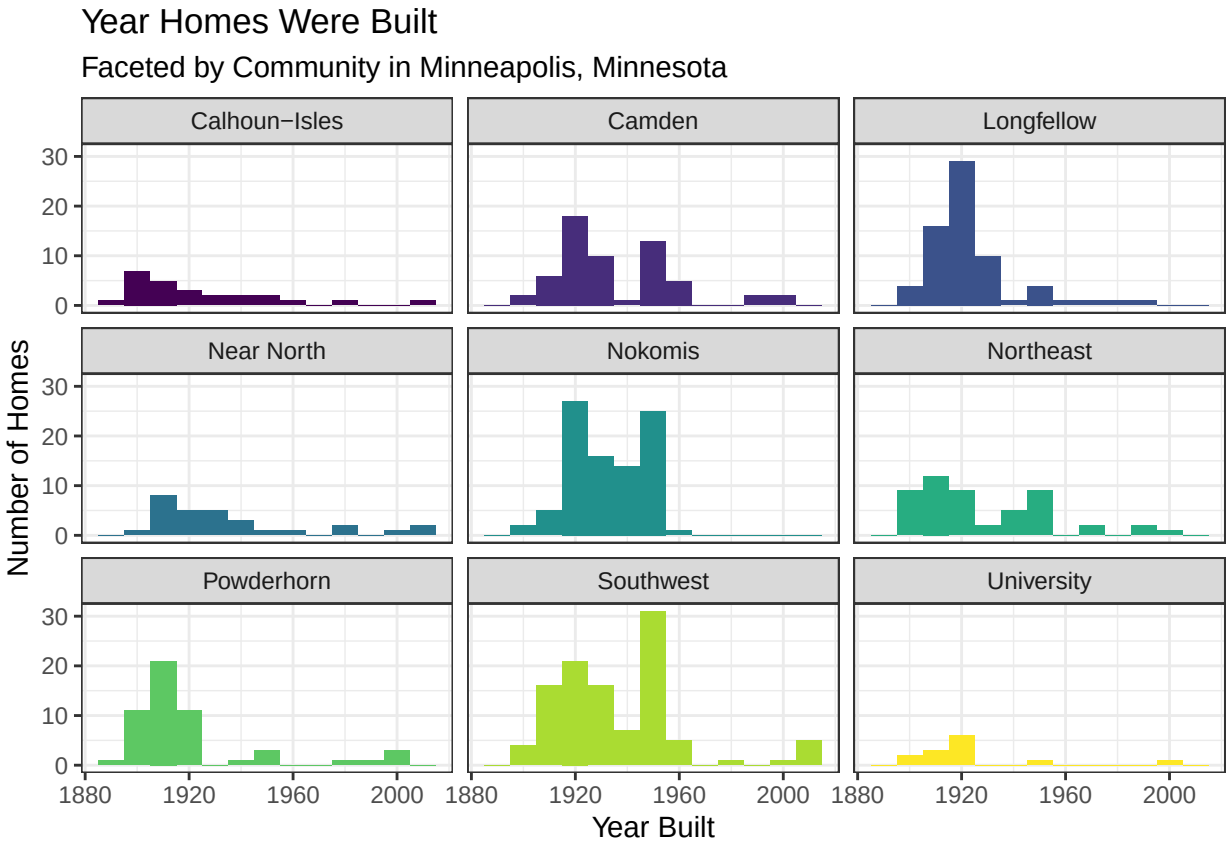
Question: Why is visualization important? Because data cannot entirely be defined by its summary statistics - it often has sneaky explanations lurking below the surface that need to be visualized to be understood.

Practice

(1) Modify the code outline to create a faceted histogram examining the distribution of year built within each community.

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

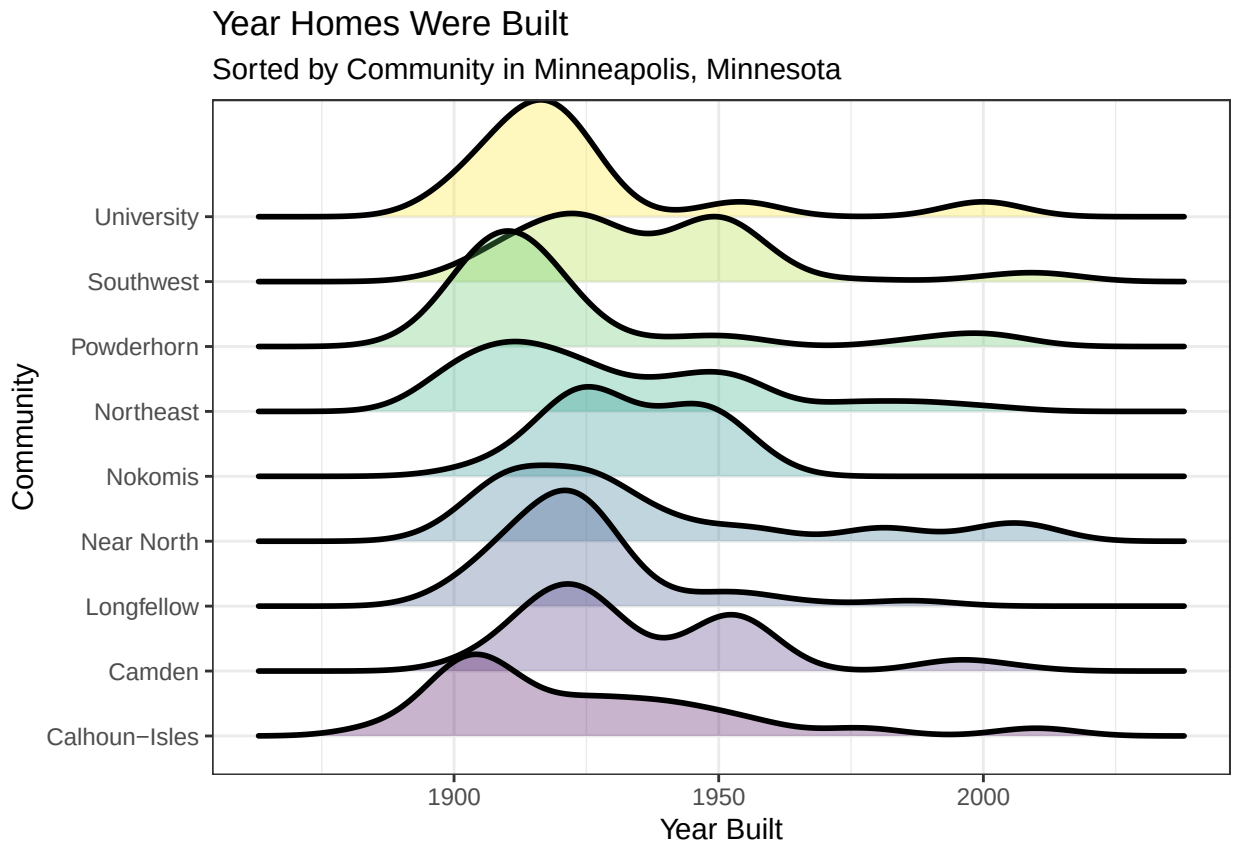
```
ggplot(data = mn_homes, mapping = aes(x = yearbuilt, fill = community)) +
  geom_histogram(binwidth = 10) +
  facet_wrap(~ community) +
  labs(x = "Year Built",
       y = "Number of Homes",
      title = "Year Homes Were Built",
      subtitle = "Faceted by Community in Minneapolis, Minnesota")+
  scale_fill_viridis(discrete = TRUE, guide="none")+
  theme_bw()
```




```
library(ggridges)

ggplot(mn_homes, aes(x = yearbuilt, y = community, fill = community)) +
  geom_density_ridges(alpha = 0.3, color = "black", lwd = 1) +
  scale_fill_viridis(discrete = TRUE, guide = "none") +
  theme_bw() +
  labs(x = "Year Built", y = "Community", title = "Year Homes Were Built",
       subtitle = "Sorted by Community in Minneapolis, Minnesota")
```

```
## Picking joint bandwidth of 7.64
```



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