ggplot tutorial

Basic syntax

First, we'll take a look at the basic syntax of ggplot and a few basic functions. Like dplyr, the basic syntax of ggplot is essentially a list of commands, with a new command on each line. Whereas in dplyr, you pipe operations together with "%>%", in ggplot you use the "+" operator. Another major difference between dplyr and ggplot is that the order of operations generally does not matter in ggplot (aside from the first command calling "ggplot").

To make a plot, the first step is to call ggplot by writing `ggplot(data=___, aes(x=___, y=___))`, where you input a dataframe, and then the name of the variable you want on the x-axis and y-axis.

Nothing will appear on the plot yet. The next step is to specify what kind of plot you want — a scatter plot, a line plot, a bar plot? To do that, the next command would be either `geom_point()`, `geom_line()`, `geom_col()`. Here's an example.

Simple scatter, line, and bar plots

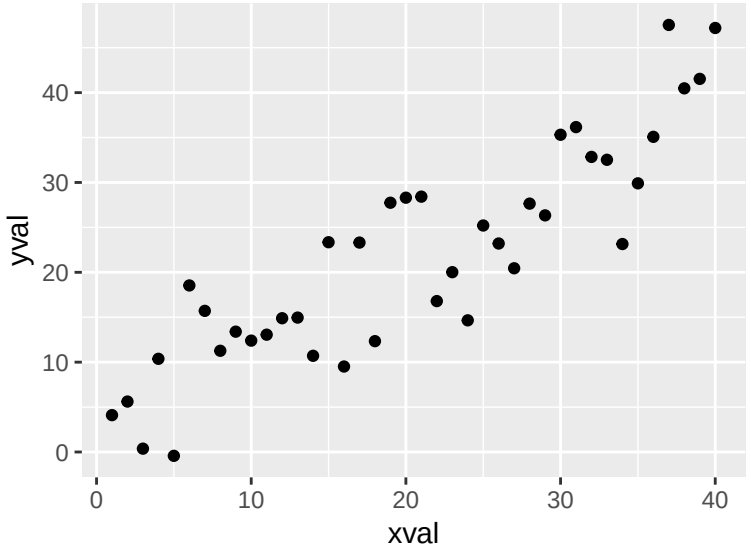
We'll first make some random data:

```
xval <- seq.int(1,40)
yval <- xval + rnorm(40,0,5)

data <- data.frame(xval,yval)

head(data) #only shows first 6 rows
```

```
##   xval       yval
## 1    1  4.1033637
## 2    2  5.6141077
## 3    3  0.3725056
## 4    4 10.3730385
## 5    5 -0.4232822
## 6    6 18.5389650
```

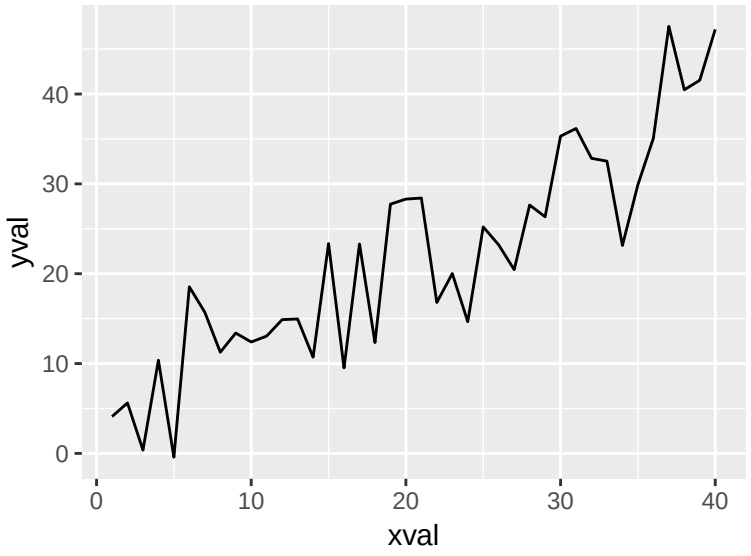
Here's a scatter plot:

```
ggplot(data=data, aes(x=xval, y=yval)) +
      geom_point()
```



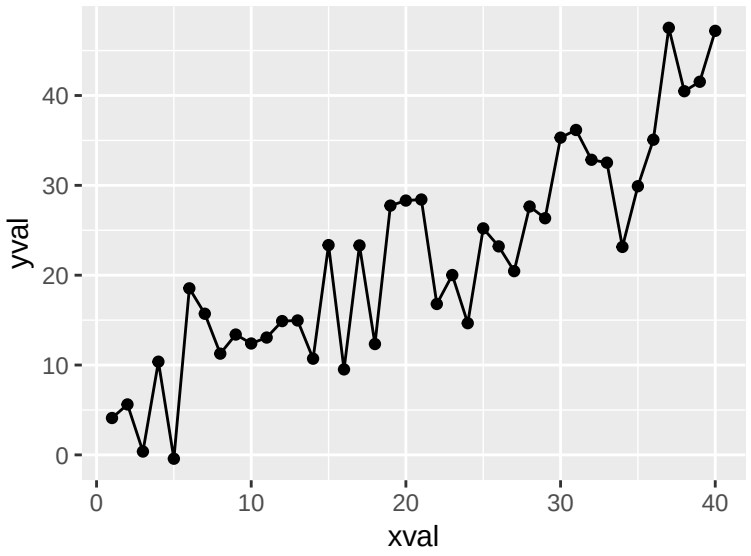
Here's a line plot:

```
ggplot(data=data, aes(x=xval, y=yval)) +
      geom_line()
```



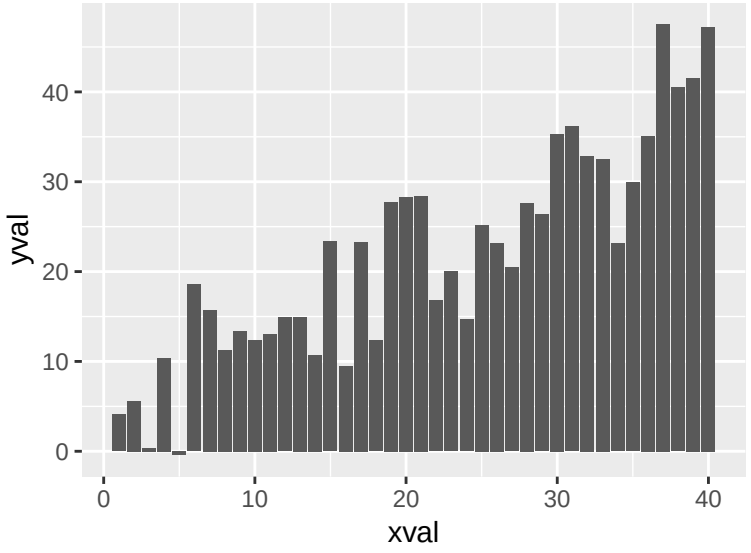
We could make both lines and points by writing:

```
ggplot(data=data, aes(x=xval, y=yval)) +
      geom_point() +
      geom_line()
```



Here's a bar plot:

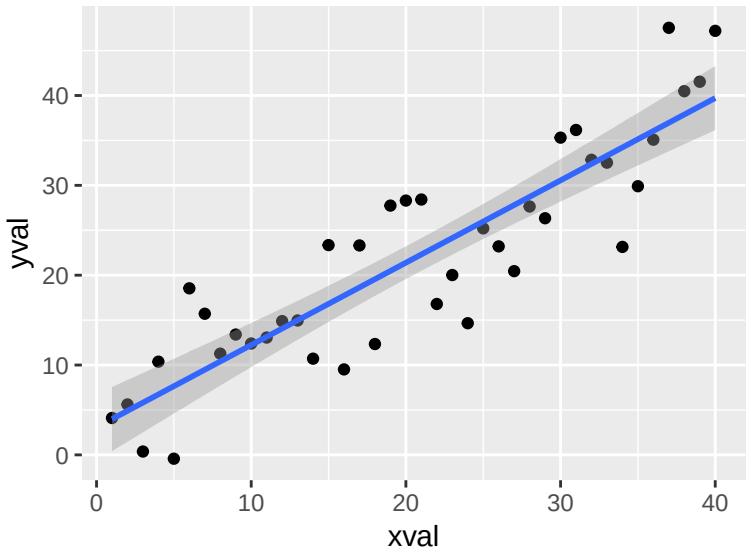
```
ggplot(data=data, aes(x=xval, y=yval)) +
      geom_col()
```



Adding a regression line

You can also put regression lines on your plots by running the `stat_smooth()` function. This function takes a "method" argument for the kind of regression. Below, I used `method="lm"` to specify that it's a linear model.

```
ggplot(data=data, aes(x=xval, y=yval)) +
      geom_point() +
      stat_smooth(method="lm")
```



Visualizing groups

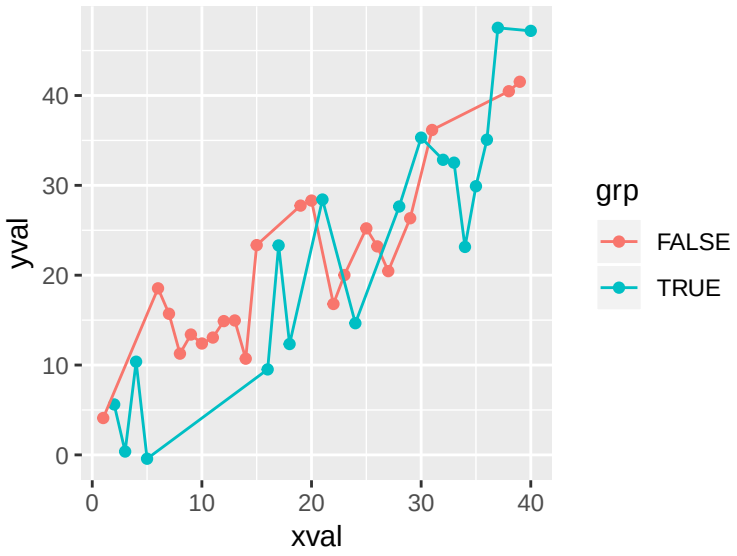
Now suppose the data came in *groups* and we want to visualize each group separately. For example, let's suppose we have multiple treatment groups with a single outcome measure. For simplicity, we'll just assign each point in our dataframe randomly to one of two groups.

```
grp <- rnorm(nrow(data),0,1) > 0
data$grp <- grp

head(data)
```

```
##   xval       yval   grp
## 1    1  4.1033637 FALSE
## 2    2  5.6141077  TRUE
## 3    3  0.3725056  TRUE
## 4    4 10.3730385  TRUE
## 5    5 -0.4232822  TRUE
## 6    6 18.5389650 FALSE
```

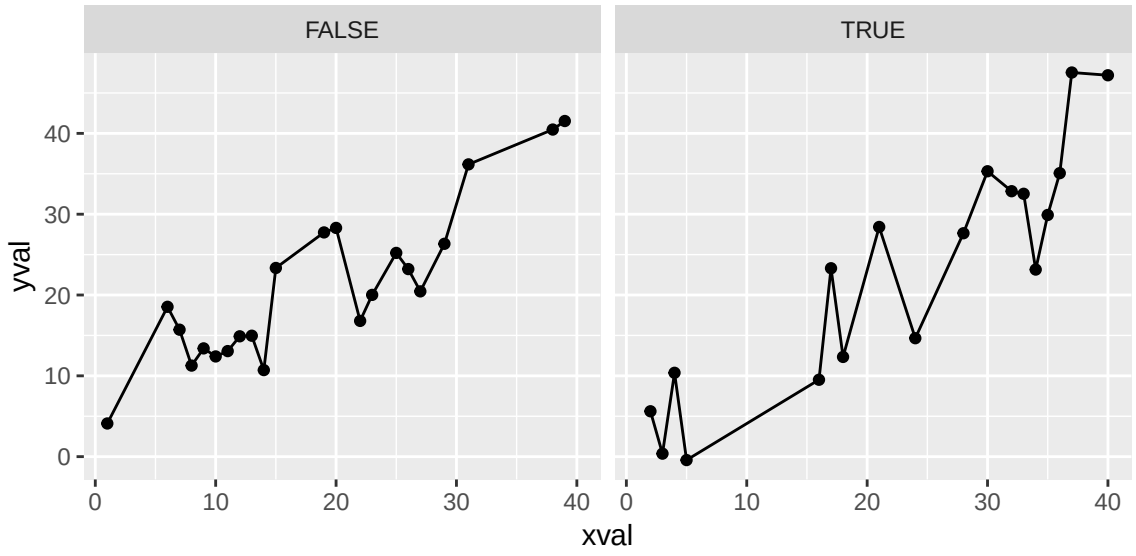
There are a couple basic ways to visualize this. The first is to put all the data on the same plot, but separate the groups visually, such as by making them a different color.

```
ggplot(data=data, aes(x=xval,y=yval,color=grp)) +
      geom_point() +
      geom_line()
```



Another common way to visualize each group separately is to make subplots or "facets" for each group. To do this, we use the function `facet_wrap`.

```
ggplot(data=data, aes(x=xval,y=yval)) +
      geom_point() +
      geom_line() +
      facet_wrap(~grp)
```



A couple other tips

- To change the name of the x- and y-axis titles, use `xlab("x-axis name")` and `ylab("y-axis name")`.
- To change the color of groups use `scale_color_manual(values=c("red","green",...))`.
- To change stylistic aspects of the plot that are not related to the data, such as the text size, the color of the background, etc..., you can modify the "theme". If you wanted to increase the size of the x-axis text, you could write `theme(axis.text.x = element_text(size=___))`.
- Check out the ggplot cheatsheet, which has a list of functions and when to use them: https://rstudio.com/wp-content/uploads/2015/03/ggplot2-cheatsheet.pdf

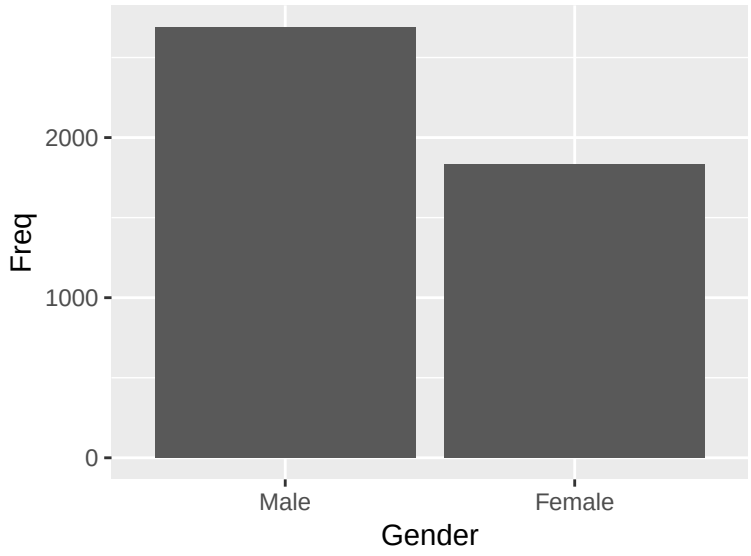
Your turn

Now it's your turn. We're going to look at the built-in R dataset showing UC Berkeley admissions of men and women from 1973, broken down by department. It comes from a lawsuit alleging a gender bias in admissions. Here you're going to figure out: were UC Berkeley admissions biased?

```
data(UCBAdmissions)
admit <- data.frame(UCBAdmissions)
```

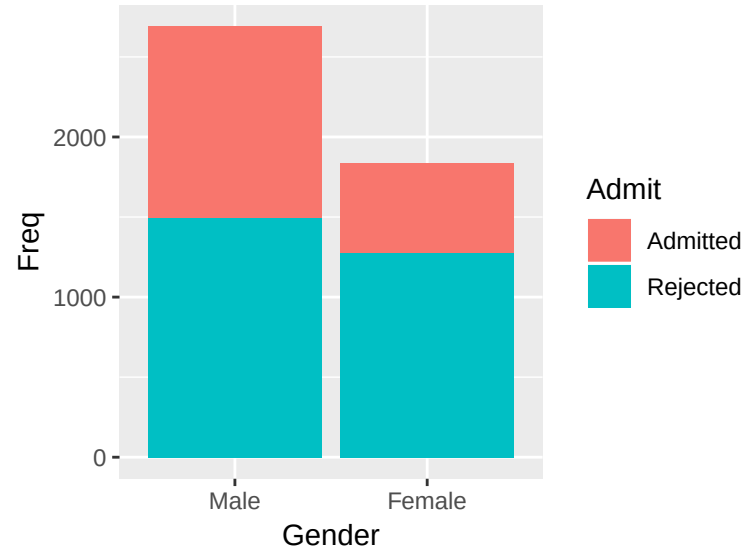
First, let's just look at how many men and women applied in total.

```
ggplot(data=admit, aes(x=Gender, y=Freq)) +
     geom_col()
```



Next, let's look at the proportion of admissions by gender.

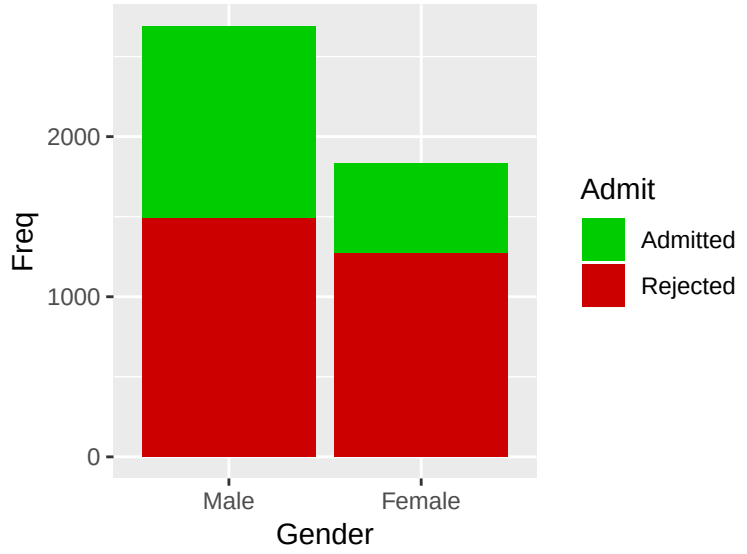
```
ggplot(data=admit, aes(x=Gender, y=Freq, fill=Admit)) +
      geom_col()
```



Ugh, the default color scheme makes 'admission' red and 'rejected' green, which is counter-intuitive. So before we go any further let's fix that (hint: `use scale_fill_manual()`). You can find color names at

http://www.stat.columbia.edu/~tzheng/files/Rcolor.pdf.

```
ggplot(data=admit, aes(x=Gender, y=Freq, fill=Admit)) +
      geom_col() +
      scale_fill_manual(values=c("green3", "red3"))
```

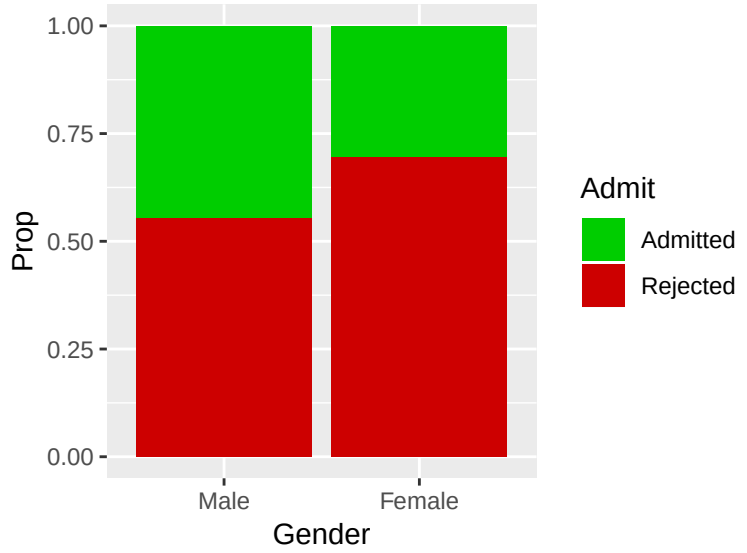


Looks pretty biased, but it's kind of hard to compare, because the total number of men and women who applied is different — so the bars are different heights. So let's turn these into proportions.

```
admit <- admit %>%
         group_by(Gender) %>%
         mutate(Prop = Freq/sum(Freq))
```

Now plot the proportions:

```
ggplot(data=admit, aes(x=Gender, y=Prop, fill=Admit)) +
      geom_col()  +
        scale_fill_manual(values=c("green3", "red3"))
```

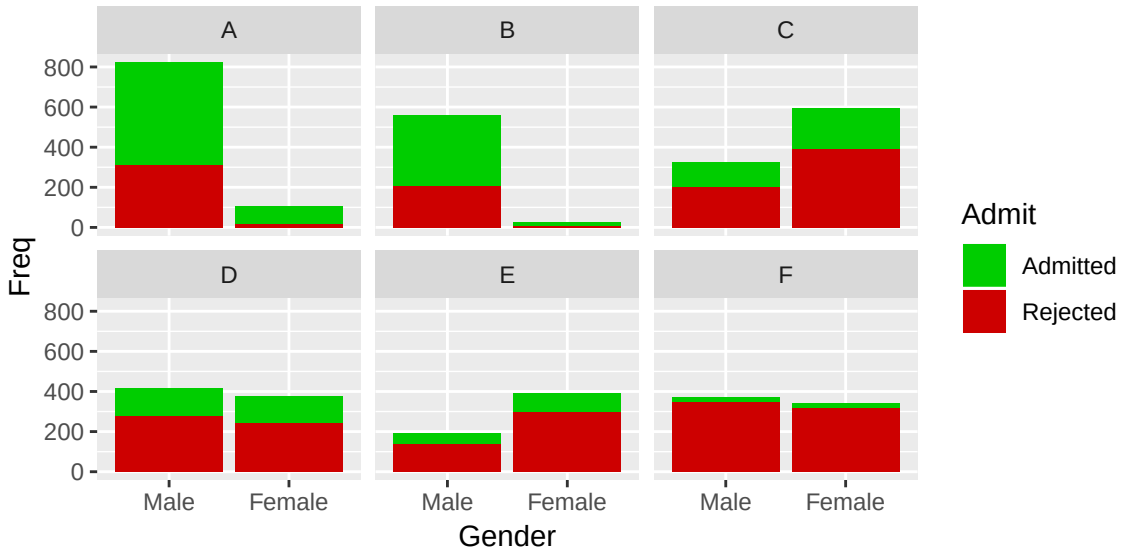


Should we conclude that the admissions are biased? The answer is no – at least, not yet. The reason is that there's a missing piece of information here, which is the admission rates by department. Let's start off with a

faceted plot of all the frequencies of admission by department (colored by gender).

Here we use the function *facet_wrap* to make subplots ("facets") for each department.

```
ggplot(data=admit, aes(x=Gender, y=Freq, fill=Admit)) +
      geom_col() +

      scale_fill_manual(values=c("green3", "red3")) +
      facet_wrap(~Dept)
```

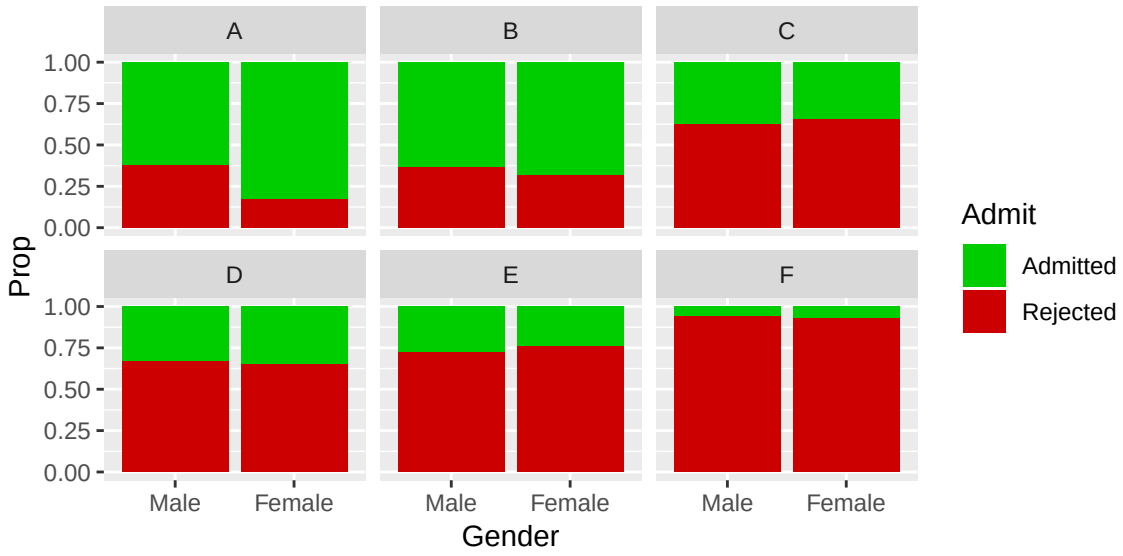


Again, because the total number of applicants of men and women varies by department, this is hard to interpret. So let's again turn each subplot into proportion instead of an overall frequency.

```
admit <- admit %>%
        group_by(Dept,Gender) %>%
        mutate(Prop=Freq/sum(Freq))
```

Now plot the proportions:

```
ggplot(data=admit, aes(x=Gender, y=Prop, fill=Admit)) +
      geom_bar(stat='identity') +
      scale_fill_manual(values=c("green3", "red3")) +
      facet_wrap(~Dept)
```

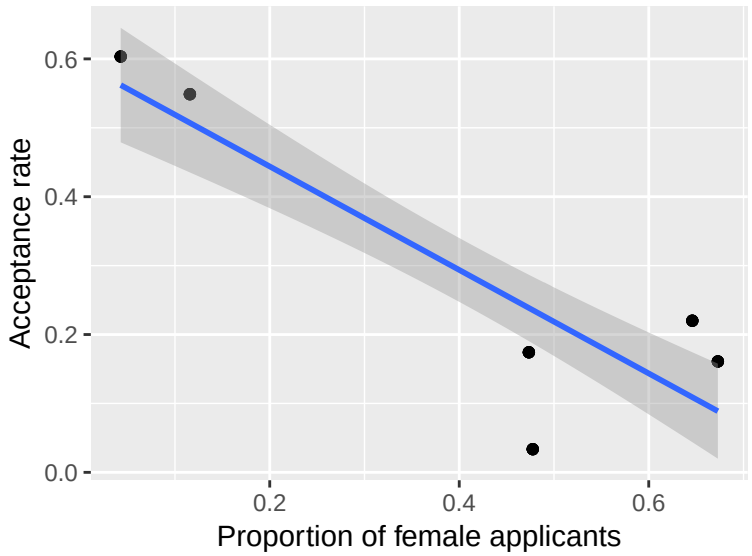


The data suddenly seem to tell an entirely different story. It seems that when you actually look at the proportion of men and women admitted to different departments, the rates of acceptance are essentially identical! This can be explained if women tend to apply to departments with lower acceptance rates. Let's demonstrate this a little more cleanly, by finding the proportion of women who applied to each department and plotting how selective those departments were overall.

```
admit <- admit %>%
        group_by(Dept, Admit) %>%
        mutate(Prop = (Admit=="Admitted") * Freq) %>%

        group_by(Dept, Gender) %>%
        mutate(Pct_Women = sum(Freq) * (Gender=="Female")) %>%

        group_by(Dept) %>%
        mutate(Pct_Women = max(Pct_Women)/sum(Freq)) %>%
        mutate(Prop=max(Prop)/sum(Freq))
```

So did women just apply to more selective departments?

```
ggplot(data=admit, aes(x=Pct_Women, y=Prop)) +
        geom_point() +
        xlab("Proportion of female applicants") +
        ylab("Acceptance rate") +
        stat_smooth(method="lm")
```



That was the legal conclusion. See: https://en.wikipedia.org/wiki/Simpson%27s_paradox