

On Titanium, you will see a dataset called *HairEyeColor*. Download the dataset and then import it to RStudio following the same procedure you used in Lab 1.

Once the data is input, open a new script (*Ctrl + Shift + N*) and type the following code in the script window, then highlight it and press *Ctrl + Enter* to run it:

```
summary(apply(HairEyeColor, 2, as.factor))
```

There's a lot going on in this command. If you remember from Lab 1, your categorical variables were imported as character variables (i.e. text), which is very nice in a lot of situations but a bit annoying here. This command converts every column in our dataset to a factor variable (remember, all factors are categorical!), then summarizes each column individually.

As a side note, actually type the commands in the script – don't use Copy/Paste. I do my best to eliminate any weird Word autoformatting, but sometimes it slips through and you might get a weird error message if you paste directly from Word.

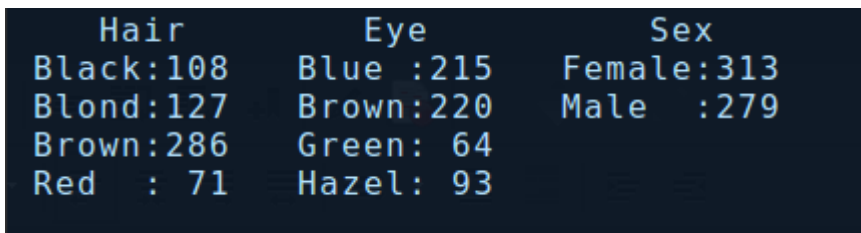
When you run the code (highlight it and then either click *Run* or press *Ctrl + Enter*), you should see it duplicated in the Console window:

```
> summary(apply(HairEyeColor, 2, as.factor))
```

For clarity, example code in the RStudio labs will include the **>** prompt to indicate separate lines of code. If a **+** sign appears at the beginning of a code line, that indicates that the line is continuing the previous command.

Below the code, you should see the output of the command.

Question #1 Copy or screenshot the Summary output and paste it below.



```
      Hair      Eye      Sex
Black:108  Blue :215  Female:313
Blond:127  Brown:220  Male  :279
Brown:286  Green: 64
Red  : 71  Hazel: 93
```

Question #2 How many variables are there in this dataset? Are the variables numerical or categorical? Specifically name one of the categorical variables and state its levels.

There are three main variables with sub variables. The variables are categorical as they describe the data.

Sex has two options; Male or Female. These sub variables have a quantity as well.

To find the number of rows in our dataset, we can use either of the following commands:

```
> nrow(HairEyeColor)
```

```
> dim(HairEyeColor)[1] # dim() gives the dimensions of the data frame; the  
first number gives the number of rows and the second gives the number of  
columns
```

Question #3 How many cases are there in this dataset?

There are 592 cases in this dataset.

Now let's graphically depict the eye colors using a bar graph. There are many different ways to produce plots in R. Most of these methods require the use of *packages*, which contain additional commands that extend the functionality of R (you can think of them as free DLC). To create our plots, we're going to use the `ggplot2` package, which is commonly used in "data science." Check the [Packages](#) tab to see if `ggplot2` is already installed. If not, click [Install](#) in the tab and type `ggplot2` in the prompt. Then type the commands (in the script window, then [Ctrl + Enter](#) to run them so the Console looks like the text below):

```
> library(ggplot2) # loads the ggplot2 package; run this at the beginning  
of the console session, or write it at the top of the script  
> eye_plot <- qplot(HairEyeColor$Eye, geom= "bar") # creates a barplot of  
the values of Eye color  
> ?qplot # brings up the help file for the qplot command, in the Help tab  
in the bottom right
```

Let's stop and examine what's going on in this code. The first command we ran loads the `ggplot2` package, which we need to do before we can use any of its functions, and there's a comment to remind us to load it at the top of the script if we use any `ggplot2` functions in the script.

Next, the `<-` sequence is used to assign a variable name to the output of the code. In this case, we use `<-` to store our counts in the variable `eye_plot`. Typing those two characters over and over can get annoying, so we can use the shortcut [Alt + -](#) instead. There are a bunch of different conventions for naming variables in R. The most common and accepted convention is to write a descriptive name for the variable, and if the name has multiple words, connect the words with either an underscore (`_`) or a period (`.`).


Next, the `$` indicates to select the `Eye` column from `HairEyeColor`. There are a variety of ways to use the `$` operator, but they mostly correspond to "find the variable with this specific name in my data set or output."

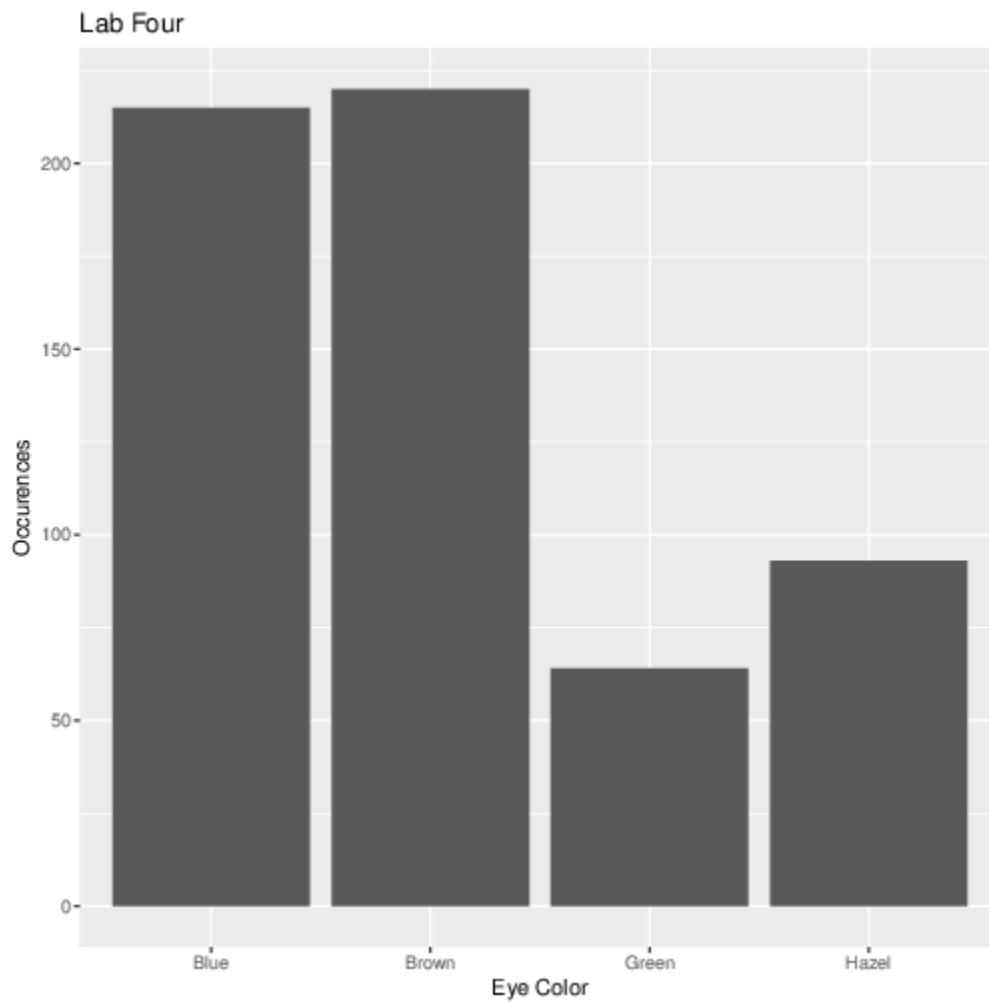
Lastly, you will notice that no plot was actually created. If we didn't store the plot as a variable, we would see a plot in the [Plots](#) window in the bottom right. But because of how the `ggplot2` package works, we can make modifications to the plot before we actually see it. We're going to modify the plot by adding a title and axis labels:

```
> eye_plot_labeled <- eye_plot + labs(title= "Plot Title", x = "x-axis  
label", y = "y-axis label") This text is here to make sure you get an error  
message if you're copy/pasting; fix your title and axis labels, then delete  
or comment out this sentence.
```

Once you're done adding and modifying things in your plot, print it to the [Plots](#) tab in the bottom right:

```
> print(eye_plot_labeled)
```

Question #4 Copy the graph (In the Plots tab, [Export](#)  [Copy to Clipboard](#)) and paste it below.



Question #5 Which category has the most people? Which has the least?

There are more people who have brown eyes with a 37.20% and green eyes with 10.8%

Now we're going to add some extra stuff to the plot. First, let's get the actual counts. We'll do this by installing the dplyr library and using some functions:

```
> library(dplyr) # the function we need is in the dplyr library, which may  
already be installed. If not, install it using the Packages tab.  
> eye_counts <- count(HairEyeColor, Eye)
```

Now let's get the actual percentages, rounded to the nearest tenth of a percent:

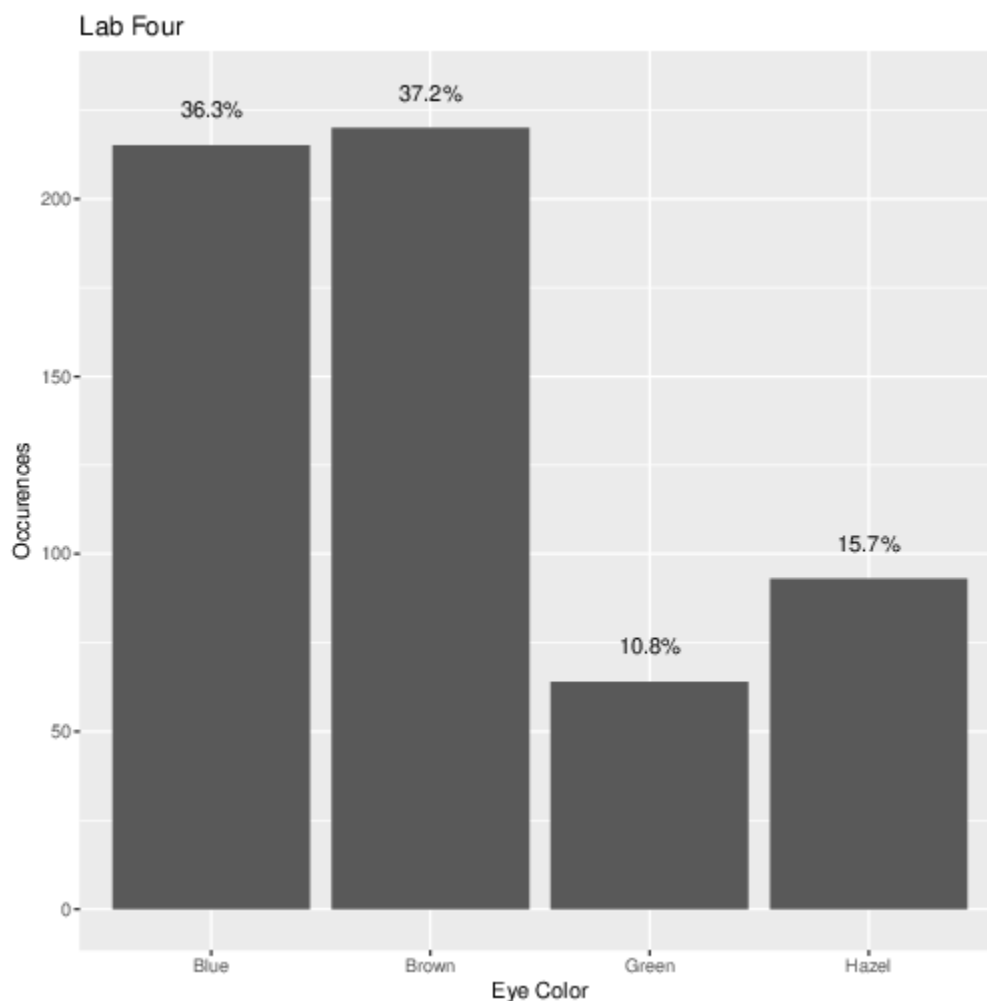
```
> eye_pcts <- eye_counts$n/sum(eye_counts$n) * 100  
> eye_pcts <- round(eye_pcts, 1)
```

And add the text percentages to the plot:

```
> eye_plot_pcts <- eye_plot_labeled + annotate("text", x = seq(1,4), y =  
eye_counts$n+10, label = paste(eye_pcts, "%", sep = ""))
```

Once again, there's a whole lot going on in these commands. Try to figure out as best you can what's going on by looking at the help files for the [annotate](#), [seq](#), and [paste](#) commands (for instance, [?annotate](#)).

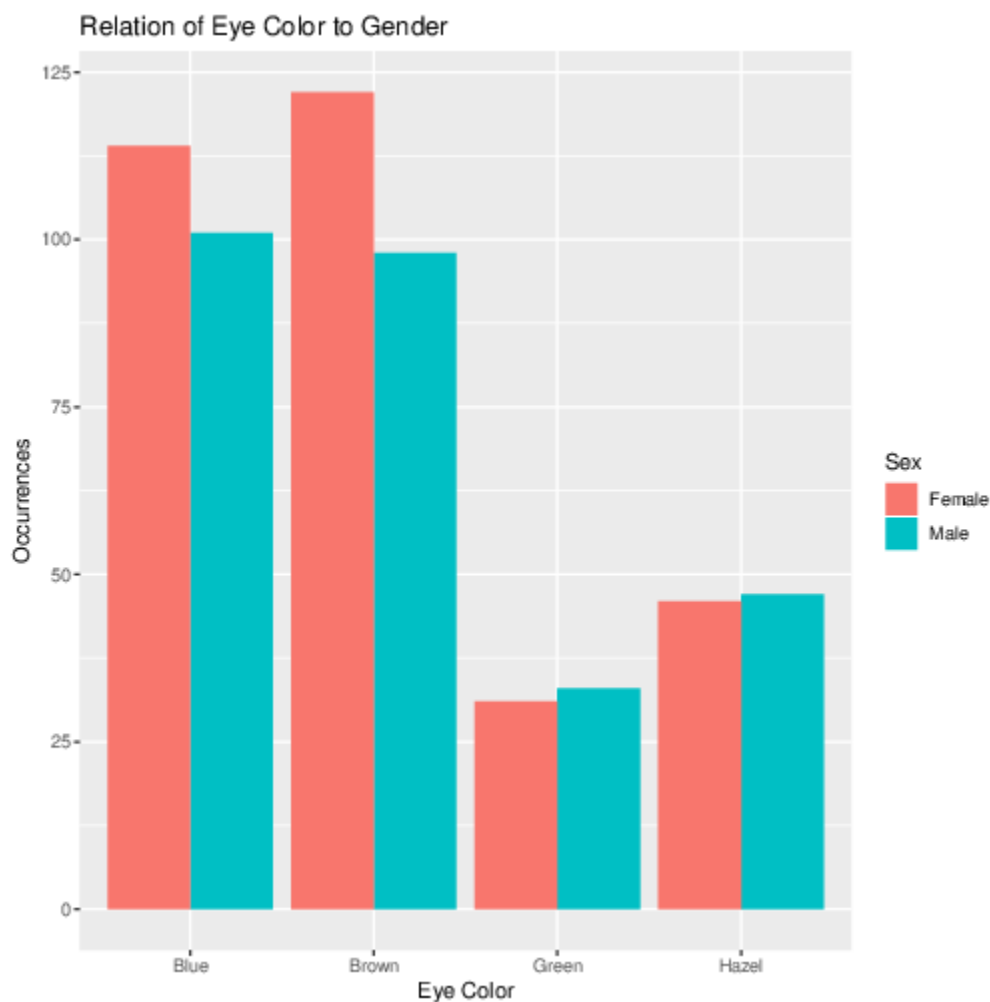
Question #6 Print the new graph to the [Plots](#) tab, then copy it and paste it below.



Now let's start looking at the other plotting command in the ggplot2 library: **ggplot**. This command is a lot more complicated but also a lot more versatile. We'll use the example of looking at the relationship between eye color and gender, so we want to visually compare the distribution of Eye Color of males and females. Type the code below, but don't forget to change the plot and axis titles to something useful before running it!

```
> eye_gender_plot <- ggplot(data = HairEyeColor, mapping = aes(x = Eye))
> eye_gender_barplot <- eye_gender_plot + geom_bar(aes(fill = Sex),
position = "dodge") + labs(title = "Plot Title", x = "x-axis label", y =
"y-axis label") AGAIN, IF YOU ARE COPY/PASTING, FIX THE LABELS AND THEN
DELETE OR COMMENT THIS SENTENCE!
> print(eye_gender_barplot)
```

Question #7 Copy the new graph and paste it below.



Question #8 Which color is most prevalent for females; which color for males?

Most prevalent for females is brown eyes and for males it is blue.