Module 2: Food Gone Bad

Keaton Wilson, modified by EKB and Jeremy McWilliams

Introduction to Plotting

Student Learning Outcomes

1. Students will be able to apply basic data science knowledge to find the cause of a real-world scenario–food poisoning!
2. Students will be able to use ggplot to generate histograms, scatter plots, and bar charts from data sets, and save them to image files.
3. Students will be able to use visual-thinking skills to create visualizations that allow them to explore patterns in data, draw inferences, and create solutions.

Introduction to the problem

We have a wave of people getting sick across the team. People are coming in complaining of stomach sickness. Doctors have ruled out a communicable viral infection like norovirus, so it seems likely to be a food contamination issue.

The two main sources of food that are grown on site and distributed to team members are plants grown in hydroponic greenhouses (mostly Swiss chard, cucumbers and radishes) and fish (tilapia, a tolerant warm-water species, and rainbow trout, a cold-water species). Team members vary in the composition of their diet; people are allowed to choose how much of different food sources they eat.

Fortunately, we have some data to work with! We have data on the following:

- which team members are sick
- how much fish or plant material they incorporate into their diets

Group Discussion

How might we go about trying to figure out what is causing the problem?

The Data

First we're going to pull in the data and give it a quick inspection/exploration before we start to work through some of the visualization tools in R.

```
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.2 --
## v ggplot2 3.3.6      v purrr   0.3.4
## v tibble  3.1.8      v dplyr   1.0.10
## v tidyr   1.2.0      v stringr 1.4.1
## v readr   2.1.2      v forcats 0.5.2
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
sick<-read_csv("sick_data.csv")
```

```
## Rows: 349 Columns: 10
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (4): last, first, sex, specialties
## dbl (6): age, height_cm, weight_kg, perc_fish, perc_plant, doctor_trips
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```
```
sick
```

```
## # A tibble: 349 x 10
##    last          first sex     age heigh~1 weigh~2 speci~3 perc_~4 perc_~5 docto~6
##    <chr>         <chr> <chr> <dbl>   <dbl>   <dbl> <chr>     <dbl>   <dbl>   <dbl>
##  1 Gonzalez      Ange~ M        35    169.    51.4 Hydrol~   0.994 0.00620       5
##  2 Navratil      John  M        19    112.    96.3 Geneti~   0.297 0.703         1
##  3 Duff          Josh~ M        26    133.    52.1 Hortic~   0.514 0.486         2
##  4 Dottson       Juli~ M        36    140.    52.6 Climat~   0.686 0.314         3
##  5 al-Sultana    Mune~ M        26    194.    52.2 Geology   0.292 0.708         1
##  6 Gallegos P~   Rich~ M        29    153.    98.1 Climat~   0.329 0.671         1
##  7 Walker        Trev~ M        33    190.   102   Psycho~   0.558 0.442         3
##  8 Hau           Kin   M        22    118.    41.1 Psycho~   0.237 0.763         1
##  9 Zhong         Tobby M        21    121.    49.3 Mechan~   0.427 0.573         2
## 10 Scherschli~   Rona~ M        36    168.    62.7 Manage~   0.529 0.471         2
## # ... with 339 more rows, and abbreviated variable names 1: height_cm,
## #   2: weight_kg, 3: specialties, 4: perc_fish, 5: perc_plant, 6: doctor_trips
```
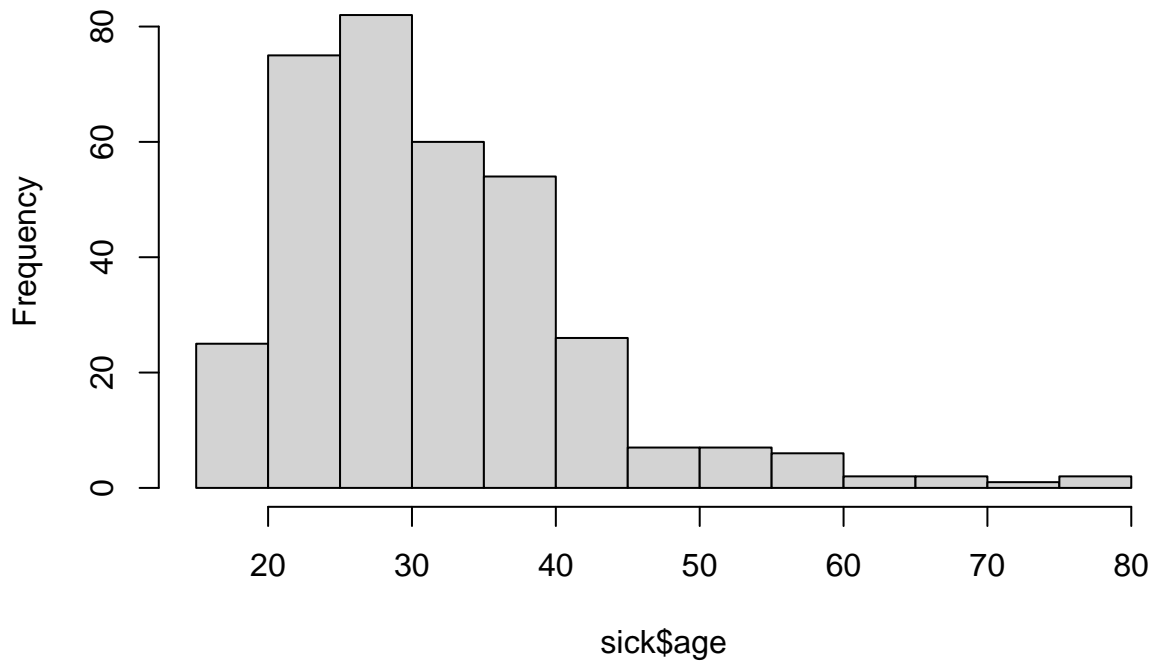
Histograms

We're going to look at a couple ways to create a histogram. The first is with the base R function `hist`, and the second is with `ggplot` (from the `tidyverse`). From today forward we're going to exclusively use ggplot for data visualizations, as it's the standard in R/data science.

Let's first look at the `hist` function.

```
hist(sick$age)
```

Histogram of sick$age

Now let's create a histogram with the same data using ggplot. First, let's talk a little about ggplot:

What is ggplot?

ggplot is an R package, and is part of the tidyverse. The "gg" translates to "grammar of graphics", and is founded in the idea that all data visualizations are comprised of three components:

- data set
- aesthetics, or visual marks that represent the data (i.e. the stuff that you see)
- geometric objects, or "geoms" (e.g the type of plot)

In the files tab (lower-right window), take a look at the ggplot-cheatsheet.pdf. In addition, there's lots of documentation on the internet on ggplot.

```
# Data set: "data=sick"              (the data set)
# Aesthetics: "mapping=aes(x=age)"   (the stuff you see)
# Geoms: "geom_histogram()"          (what type of plot)

# Notice that the "+" is used to chain functions together in ggplot

ggplot(data=sick, mapping=aes(x=age))+
  geom_histogram(bins=12)
```
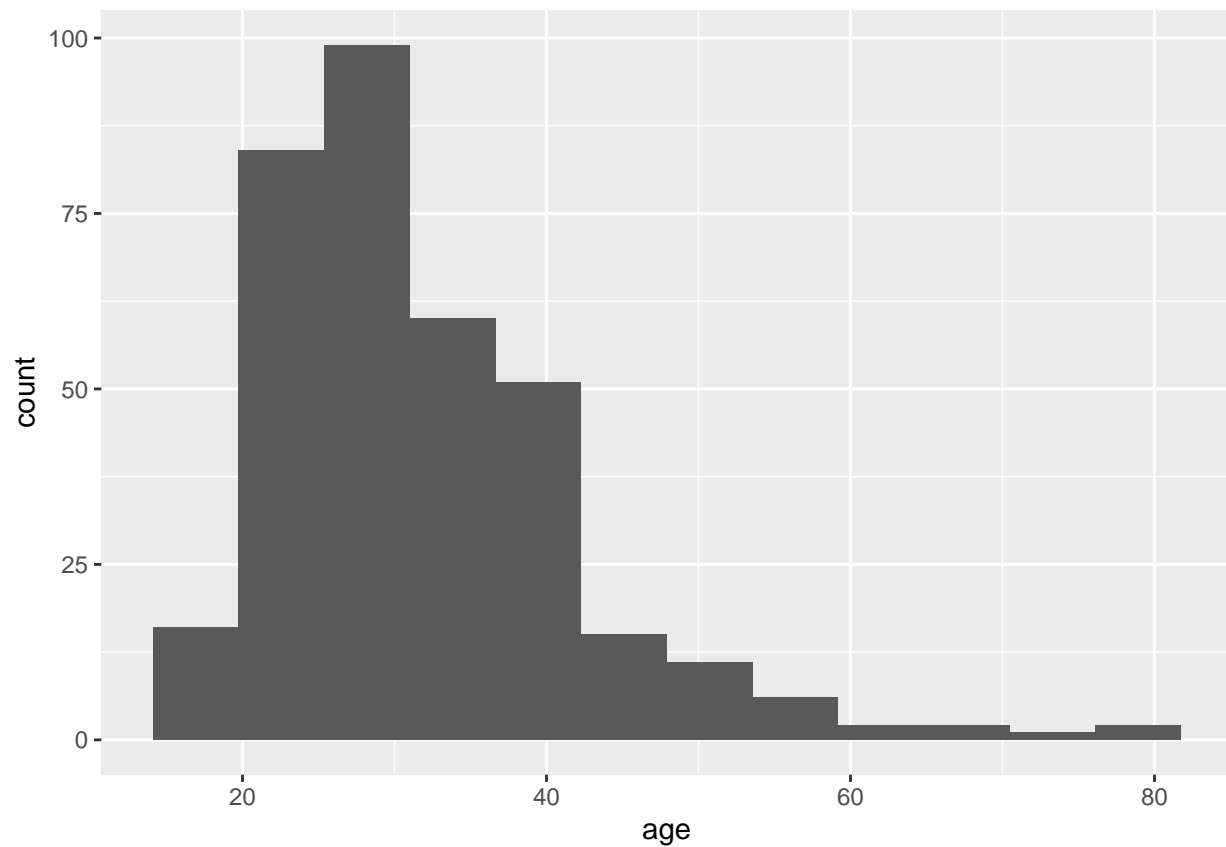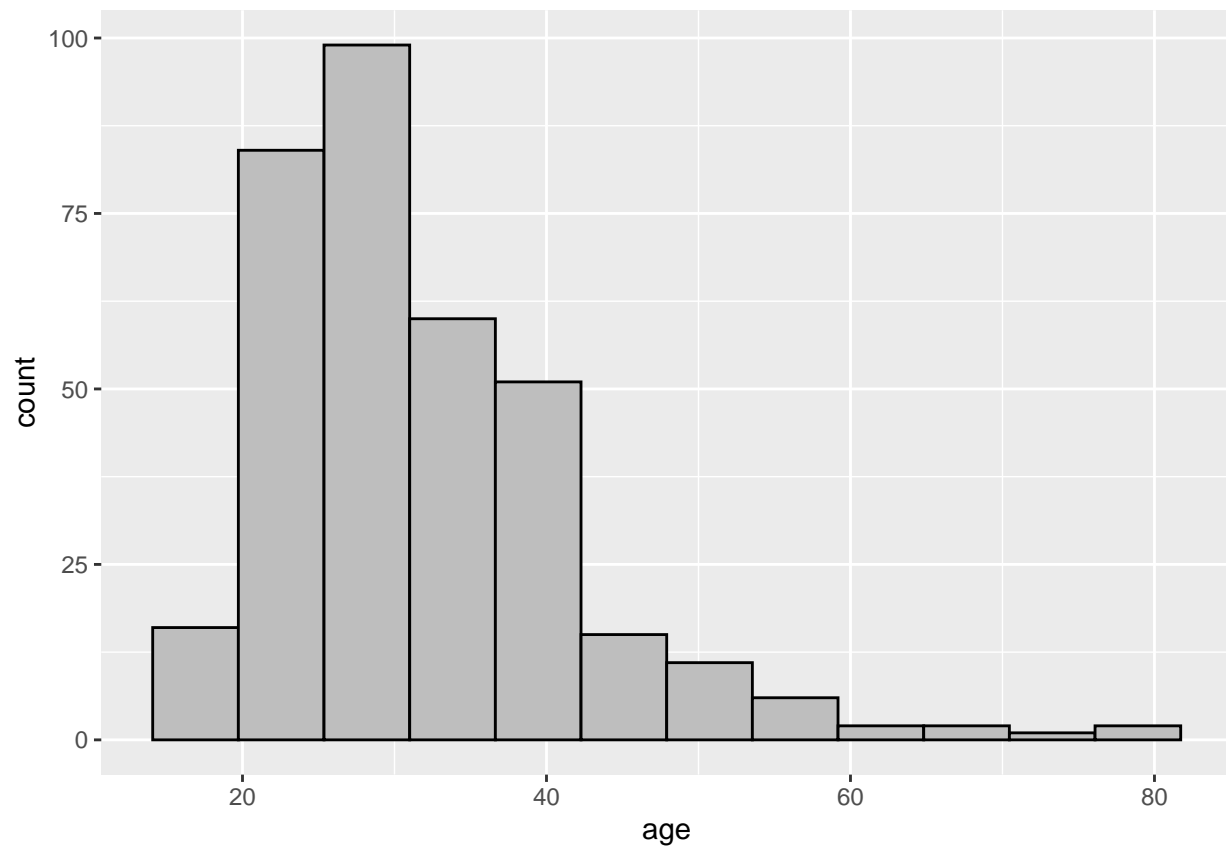
```
# 'bins' refers to the number of ranges data can fall in
```

We can improve the look by adding the "color" and "fill" attributes to geom_histogram. `color` indicates the outline color, and `fill` specifies the background color:

```
ggplot(data=sick, mapping=aes(x=age))+
  geom_histogram(bins=12, color="black", fill="grey")
```

One of the keys to data visualization is making sure the data and units are very clear to the viewer. We can chain on the labs() function to create custom axis labels and title:

```
ggplot(data=sick, mapping=aes(x=age))+
  geom_histogram(bins=12, color="black", fill="grey")+
  labs(title="Distribution of Team Anarctica Ages", x="Age (years)", y="Frequency")
```

Distribution of Team Anarctica Ages
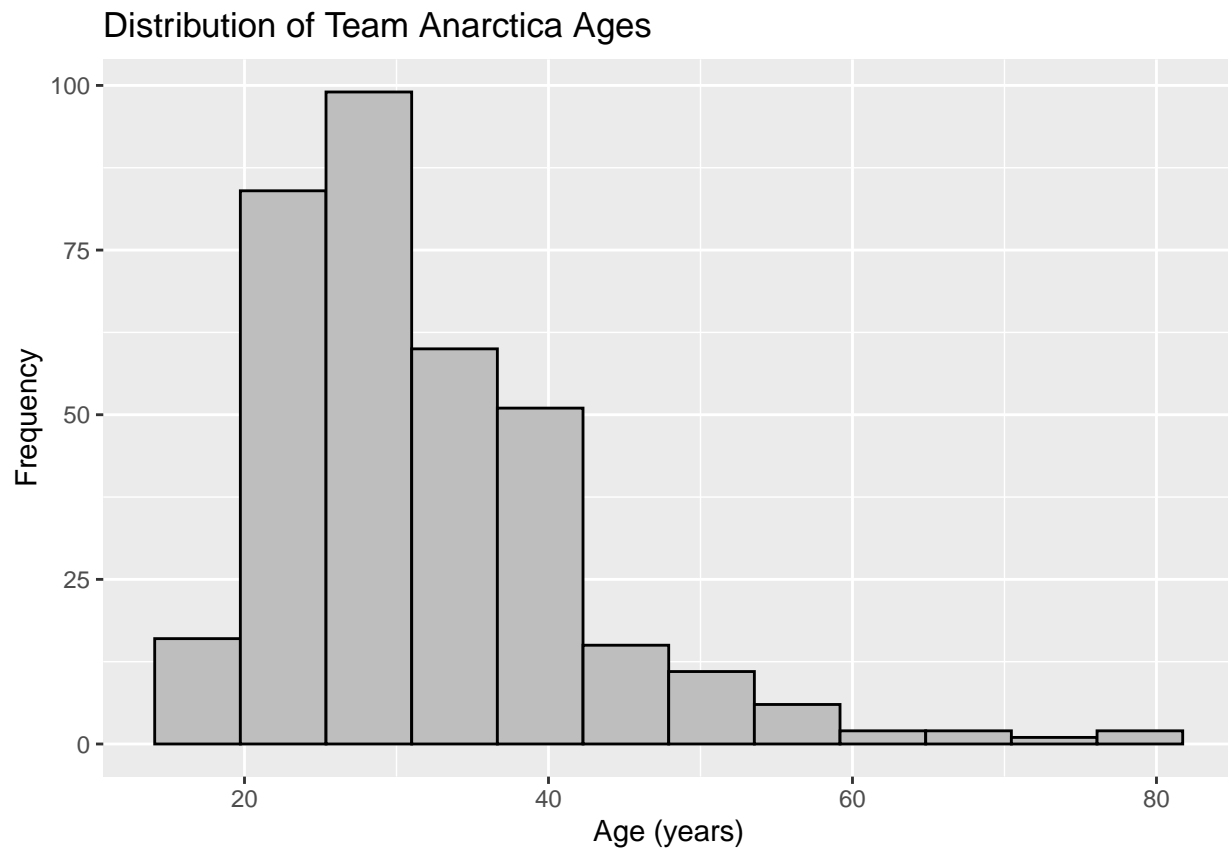
Create a histogram showing the distribution of height from the sick data set. Use the techniques show above to style it, and provide custom labels:

```
ggplot(data=sick, mapping=aes(x=height_cm))+
  geom_histogram(bins=12, color="black", fill="grey")+
  labs(title="Distribution of Team Anarctica Height", x="Height (cm)", y="Frequency")
```

Distribution of Team Anarctica Height
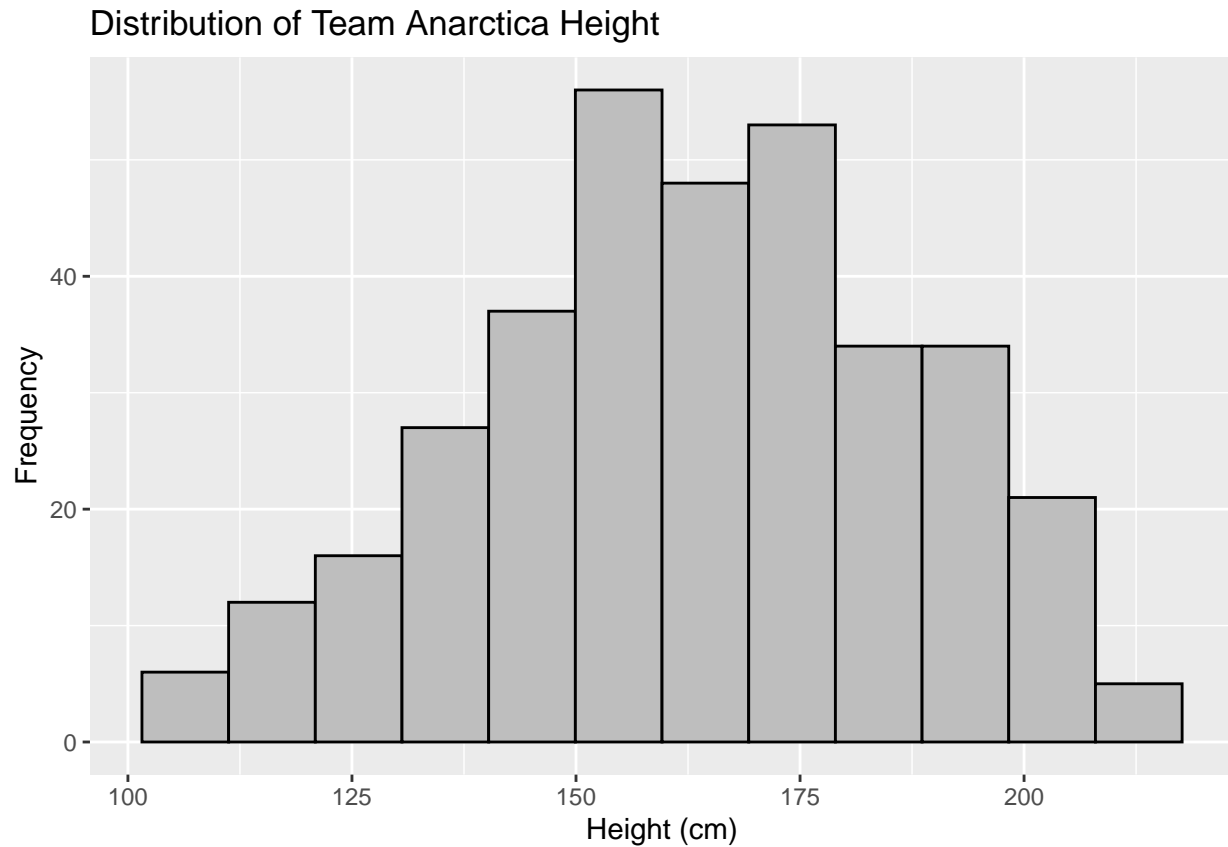
Scatter plots

Scatter plots are useful for visualizing if there is a potential association between two variables. Let's take a look at an example scatter plot, displaying age on the x-axis (independent variable), and weight on the y-axis (dependent variable). You'll see this is pretty similar to histogram code above, except for an additional attribute in the mapping aes function (y=), and a different geom function (geom_point):

```
ggplot(data=sick, mapping=aes(x=age, y=weight_kg))+
  geom_point()
```
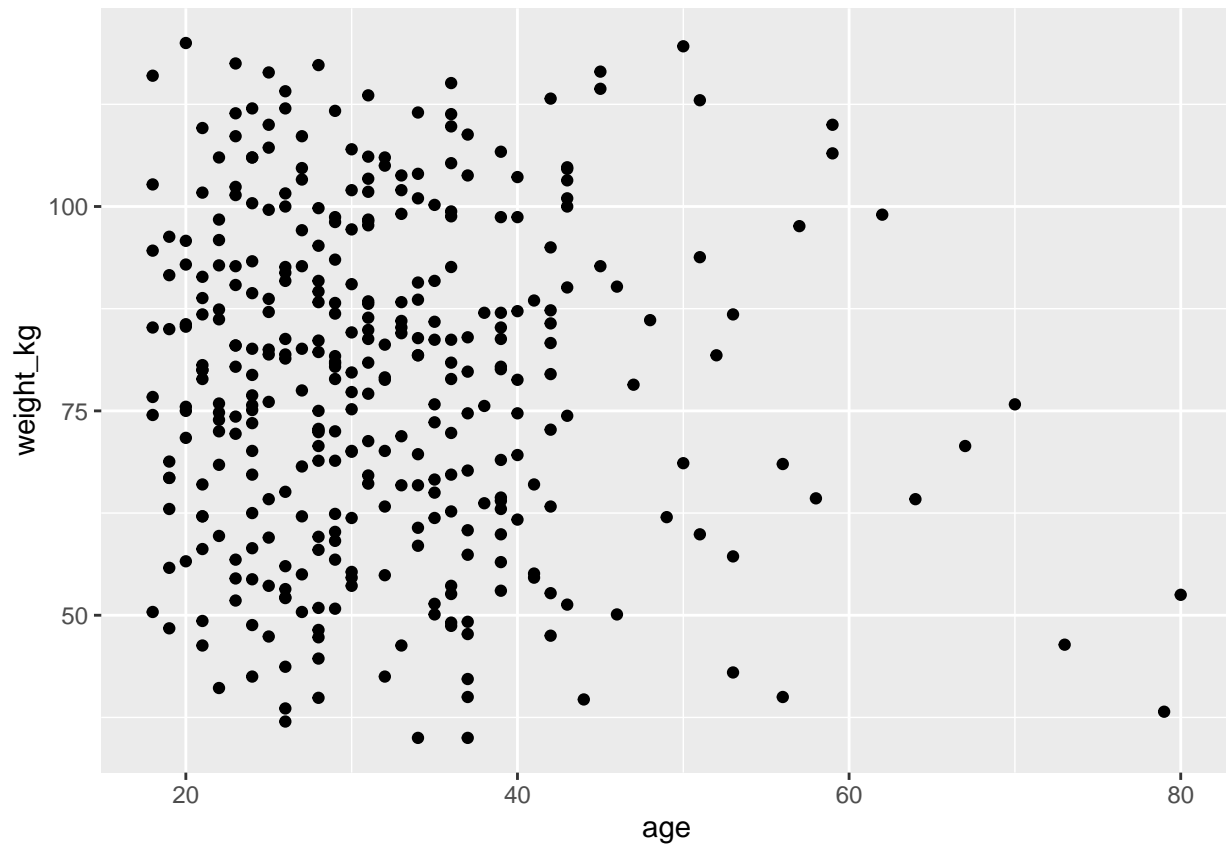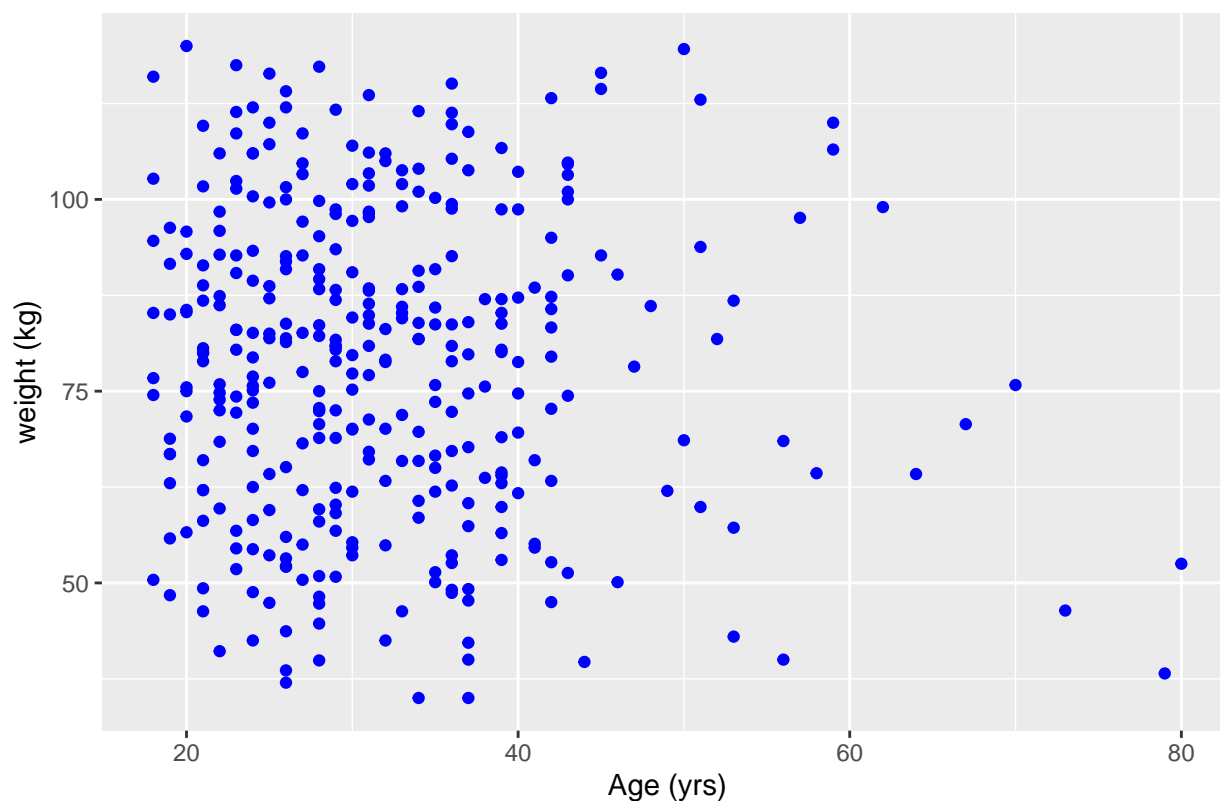
Try improving this plot by adding a title, x- and y-axis labels, and perhaps some color.

```
ggplot(data=sick, mapping=aes(x=age, y=weight_kg))+
  geom_point(color="blue")+
  labs(title="Weight x Age of Team members", x="Age (yrs)", y="weight (kg)")
```

Weight x Age of Team members

In your group or with your neighbor, create a scatter plot to show what variables might be associated to give some clues about the root of the sickness. Use proper labels and add some color. Try some different variables until you see what might be a pattern.

```
ggplot(data=sick, mapping=aes(x=perc_fish, y=doctor_trips))+
  geom_point(color="red")+
  labs(title="Association of % fish in diet with trips to the doctor", x="Percentage of fish in diet", y
```

Association of % fish in diet with trips to the doctor
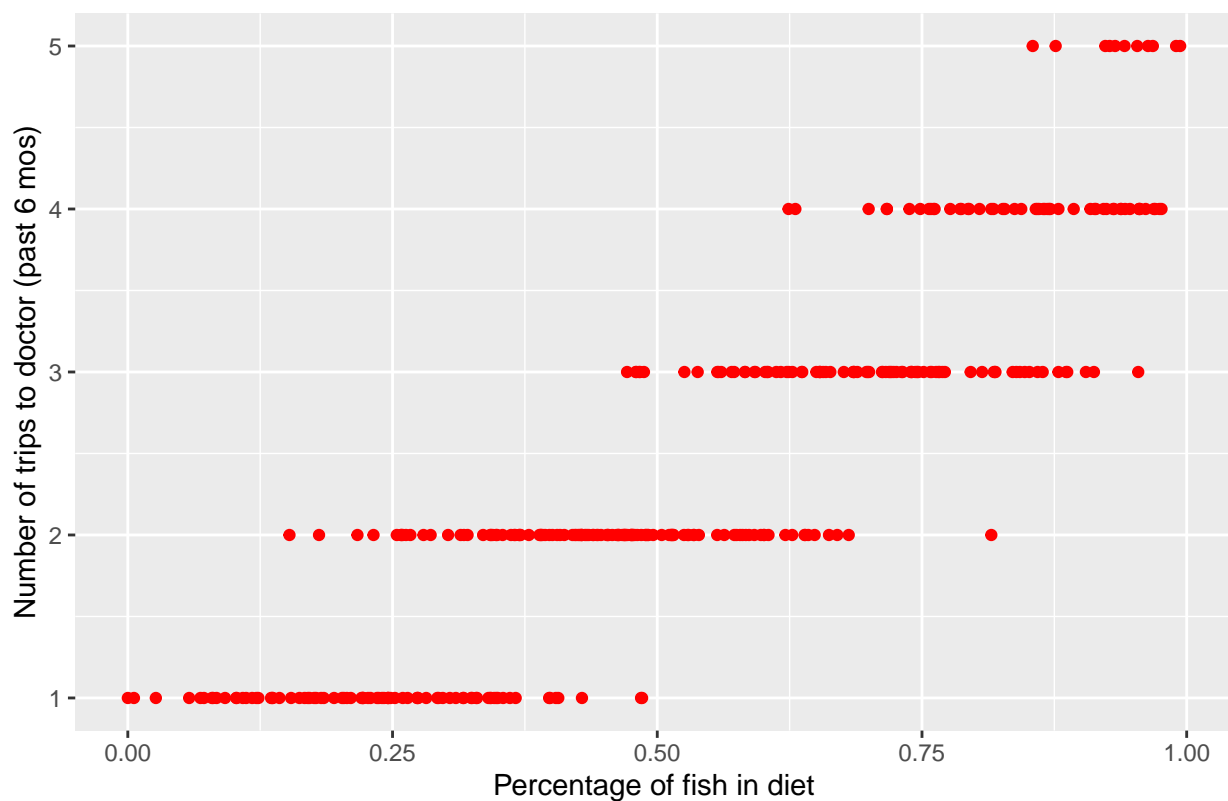
Bar Plots

Lastly we're going to look at bar plots. One key difference between bar plots and the previous two plots we examined is that bar plots often show summary statistics of categorical variables, instead of all data. As such, we'll need to use strategies from Module 1 to group data and generate statistics.

Let's say we're interested in creating a visualization showing the means on doctor visits in males and females. First we'll need to group the data and generate the means, using group_by/summarize:
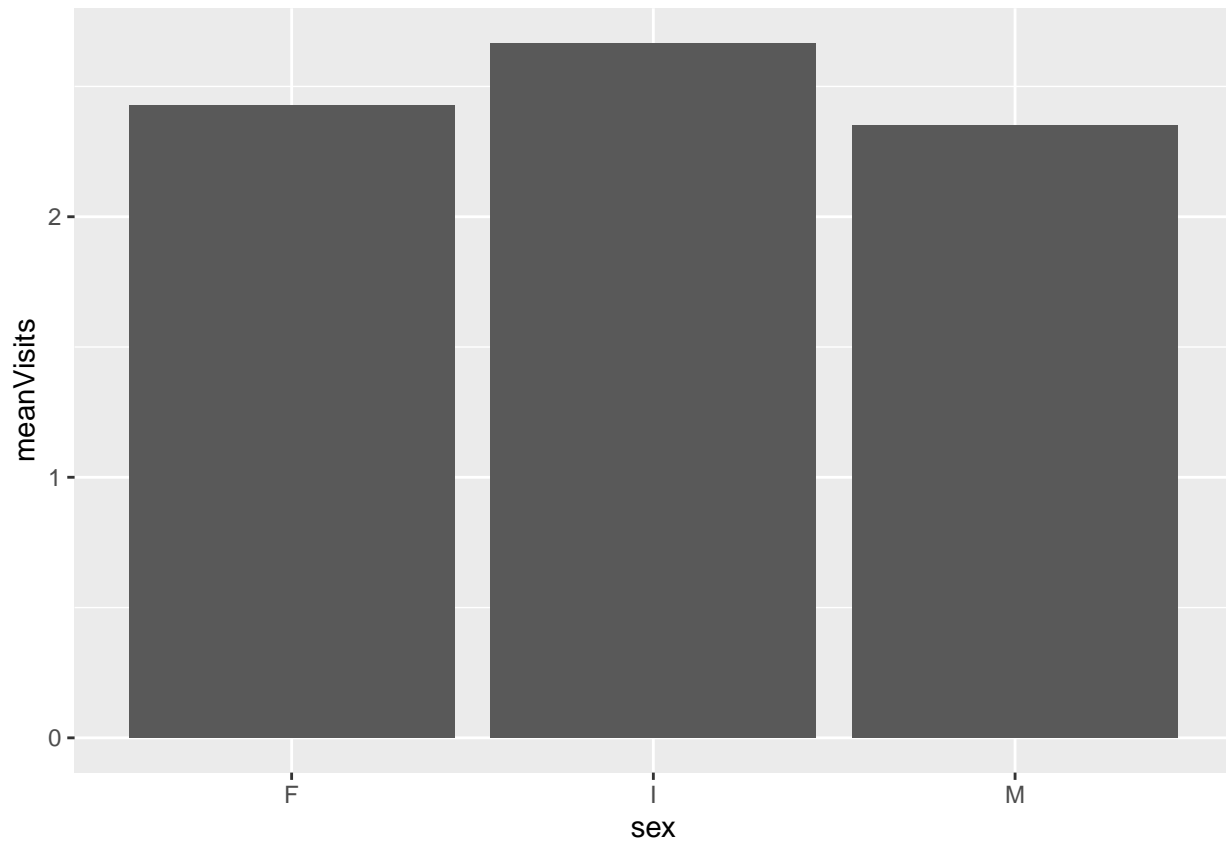
```
drVisits<-sick %>% group_by(sex) %>%
  summarize(meanVisits=mean(doctor_trips))

drVisits
```

```
## # A tibble: 3 x 2
##   sex   meanVisits
##   <chr>      <dbl>
## 1 F           2.43
## 2 I           2.67
## 3 M           2.35
```

Now that we have our data grouped with the average doctor visits, let's use it to create a bar chart:

```
ggplot(data=drVisits, mapping=aes(x=sex, y=meanVisits))+
  geom_bar(stat="identity")
```

We can improve this in a couple ways. Like before we can add better labeling with labs. But we can also add some color and a legend by using the "fill" attribute in the aes function, and set it to the sex variable:

```
ggplot(data=drVisits, mapping=aes(x=sex, y=meanVisits, fill=sex))+
  geom_bar(stat="identity")+
  labs(title="Average doctor visits by sex", x="Sex", y="Average dr visits, last 6 months")
```

Average doctor visits by sex
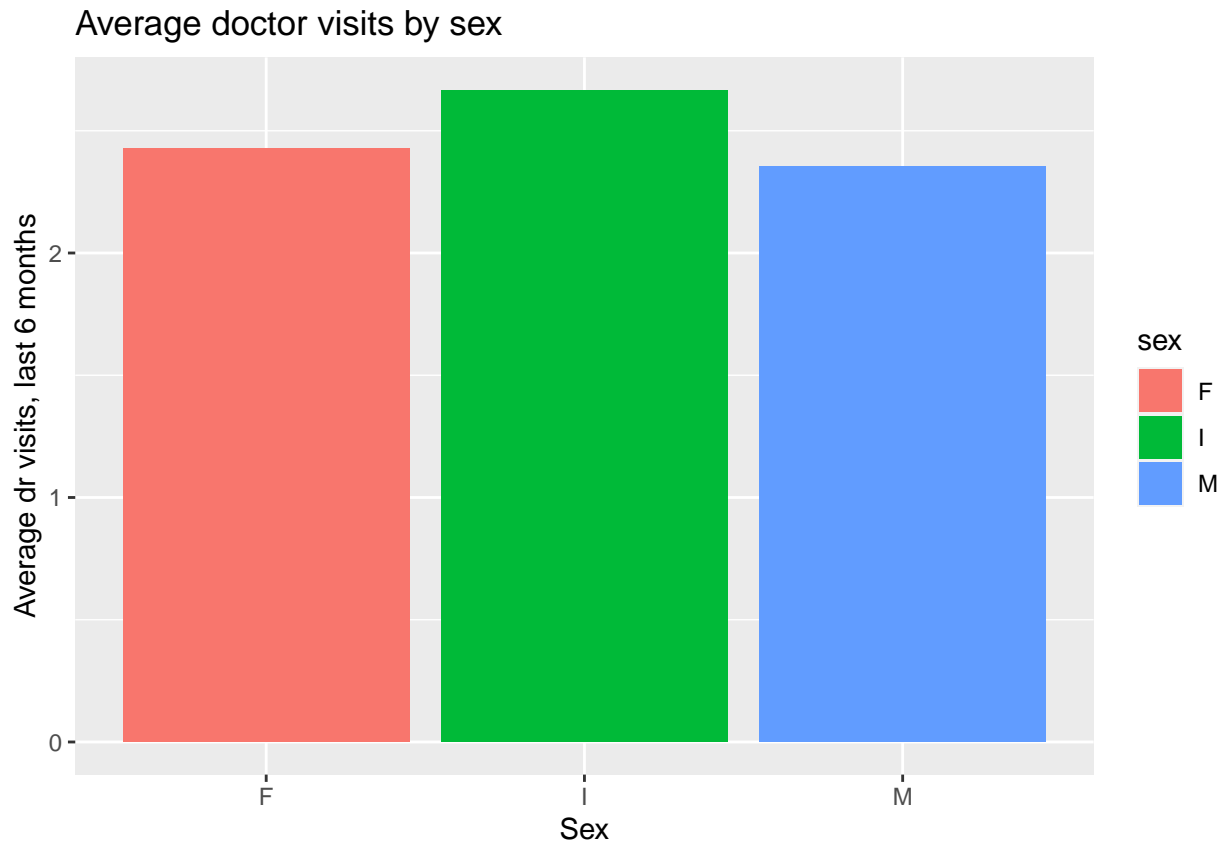
Let's say were interested in comparing average plant consumption among team members who specialize in Climatology, Data Science, and Psychology. Create a bar plot below showing the average plant consumption among the three groups (hint: you may need to filter your data first).

```
cliDsPsy<-sick %>%
  filter(specialties=="Climatology" | specialties=="Data Science" | specialties=="Psychology") %>%
  group_by(specialties) %>% summarize(avgPlant=mean(perc_plant))



ggplot(data=cliDsPsy, mapping=aes(x=specialties, y=avgPlant, fill=specialties))+
  geom_bar(stat="identity")+
  labs(title="Plant consumption among Aquaculture, Geology, and Psycology Specialists", x="Specialty", )
```

Plant consumption among Aquaculture, Geology, and Psycology Specialists
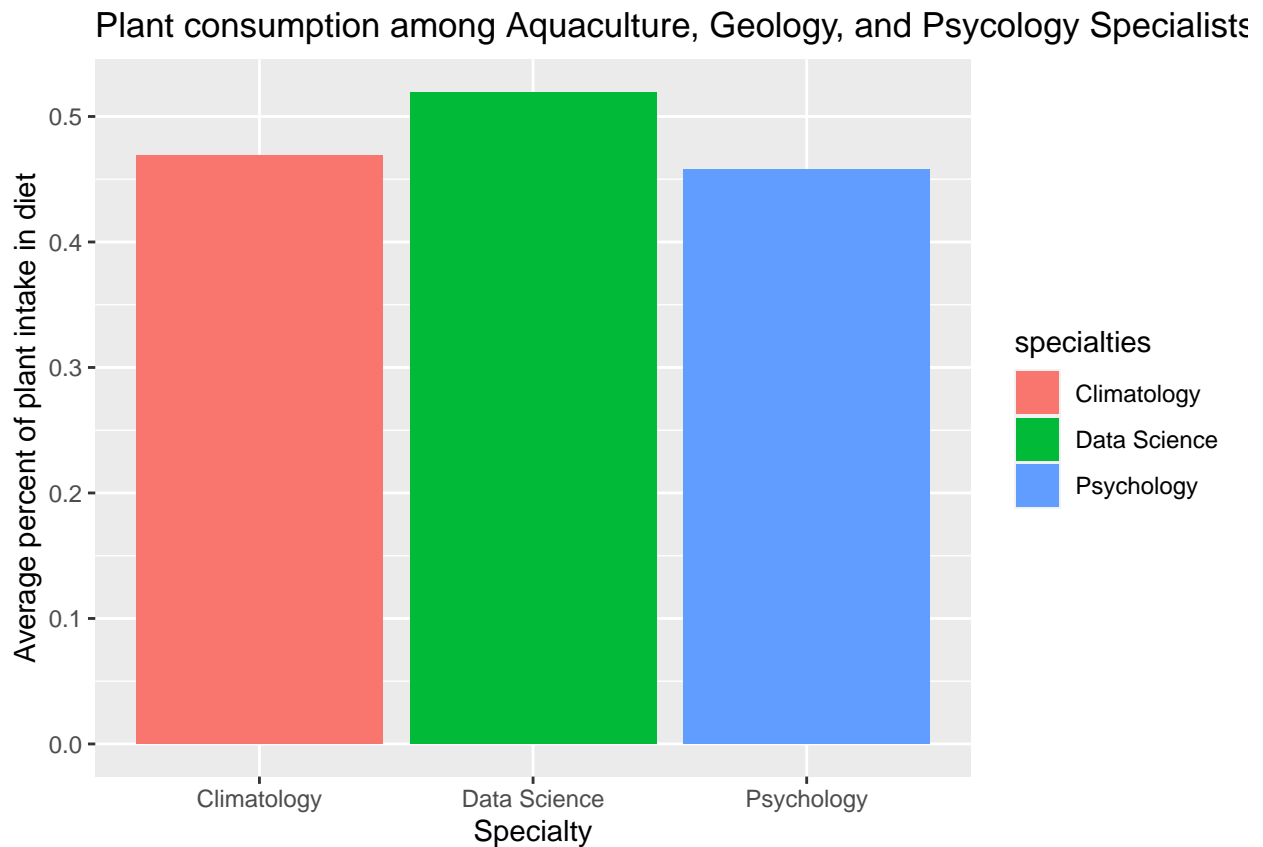
Saving plots

It's likely at some point you may want to save your plots to image files. There's a handy function to do this: **ggsave**. ggsave takes the argument of an image file name (e.g. myPlot.jpg), and optionally a variable, if you have your plot assigned to a variable - if not, it will save the last plot rendered.

```
ageHist<-ggplot(data=sick, mapping=aes(x=age))+
  geom_histogram(bins=12,color="black", fill="grey")+
  labs(title="Age distributiuon of team members", x="Age(years)", y="Distribution")

ggsave("ageHistPlot.jpg", ageHist)
```

```
## Saving 6.5 x 4.5 in image
```