Module 1 Lab DSE 5001 Homework

HDS

2024-08-06

Module 1 Lab DSE 5001 Problems

HD Sheets

August 6, 2024 Checked 1/3/2025

Student

-Ryan Waterman 1/14/2025

-if you look things up, paste in the link as you go

Related Reading

https://r4ds.hadley.nz/data-visualize (https://r4ds.hadley.nz/data-visualize)

https://r4ds.hadley.nz/workflow-basics (https://r4ds.hadley.nz/workflow-basics)

Loading Libraries

It is good practice to load the libraries in use at the start of a document, although in practice we often load libraries later as we need them.

We install documents on a machine once. Installing means downloading from a repository and storing them on your hard drive.

To install ggplot, we type this into the console

install.packages(“ggplot2”)

to install tidyr, it is

install.packages(“tidyr”)

TO use the libraries we have to install them into the working environmen in R

“install” means to download a copy of the package from a storage repository and store it on the harddrive of your computer. You only have to do this once

“load” means to move the package into the memory “workspace” of R so you can use it in that work session. You need to load a package into the workspace each time you want to use it.

```
library("ggplot2")
library("tidyr")
```

Using built-in data

Most R packages come with example data that you can use to learn how to use the package.

Some of this data is what I call “lab-rats” which are the common example sets used over and over in data science, Fisher’s Iris data, the mtcars data set, the NIST digits set, etc.

The command below will show us the available data sets

```
data()
```

The chickwts data set

We will use the chickwts data set

```
data(chickwts)
```

Question 1

What are we dealing with?

In the pair programming exercises and in chapters 1 and 2 of the Wickham text, there are various ways to help us understand the nature of the data we have. Go back and look a these if you don’t remember them.

```
Show the code you used to figure out the answers to each of the     questions below.  In some c
ases, you don't need code just
```

What type of data storage structure is chickwts?

```
str(chickwts)
```

```
## 'data.frame':    71 obs. of  2 variables:
##  $ weight: num  179 160 136 227 217 168 108 124 143 140 ...
##  $ feed  : Factor w/ 6 levels "casein","horsebean",..: 2 2 2 2 2 2 2 2 2 2 ...
```

- The data storage structure is a data frame.

What are the names of the the columns or variables? How many columns are there?

- There are tow columns, named weight and feed, as seen from the structure function output, above.

How many rows (individual chickens) are there in this set?

- There are 71 observations, as seen from the structure function output, above.

What are the data types of the columns?

- The data types are number and factor.

What is meant by the term “factor” in R (you may need to google this).

- A factor is a data structure that is used for categorical data

Use the help function or google to look up the dataset

```
help(chickwts)
```

```
## starting httpd help server ... done
```

How many different types of feeds were used?

```
# your code here
categories <- unique(chickwts$feed)
length(categories)
```

```
## [1] 6
```

- I found this code snippet here (https://stackoverflow.com/questions/46017812/r-get-all-categories-in-column). I just needed some help with the syntax :).

Question 2

What is the main question you would ask about this data set? Are there alternative questions?

- The main question would be: Which supplement is the most effective for weight gain. Alternative questions could be: Which supplement was the least effective or which supplement had the least variance in final weight.

How would you decide on an answer?

- The naive approach would be to find the corresponding feed for the maximum weight. See the code block, below:

```
#First show the summary to get an idea of the data
summary(chickwts)
```

```
##      weight             feed
##  Min.   :108.0   casein   :12
##  1st Qu.:204.5   horsebean:10
##  Median :258.0   linseed  :12
##  Mean   :261.3   meatmeal :11
##  3rd Qu.:323.5   soybean  :14
##  Max.   :423.0   sunflower:12
```

```
#Find the index of the max weight
index <- match(max(chickwts['weight']),chickwts$weight)
index
```

```
## [1] 37
```

```
#Find the corresponding feed for the maximum weight
chickwts$feed[index]
```

```
## [1] sunflower
## Levels: casein horsebean linseed meatmeal soybean sunflower
```

NOTE: I used this (https://www.tutorialspoint.com/how-to-extract-a-particular-value-based-on-index-from-an-r-data-frame-column) resource to find the syntax for searching by index.

- A more thorough approach that eliminates false conclusions based on outliers would be to compare the averages of all of the categories. See the code block, below:

```
# Make an object to store the maximum mean weight across the categories
max_mean <- 0

for (cat in categories) {

  #Get a subset based on the category
  sub <- subset(chickwts, feed==cat)
  sub

  #find the mean of the subset's weight
  cat_mean <- mean(sub$weight)
  cat_mean

  #If the mean is greater than the current max mean, overwrite the max mean
  if (cat_mean > max_mean) {
    max_mean <- cat_mean
    #Also store the category of the max mean
    max_cat <- cat
  }
}

max_mean
```

```
## [1] 328.9167
```

```
max_cat
```

```
## [1] "sunflower"
```

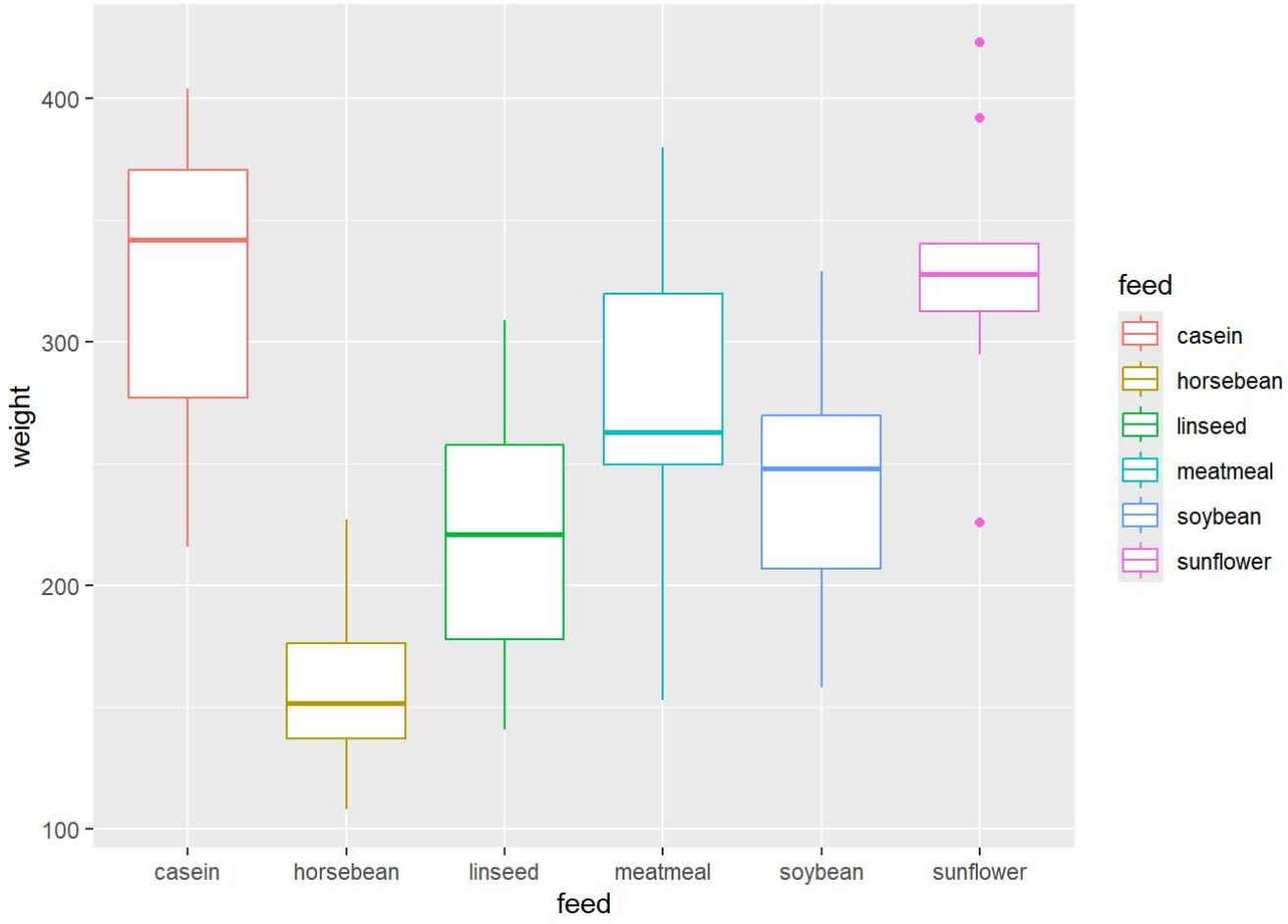
NOTE: I used this (https://statisticsglobe.com/loops-in-r/) resource to learn the syntax for looping, this (https://www.geeksforgeeks.org/r-if-statement/) resource for if statements, and this (https://stackoverflow.com/questions/5052621/how-do-i-filter-a-data-frame-in-r-by-categorical-variable) source to learn how to find subsets.

Creating a box plot

Use ggplot to create a box plot showing the chickwt as y, using “feed” as the color

See the pair programming code for an example of how to do this

```
ggplot(
  data = chickwts,
  mapping = aes(x = feed, y = weight, color = feed)
  ) + geom_boxplot()
```



Question 3

Which feed produces

1.) the largest chicks 2.) the smallest chicks 3.) the greatest range of chick weight

Note: These questions are deliberately (and annoyingly) vague.

Explain your reasoning, and what feature of the box plots you used

You may need to look up boxplots to do this. > I used the section 1.5.1 of the book (https://r4ds.hadley.nz/data-visualize) to do aid in this question.

Explain which feed you would use and why.

> Looking back on my answer to question 2… I feel a bit silly now. Alas, I have lived and learned.

1. Casein reliably produces the largest chicks, as the IQR makes up the highest set of weight values with a comparable distribution to the rest of the feed types.
2. Horsebean reliably produces the smallest chicks, as the IQR makes up the lowest set of weight values with a tight distribution compared to the rest of the feed types.
3. The greatest range of chick weight is from the sunflower feed, as it has the largest distribution.
4. Although sunflower produced the largest chicks and had otherwise similar performance to casein, the reliability of the result was far lower than casein. This is due to the number of outliers in the sunflower data set, and as a result, would lead me to choose casein.

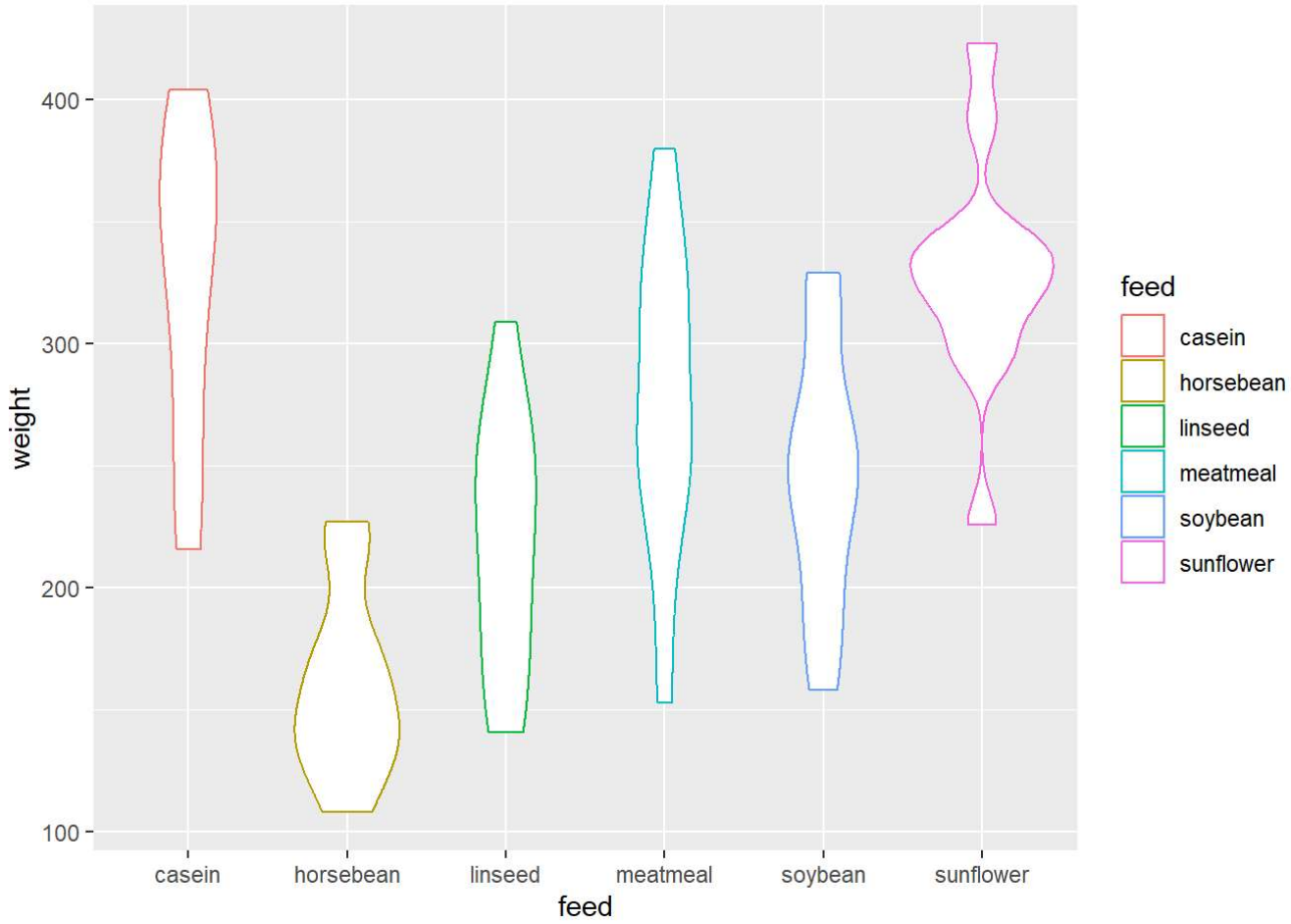
Question 4

Look up ggplots ‘violin plot’, a google search of “ggplot violin plot” will bring up examples and instructions of how to create it.

Make a violin plot of the chickwt data (ie y=weight), using ggplot and color coding by feed

Explain what the plots mean. What do they tell you about the optimum feed?

```
ggplot(
  data = chickwts,
  mapping = aes(x = feed, y = weight, color = feed)
  ) + geom_violin()
```



- The violin plot shows the density of the distribution over the range of the dataset. Casein, for example, has a range from ~215 grams to ~405 grams, with a relatively even density except for the top of the range. The “candlepin” shape of the casein plot shows a higher density of measurements at the top of the range, which is an indication that a larger dataset would converge to about 375 grams on average.

Exercises from Wickham’s book (No.s 5 and 6)

Exercise 5

Make a scatterplot of bill_depth_mm vs. bill_length_mm and color the points by species. What does adding coloring by species reveal about the relationship between these two variables? What about faceting by species?

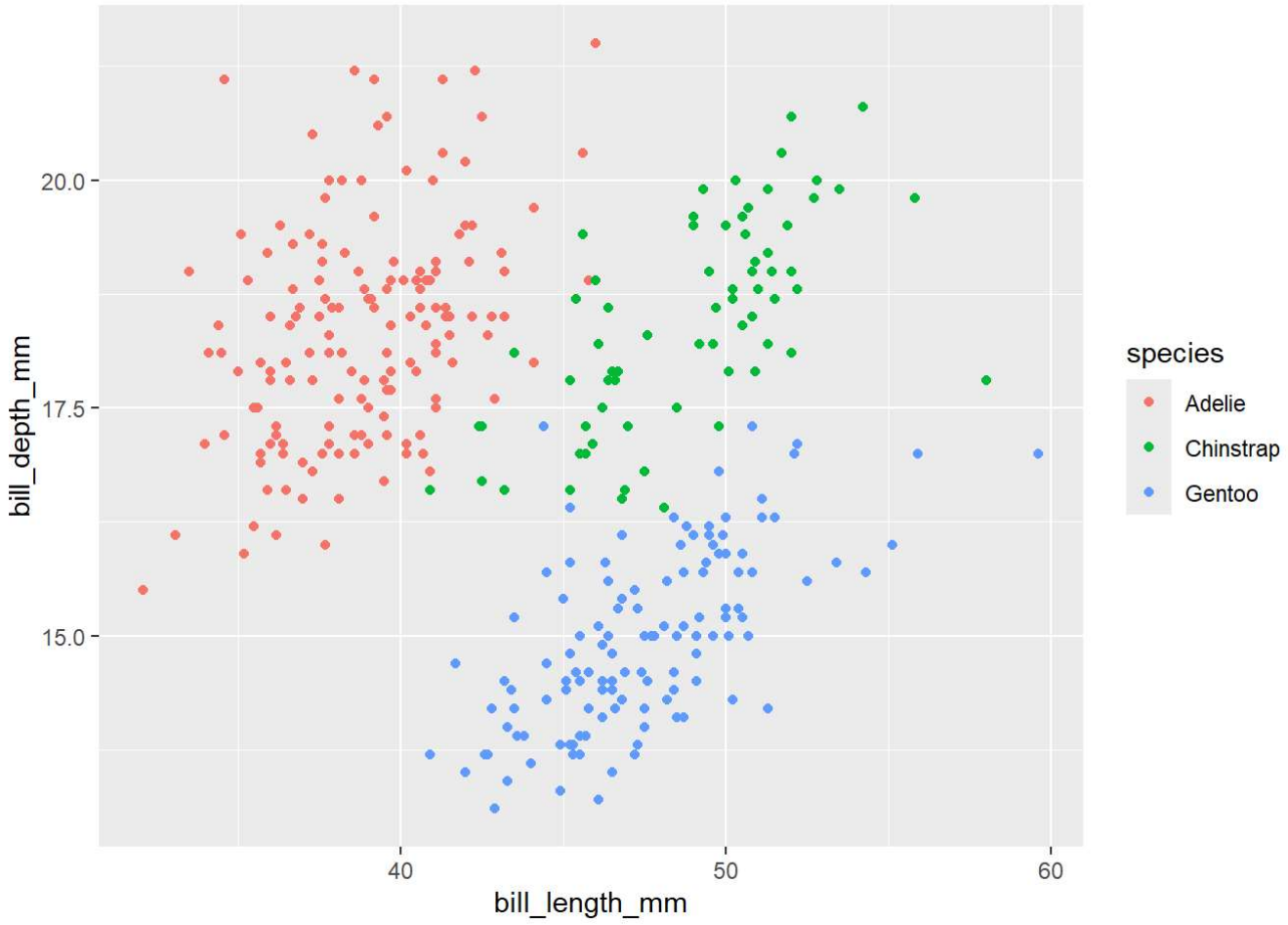
```
#Load the data for exercise 5:
library(palmerpenguins)
```

```
#Take a look at the data
summary(penguins)
```

```
##       species          island    bill_length_mm  bill_depth_mm
##  Adelie   :152   Biscoe   :168   Min.   :32.10   Min.   :13.10
##  Chinstrap: 68   Dream    :124   1st Qu.:39.23   1st Qu.:15.60
##  Gentoo   :124   Torgersen: 52   Median :44.45   Median :17.30
##                                  Mean   :43.92   Mean   :17.15
##                                  3rd Qu.:48.50   3rd Qu.:18.70
##                                  Max.   :59.60   Max.   :21.50
##                                  NA's   :2       NA's   :2
##  flipper_length_mm  body_mass_g       sex           year
##  Min.   :172.0     Min.   :2700   female:165   Min.   :2007
##  1st Qu.:190.0     1st Qu.:3550   male  :168   1st Qu.:2007
##  Median :197.0     Median :4050   NA's  : 11   Median :2008
##  Mean   :200.9     Mean   :4202                Mean   :2008
##  3rd Qu.:213.0     3rd Qu.:4750                3rd Qu.:2009
##  Max.   :231.0     Max.   :6300                Max.   :2009
##  NA's   :2         NA's   :2
```

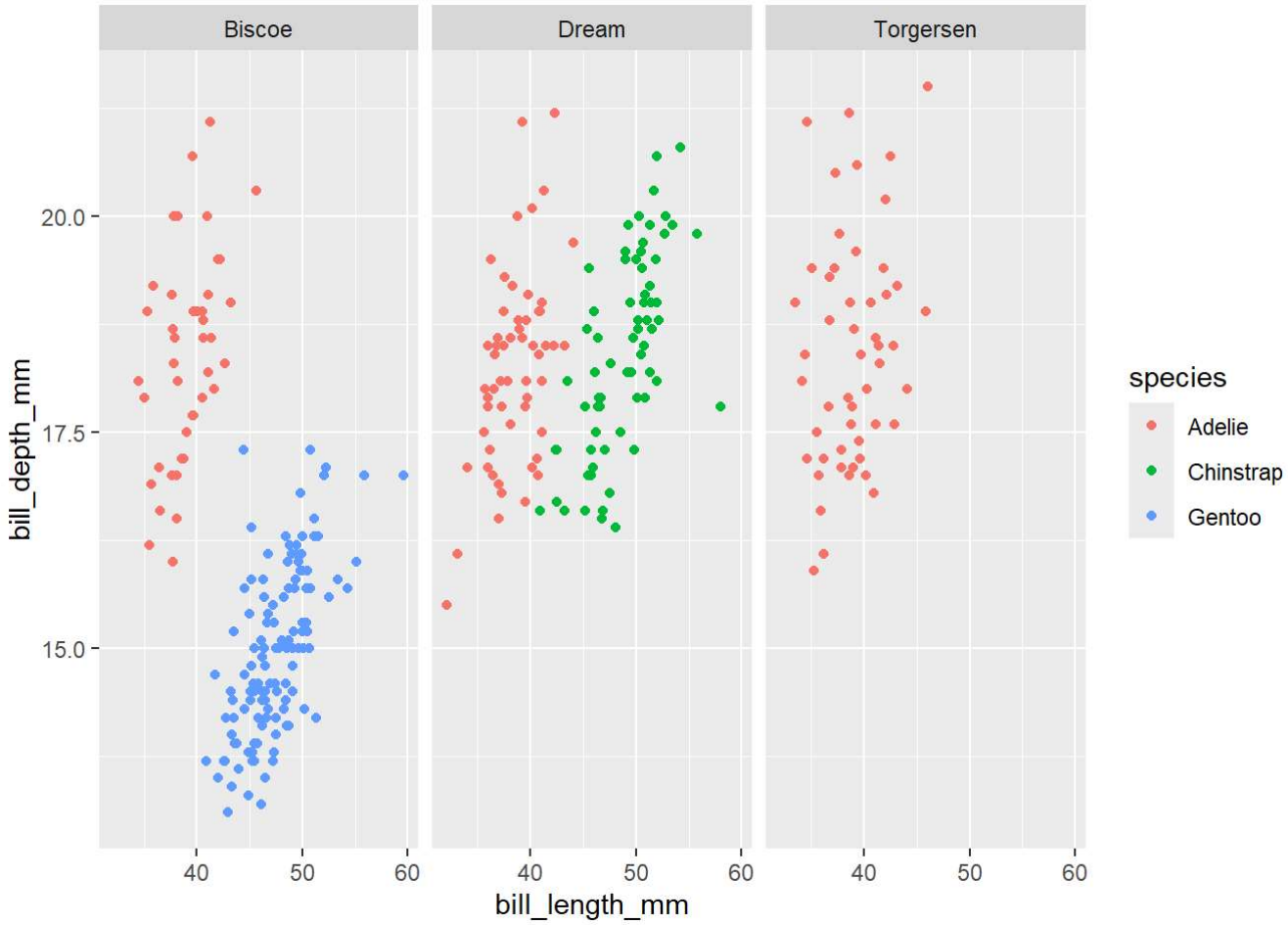
```
#Scatter plot of bill length vs bill depth, color coded by species
ggplot(
  data = penguins,
  mapping = aes(
  x = bill_length_mm, y = bill_depth_mm,
  color = species)
) + geom_point()
```

```
## Warning: Removed 2 rows containing missing values or values outside the scale range
## (`geom_point()`).
```




```
#Scatter plot of bill length vs bill depth, color coded by species and faceted by island
ggplot(
  data = penguins,
  mapping = aes(
  x = bill_length_mm, y = bill_depth_mm,
  color = species)
) + geom_point() + facet_wrap(~island)
```

```
## Warning: Removed 2 rows containing missing values or values outside the scale range
## (`geom_point()`).
```

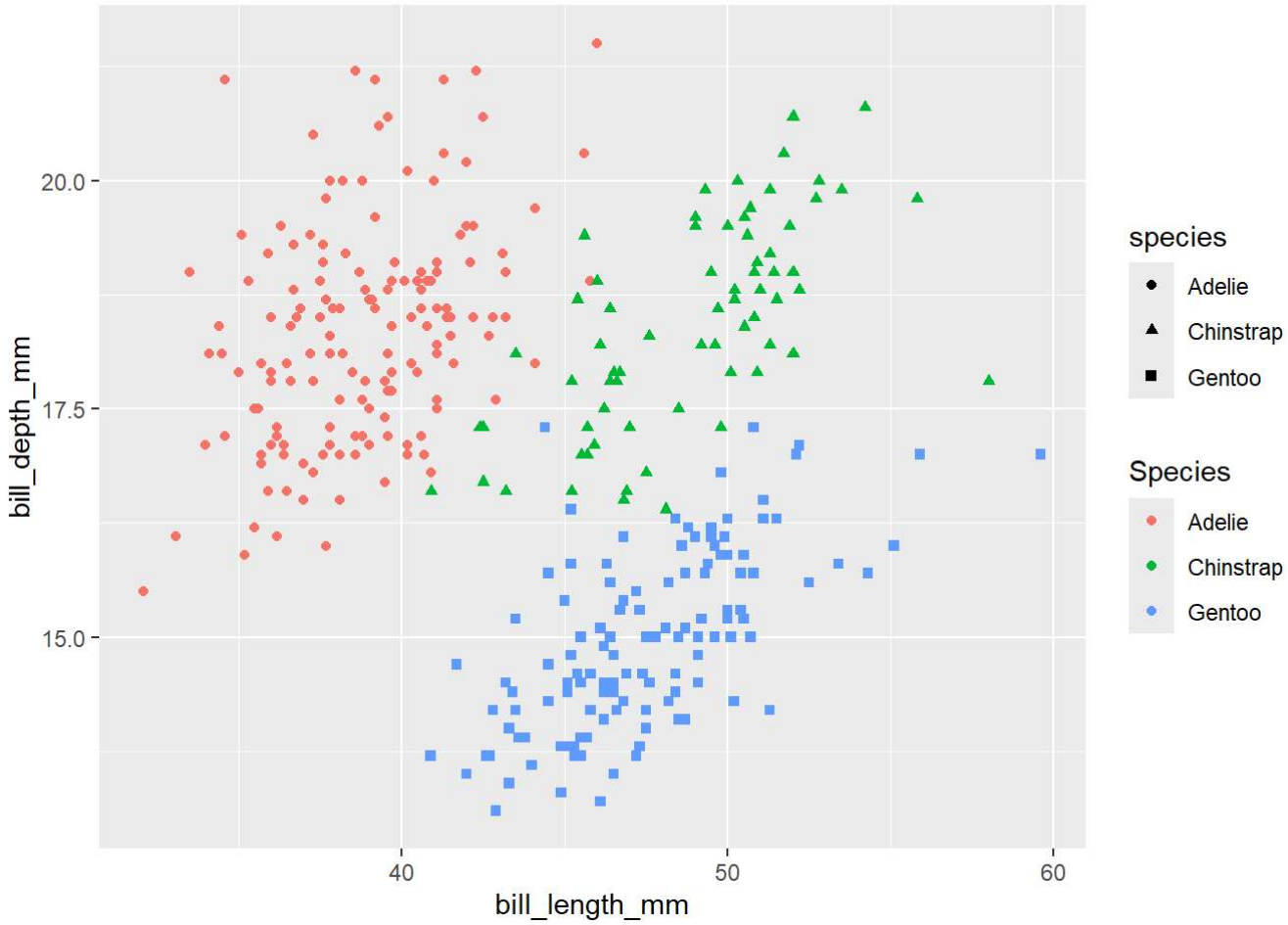


Exercise 6

Why does the following yield two separate legends? How would you fix it to combine the two legends?

```
ggplot(
  data = penguins,
  mapping = aes(
    x = bill_length_mm, y = bill_depth_mm,
    color = species, shape = species
  )
) +
  geom_point() +
  labs(color = "Species")
```

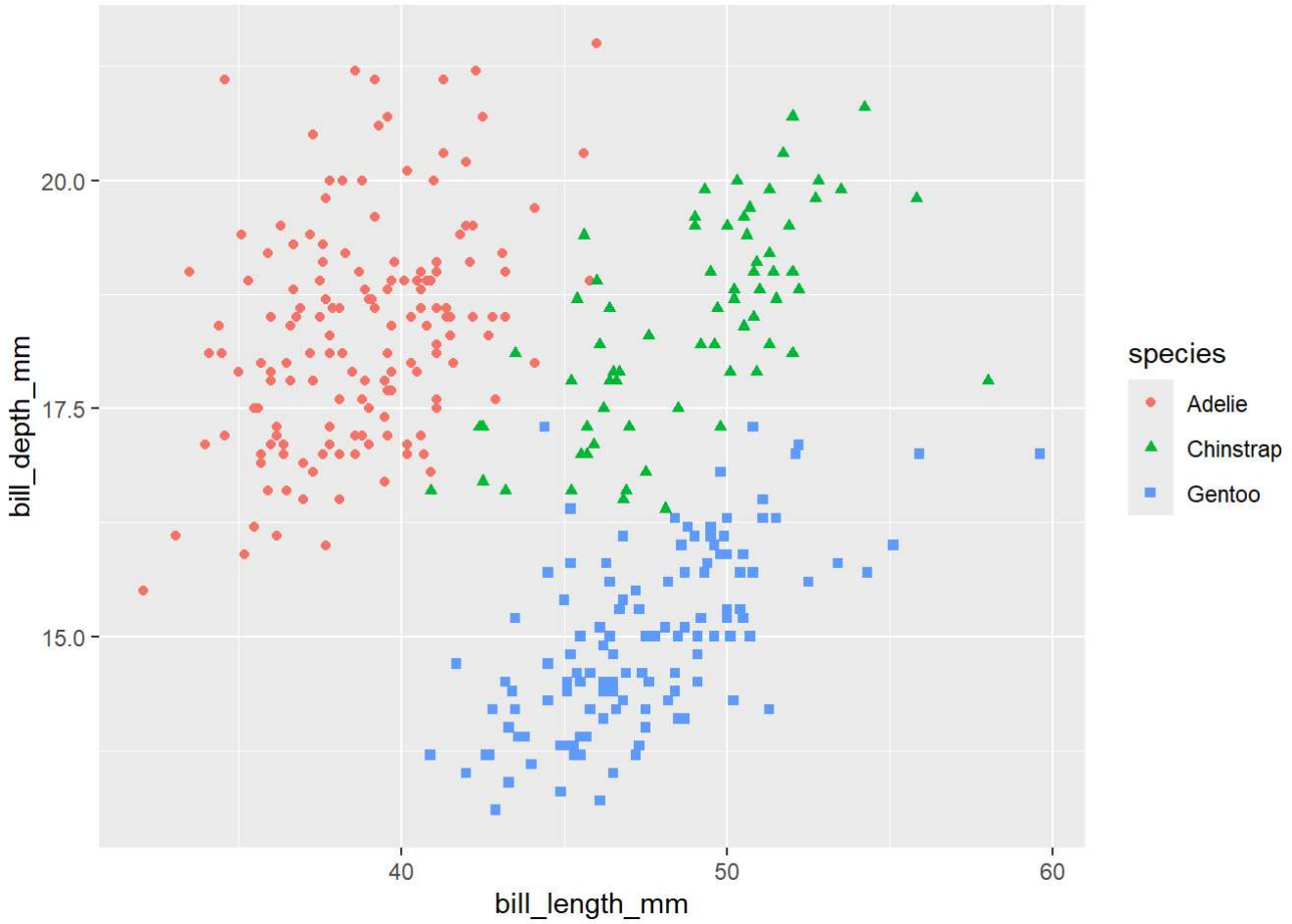
```
## Warning: Removed 2 rows containing missing values or values outside the scale range
## (`geom_point()`).
```



- This creates two legends because the labs function is not contained within the mapping constructor. See the fix, below:

```
ggplot(
  data = penguins,
  mapping = aes(
    x = bill_length_mm, y = bill_depth_mm,
    color = species, shape = species, labs(color = "Species")
  )
) +
  geom_point()
```

```
## Warning: Removed 2 rows containing missing values or values outside the scale range
## (`geom_point()`).
```



Question 5

Why does this code not work? - The code does not work because of a typo on “i” in “variable.”

```
my_variable <- 10
my_variable
```

```
## [1] 10
```

Look carefully! (This may seem like an exercise in pointlessness, but training your brain to notice even the tiniest difference will pay off when programming.)

Fix the problem

Question 6

Tweak each of the following R commands so that they run correctly:

```
library(tidyverse)
```

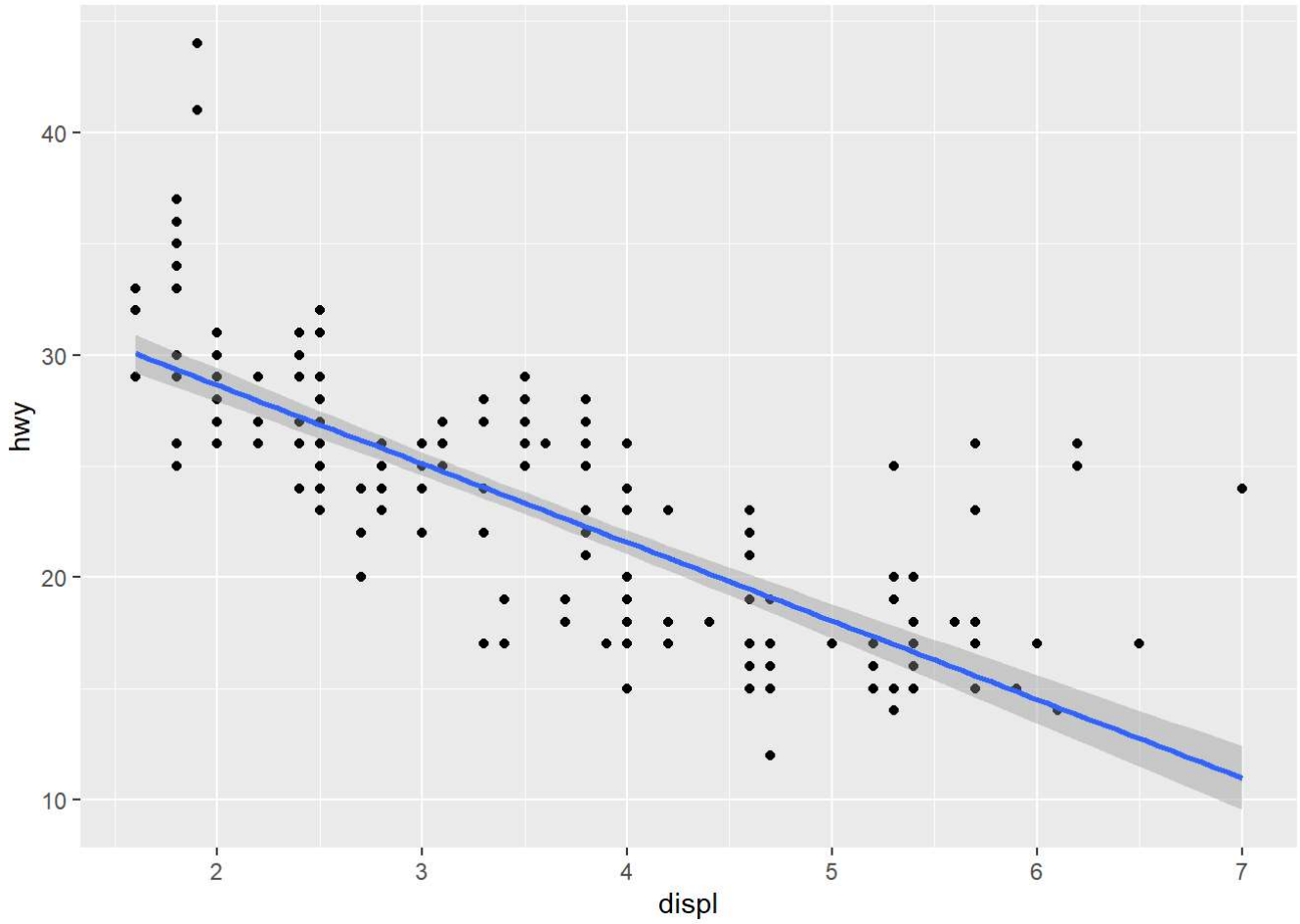
```
## ── Attaching core tidyverse packages ──────────────────────── tidyverse 2.0.0 ──
## ✓ dplyr     1.1.4     ✓ readr     2.1.5
## ✓ forcats   1.0.0     ✓ stringr   1.5.1
## ✓ lubridate 1.9.4     ✓ tibble    3.2.1
## ✓ purrr     1.0.2
## ── Conflicts ────────────────────────────────────────── tidyverse_conflicts() ──
## ✖ dplyr::filter() masks stats::filter()
## ✖ dplyr::lag()    masks stats::lag()
## ℹ Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to becom
e errors
```

```
str(mpg)
```

```
## tibble [234 × 11] (S3: tbl_df/tbl/data.frame)
##  $ manufacturer: chr [1:234] "audi" "audi" "audi" "audi" ...
##  $ model       : chr [1:234] "a4" "a4" "a4" "a4" ...
##  $ displ       : num [1:234] 1.8 1.8 2 2 2.8 2.8 3.1 1.8 1.8 2 ...
##  $ year        : int [1:234] 1999 1999 2008 2008 1999 1999 2008 1999 1999 2008 ...
##  $ cyl         : int [1:234] 4 4 4 4 6 6 6 4 4 4 ...
##  $ trans       : chr [1:234] "auto(l5)" "manual(m5)" "manual(m6)" "auto(av)" ...
##  $ drv         : chr [1:234] "f" "f" "f" "f" ...
##  $ cty         : int [1:234] 18 21 20 21 16 18 18 18 16 20 ...
##  $ hwy         : int [1:234] 29 29 31 30 26 26 27 26 25 28 ...
##  $ fl          : chr [1:234] "p" "p" "p" "p" ...
##  $ class       : chr [1:234] "compact" "compact" "compact" "compact" ...
```

```
ggplot(
  data = mpg,
  mapping = aes(x = displ, y = hwy)
) +
  geom_point() +
  geom_smooth(method = "lm")
```

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



Question 7: The Midwest data set

Okay, this is a bigger data set on US midwestern state demographics

The variable percbelowpoverty is the percentage of the population living below the poverty line.

Look at the data set (or look it up) and look at the other variables included.

State a hypothesis about how percbelowpoverty (the y axis, or target, or dependent variable) depends on another measured variable (which will be the x-axis)

Use ggplot to create a scatter plot using the ideas and methods in chapter 1 of Wickham that will test your hypothesis

Determine whether or not the data supported your hypothesis and how this is shown in the plot

State clearly what your results tell you.

```
head(midwest)
```

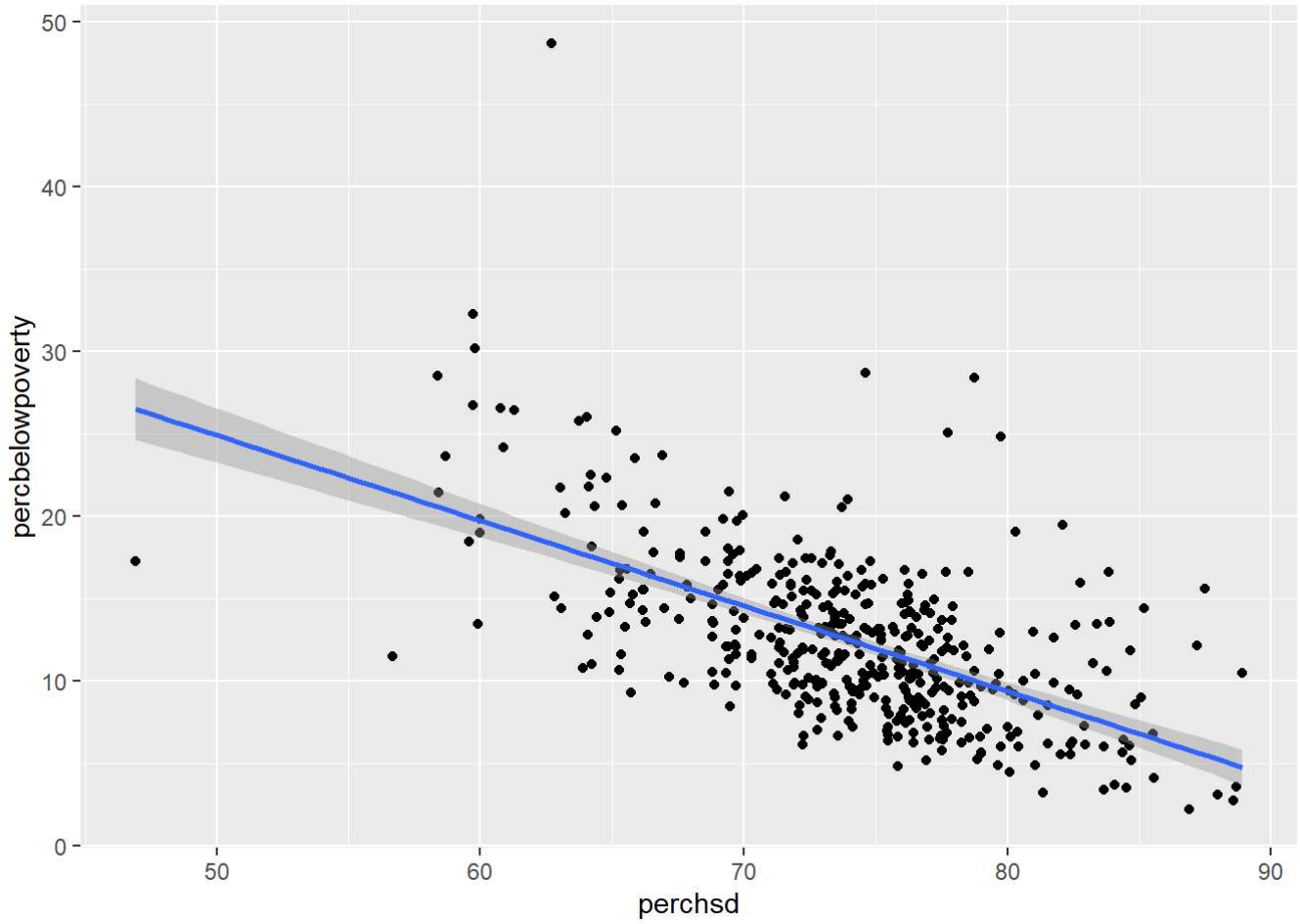
```
## # A tibble: 6 × 28
##     PID county   state  area poptotal popdensity popwhite popblack popamerindian
##   <int> <chr>    <chr> <dbl>    <int>      <dbl>    <int>    <int>         <int>
## 1   561 ADAMS    IL    0.052    66090      1271.    63917     1702            98
## 2   562 ALEXAND… IL    0.014    10626       759      7054     3496            19
## 3   563 BOND     IL    0.022    14991       681.    14477      429            35
## 4   564 BOONE    IL    0.017    30806      1812.    29344      127            46
## 5   565 BROWN    IL    0.018     5836       324.     5264      547            14
## 6   566 BUREAU   IL    0.05     35688       714.    35157       50            65
## # ℹ 19 more variables: popasian <int>, popother <int>, percwhite <dbl>,
## #   percblack <dbl>, percamerindan <dbl>, percasian <dbl>, percother <dbl>,
## #   popadults <int>, perchsd <dbl>, percollege <dbl>, percprof <dbl>,
## #   poppovertyknown <int>, percpovertyknown <dbl>, percbelowpoverty <dbl>,
## #   percchildbelowpovert <dbl>, percadultpoverty <dbl>,
## #   percelderlypoverty <dbl>, inmetro <int>, category <chr>
```

Hypothesis

- I believe the percbelowpoverty variable is inversely proportional to perchsd, percollege, and percprof.

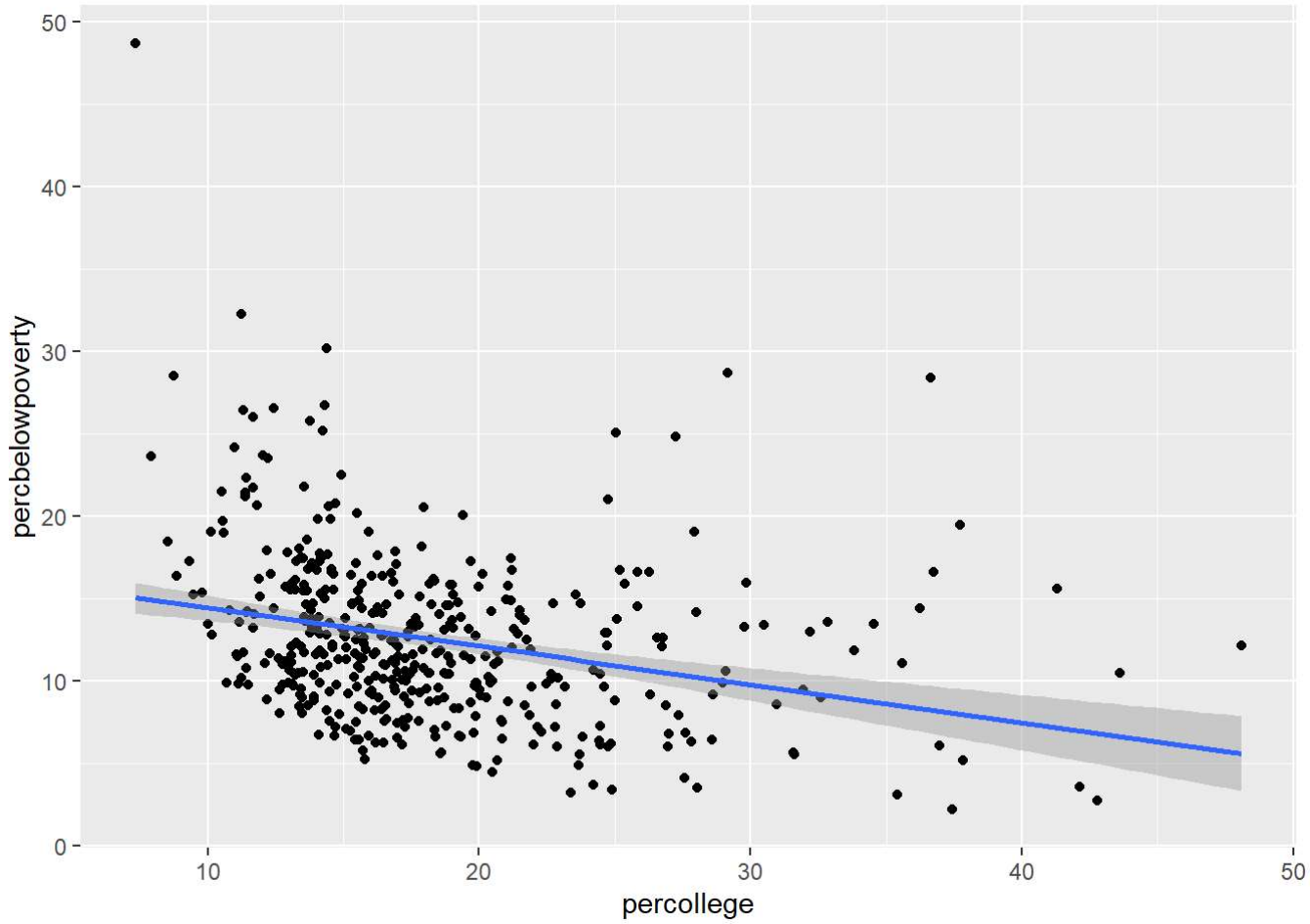
```
#Plot percbelowpoverty against perchsd
ggplot(
  data = midwest,
  mapping = aes(x = perchsd, y = percbelowpoverty)
) +
  geom_point() +
  geom_smooth(method = "lm")
```

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



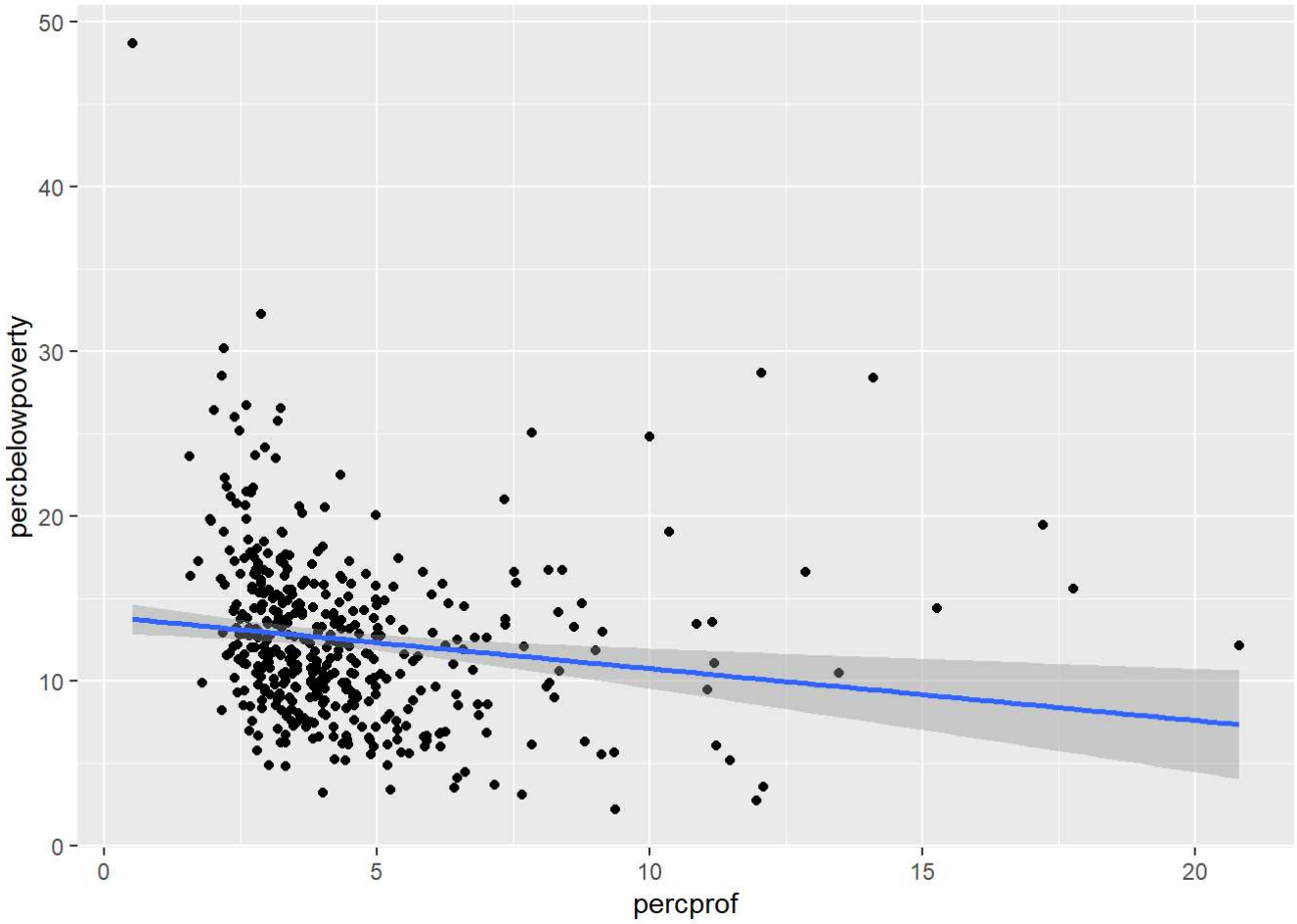

```
#Plot percbelowpoverty against percollege
ggplot(
  data = midwest,
  mapping = aes(x = percollege, y = percbelowpoverty)
) +
  geom_point() +
  geom_smooth(method = "lm")
```

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




```
#Plot percbelowpoverty against percprof
ggplot(
  data = midwest,
  mapping = aes(x = percprof, y = percbelowpoverty)
) +
  geom_point() +
  geom_smooth(method = "lm")
```

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


50
40
30
20
10
0
percbelowpoverty
0
5
10
15
20
percprof