Homework 4

Sarah Cannon

2024-09-24

Classmates/other resources consulted: N/A

To make the figures show up smaller in the knitted file:

```
# To make the figures show up smaller in the knitted file
# This just improves readability for your graders,
# so a single figure doesn't take up almost an entire page
knitr::opts_chunk$set(fig.width=4, fig.height=3)
```

> **Be sure to load the tidyverse and import the following data sets.**

```
library(tidyverse)

species <- read_csv("https://raw.githubusercontent.com/rfordatascience/tidytuesday/maste
r/data/2023/2023-05-02/species.csv") %>%
  mutate(commontype = ifelse(str_detect(commonname, "ouse"), "Mouse", "Rat")) %>%
  select(scientificname, commonname, commontype, granivore, meanhfl, meanwgt, juvwgt)
```

```
## Rows: 21 Columns: 15
## ── Column specification ────────────────────────────────────────────────────────
────────────────
## Delimiter: ","
## chr  (4): species, scientificname, taxa, commonname
## dbl (11): censustarget, unidentified, rodent, granivore, minhfl, meanhfl, maxhfl, min
wgt, meanwg...
##
## 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.
```

```
london_marathon <- read_csv("https://raw.githubusercontent.com/rfordatascience/tidytuesd
ay/master/data/2023/2023-04-25/london_marathon.csv") %>%
  select(Year, Applicants, Accepted, Starters, Finishers) %>%
  pivot_longer(Applicants:Finishers, names_to = "Category", values_to = "Count")
```

```
## Rows: 42 Columns: 8
## ── Column specification ────────────────────────────────────────────────────
────────────────
## Delimiter: ","
## chr  (1): Official charity
## dbl  (6): Year, Applicants, Accepted, Starters, Finishers, Raised
## date (1): Date
##
## 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.
```

```
artists <- read_csv("https://raw.githubusercontent.com/rfordatascience/tidytuesday/maste
r/data/2023/2023-01-17/artists.csv")
```

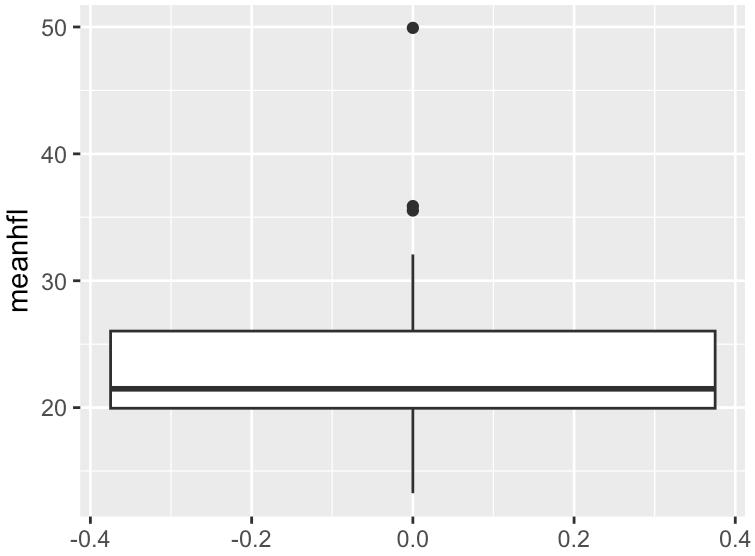
```
## Rows: 3162 Columns: 14
## ── Column specification ────────────────────────────────────────────────────
────────────────
## Delimiter: ","
## chr (8): artist_name, artist_nationality, artist_nationality_other, artist_gender, ar
tist_race, ...
## dbl (6): edition_number, year, space_ratio_per_page_total, artist_unique_id, moma_cou
nt_to_year,...
##
## 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.
```

Question 1 (8 points)

> **This question explore the species data set.**

> a. (2 points) **Make a boxplot of the mean hind foot length for the species in this data set.**

```
ggplot(data = species) +
  geom_boxplot(mapping = aes(y = meanhfl))
```



b. (2 points) **Looking at the boxplot you made in the previous part, what is the inter-quartile range (IQR) for the values in the meanhfl column?**

~26 - ~20 = ~6

Calculated:

```
species %>%
  summarize(IQR = IQR(meanhfl))
```

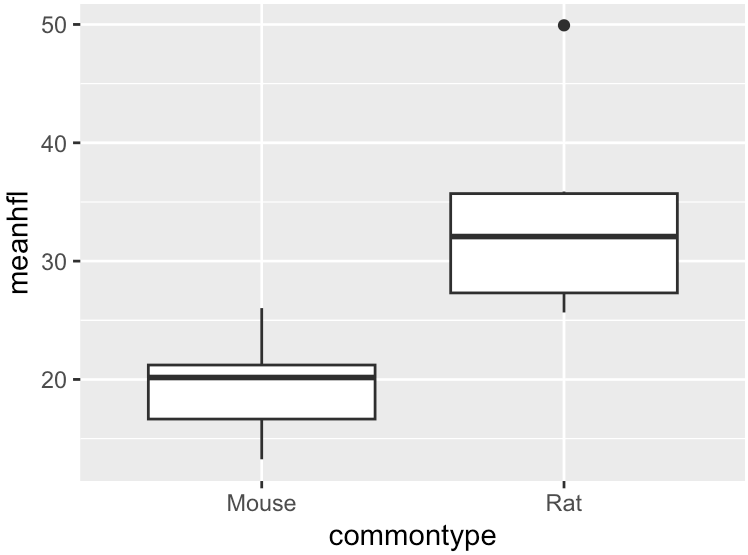
```
## # A tibble: 1 × 1
##     IQR
##   <dbl>
## 1  6.08
```

c. (2 points) **Explain, in your own words, why your boxplot from part (a) has three points on one side of the box but no points on the other side of the box.**

There are 3 points on the top of the boxplot because there are 3 outliers outside the whiskers which represent 1.5 * IQR.

d. (2 points) **Make a plot showing two boxplots, one for the mean hindfoot length when the commontype is Mouse and one for the mean hindfoot length when the commontype is Rat. They should be displayed on the same plot, not on separate plots or in separate facets.**

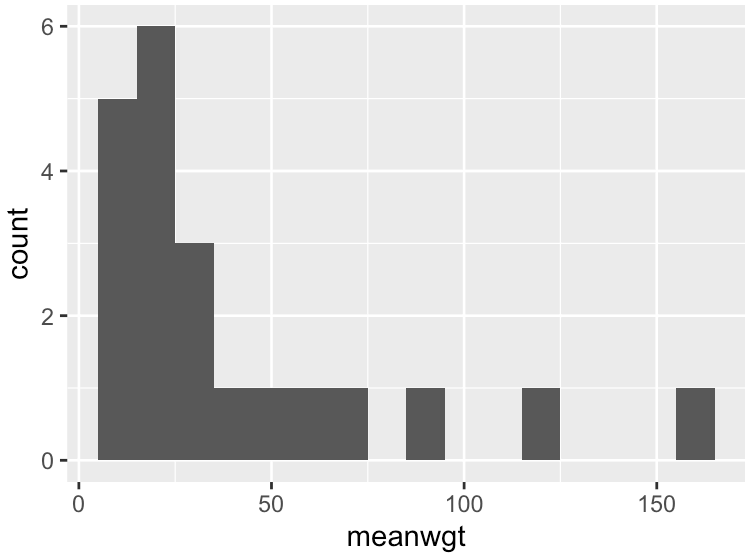
```
species %>%
  ggplot() +
  geom_boxplot(mapping = aes(x = commontype, y = meanhfl))
```



Question 2 (8 points)

a. (3 points) **Make a histogram of the mean weight of the species in the species data set. Pick an appropriate binwidth, and justify (in your own words) why you chose that binwidth.**

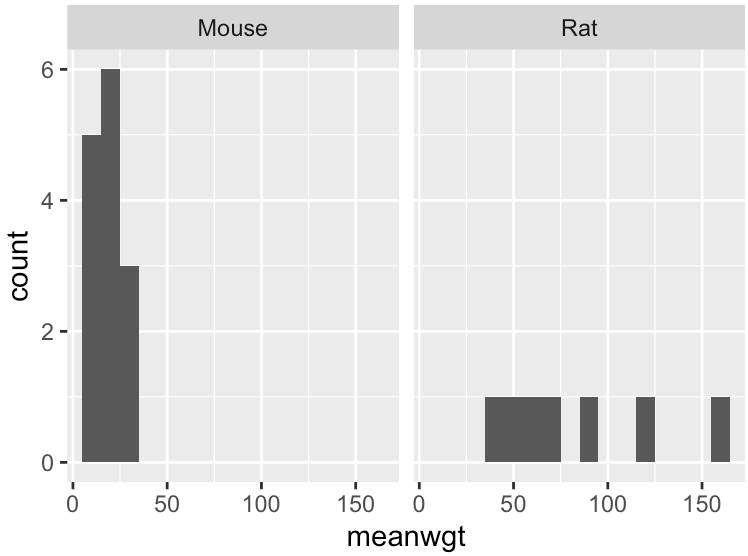
```
species %>%
  ggplot() +
  geom_histogram(mapping = aes(x = meanwgt), binwidth = 10)
```



A binwidth of 10 was best because it shows the story data well without being too cluttered and without interuptions in the histogram.

b. (2 points) **Modify your plot from the previous part to have two histograms, in two different facets, one showing the mean weight for species where the common type is Mouse and one showing the mean weight for species where the common type is Rat.**

```
species %>%
  ggplot() +
  geom_histogram(mapping = aes(x = meanwgt), binwidth = 10) +
  facet_wrap(~commontype)
```



c. (3 points) **Suppose you some data where every value in a particular column is a multiple of 100, such as the amount column in the following sales table:**

```
sales <- tibble(amount = c(100, 300, 200, 500, 900, 1000, 800, 800, 700, 1500, 300, 400,
600, 600, 500, 700, 800, 900, 1000, 1200, 1300, 300, 800, 1000, 1500, 1800, 1900, 2100,
2300, 2500, 2100, 1600, 1700, 1900, 3000, 2000, 2300, 2600, 1700, 1800))
sales
```

```
## # A tibble: 40 × 1
##    amount
##     <dbl>
##  1    100
##  2    300
##  3    200
##  4    500
##  5    900
##  6   1000
##  7    800
##  8    800
##  9    700
## 10   1500
## # ℹ 30 more rows
```

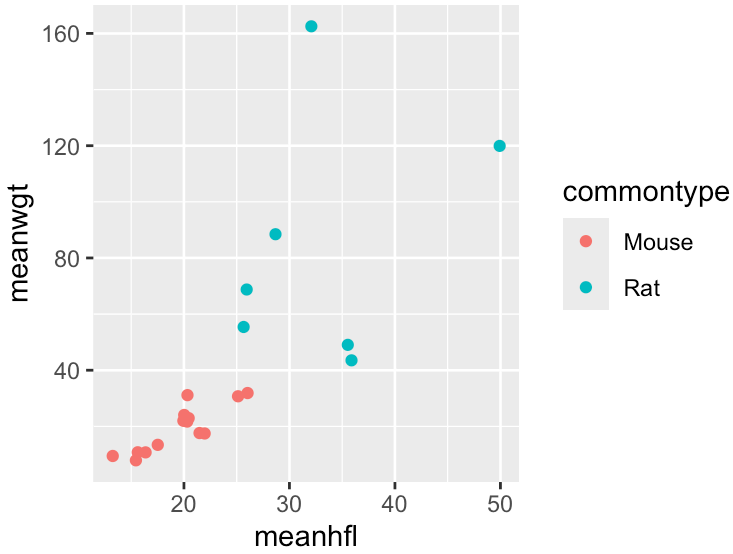
Suppose you want to make a histogram with this data. Explain, in your own words, why a binwidth of 150 is a bad choice.

A binwidth of 150 is a bad choice because it is too small to show the distribution of the data. There are too many bins to show the data well and it will appear cluttered and jittery.

Question 3 (6 points)

Consider the following scatterplot you made on Homework 2.

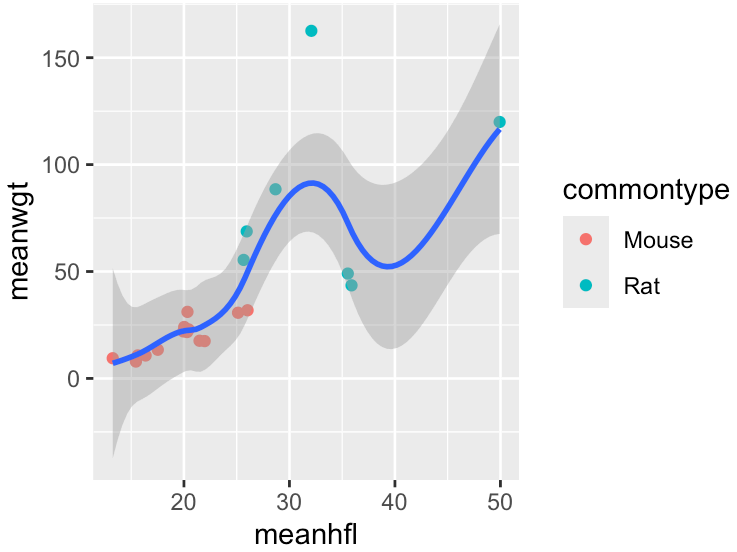
```
ggplot(data = species) +
  geom_point(mapping = aes(x = meanhfl, y = meanwgt, color = commontype))
```



a. (2 points) **Modify this scatterplot to draw a single curve on top of this data that approximates the trend of the data.**

```
species %>%
  ggplot() +
  geom_point(mapping = aes(x = meanhfl, y = meanwgt, color = commontype)) +
  geom_smooth(mapping = aes(x = meanhfl, y = meanwgt))
```

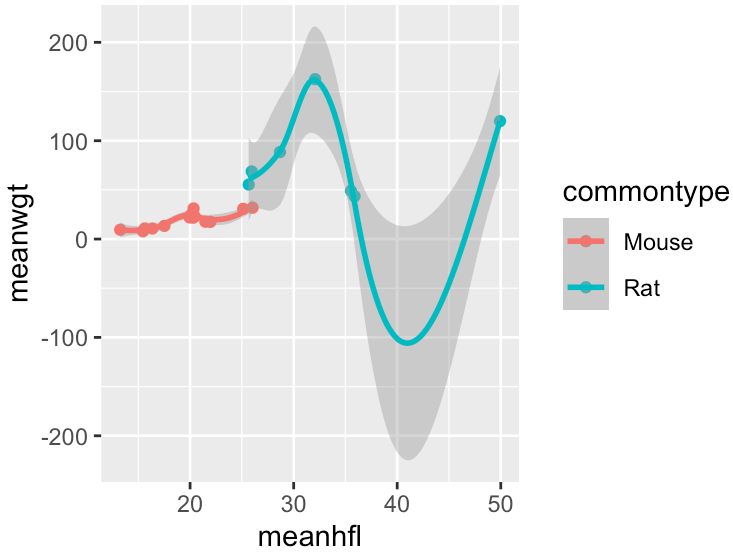
```
## `geom_smooth()` using method = 'loess' and formula = 'y ~ x'
```



b. (2 points) **Modify the scatterplot at the start of the question to draw two curves on top of this data that show the general trend of the points for each of the two commontypes - there should be one curve for mouse, and one curve for rat.**

```
ggplot(data = species) +
  geom_point(mapping = aes(x = meanhfl, y = meanwgt, color = commontype)) +
  geom_smooth(mapping = aes(x = meanhfl, y = meanwgt, color = commontype))
```

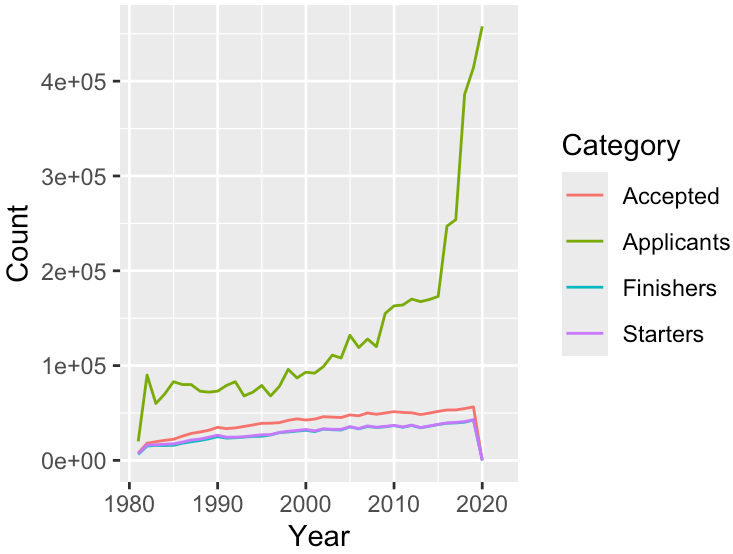
```
## `geom_smooth()` using method = 'loess' and formula = 'y ~ x'
```



c. (2 points) **Consider the London_marathon data. Make a line graph showing the counts of runners in different categories over time; there should be a separate line for each Category.**

```
london_marathon %>%
  ggplot() +
  geom_line(mapping = aes(x = Year, y = Count, color = Category))
```

```
## Warning: Removed 8 rows containing missing values or values outside the scale range
## (`geom_line()`).
```



Question 4 (4 points)

For each situation below, give two different examples of types of plots you could make. For each, be clear about how each variable involved will be represented in the plot.

a. (2 points) **You want to understand the distribution of one numeric variable.**

1. Histogram The one variable is the x-axis and the y-axis is the frequency of the variable. The observations are sorted into bins and the height of each bar represents the number of observations in that bin.
2. Boxplot The one variable is the y-axis and the x-axis is the categorical variable. The boxplot shows the distribution of the variable by showing the median, the interquartile range, and the outliers.

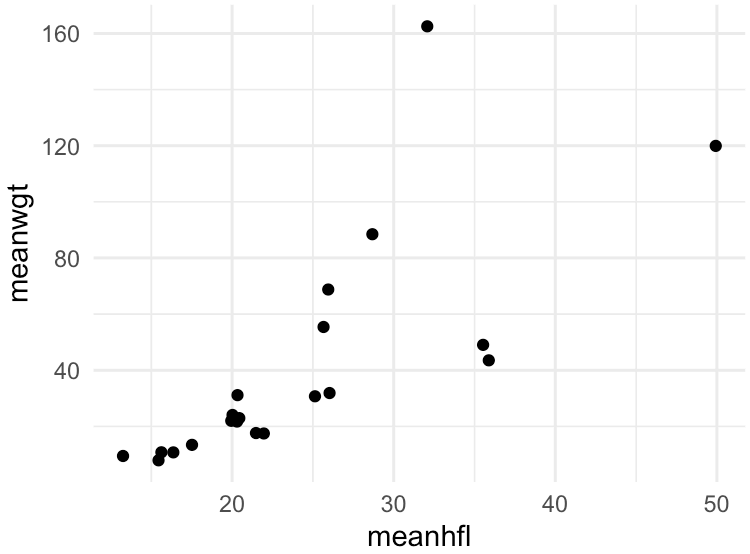
b. (2 points) **You want to understand the relationship between one categorical variable and one numerical variable**

1. Bar chart The categorical variable is the x-axis and the y-axis is the frequency of the variable. The bars represent the number of observations in each category.
2. Pie chart The catagorical variable is the different slices of the pie and the numerical variable is the proportion of the pie.

Question 5 (3 points)

You can change the style of your plot by adding a theme. For the follow plot, try out several of the other theme options (https://ggplot2.tidyverse.org/reference/ggtheme.html (https://ggplot2.tidyverse.org/reference/ggtheme.html)). Pick your favorite, display it here, and describe what you like about it.

```
ggplot(data = species) +
  geom_point(mapping = aes(x = meanhfl, y = meanwgt)) +
  theme_minimal()
```

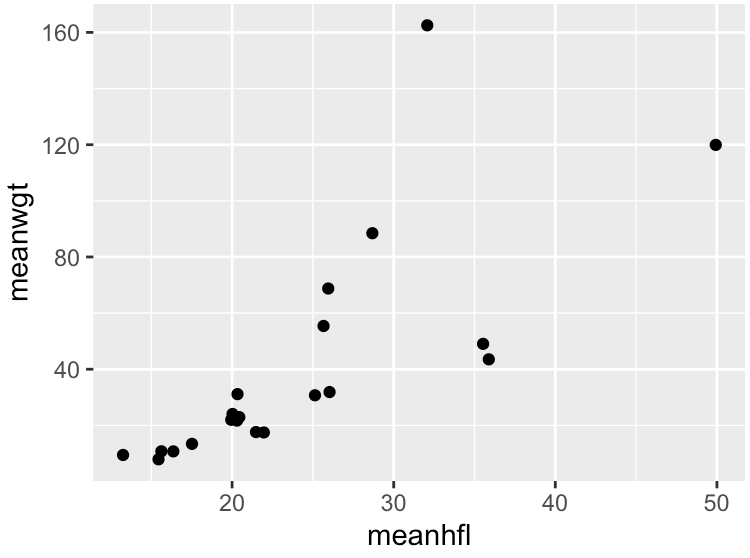


I like the minimal theme because it is simple and clean. It focuses on the data and makes the plot easy to read. It also has a grid which helps with readability.

Question 6 (7 points)

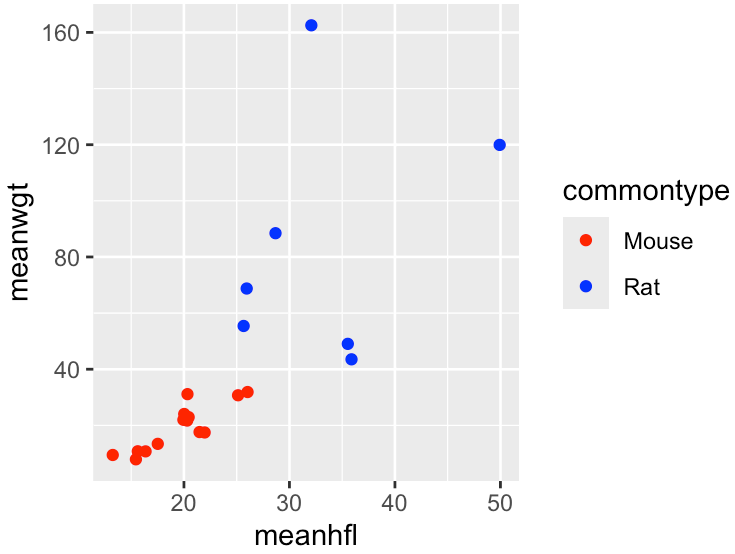
For this question, consider the following plot:

```
ggplot(data = species) +
  geom_point(mapping = aes(x = meanhfl, y = meanwgt))
```



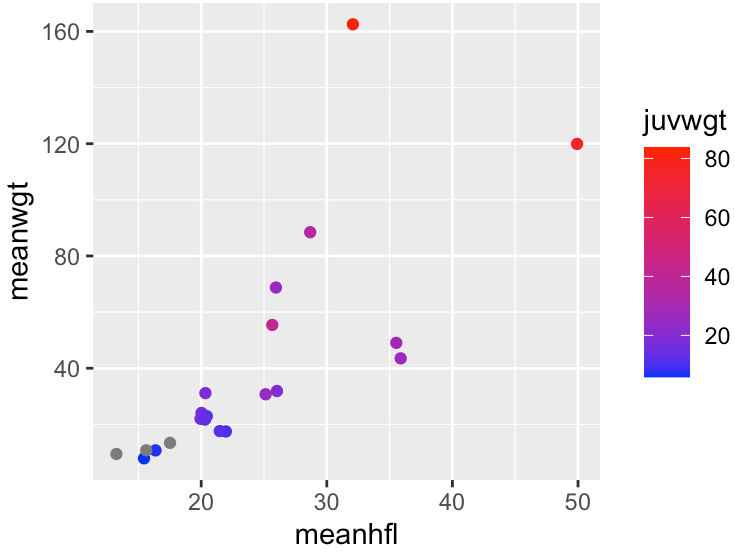
a. (2 points) **Add a third categorical variable to this plot using color, and change the default colors used to be some other colors of your choice.**

```
ggplot(data = species) +
  geom_point(mapping = aes(x = meanhfl, y = meanwgt, color = commontype)) +
  scale_color_manual(values = c("Mouse" = "red", "Rat" = "blue"))
```



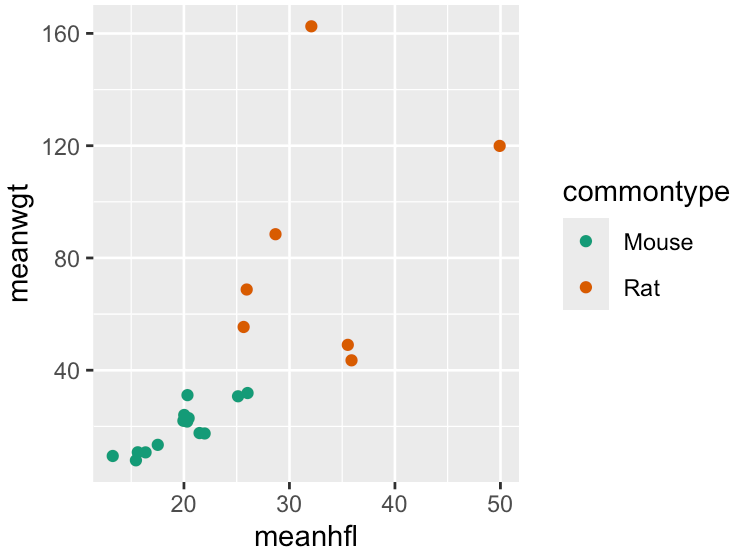
b. (2 points) **Add a third numeric variable to this plot using color, and change the default colors used at each end of the color gradient to be some other colors of your choice.**

```
ggplot(data = species) +
  geom_point(mapping = aes(x = meanhfl, y = meanwgt, color = juvwgt)) +
  scale_color_gradient(low = "blue", high = "red")
```



c. (3 points) **Add a third categorical variable to this plot using color, and change the default colors used to be one of the standard color palettes in the RColorBrewer package.**

```
ggplot(data = species) +
  geom_point(mapping = aes(x = meanhfl, y = meanwgt, color = commontype)) +
  scale_color_brewer(palette = "Dark2")
```



Question 7 (3 points)

Explain, in your own words, what a weighted average is. Give an example (different from any we discussed in class or saw in the activity) of when you might want to do a weighted average.

A weighted average is an average that takes into account the relative importance of each value in the data set. It is calculated by multiplying each value by its weight and then dividing by the sum of the weights.

Question 8 (17 points)

a. (2 points) **Explain, in your own words, what the following code is doing and why you get the answer you do.**

```
mean(c(TRUE, FALSE, FALSE, TRUE, FALSE, FALSE, FALSE, TRUE))
```

```
## [1] 0.375
```

This is finding the mean of the boolean values TRUE and FALSE. TRUE is 1 and FALSE is 0, so the mean is 0.375.

b. (2 points) **Explain, in your own words, what the data type is in the new column is_small that is created by the following code, and why it is that data type.**

```
species %>%
  mutate(is_small = (meanwgt < 50))
```

```
## # A tibble: 21 × 8
##    scientificname           commonname          commontype granivore meanhfl meanwgt
juvwgt is_small
##    <chr>                    <chr>               <chr>          <dbl>   <dbl>   <dbl>
<dbl> <lgl>
##  1 Baiomys taylori          Northern pygmy mou… Mouse              1    13.3    9.45
NA   TRUE
##  2 Chaetodipus baileyi      Bailey's pocket mo… Mouse              1    26.0   31.9
19.0 TRUE
##  3 Chaetodipus hispidus     Hispid pocket mouse Mouse              1    25.1   30.7
24   TRUE
##  4 Chaetodipus intermedius  Rock pocket mouse   Mouse              1    22.0   17.5
10   TRUE
##  5 Chaetodipus penicillatus Desert pocket mouse Mouse              1    21.5   17.6
11.7 TRUE
##  6 Dipodomys merriami       Merriam's kangaroo… Rat                1    35.9   43.5
26.4 TRUE
##  7 Dipodomys ordii          Ord's kangaroo rat  Rat                1    35.5   49.0
29.5 TRUE
##  8 Dipodomys spectabilis    Banner-tailed kang… Rat                1    49.9  120.
76.8 FALSE
##  9 Neotoma albigula         White-throated woo… Rat                0    32.1  163.
83.8 FALSE
## 10 Onychomys leucogaster    Northern Grasshopp… Mouse              0    20.3   31.1
18.4 TRUE
## # ℹ 11 more rows
```

The data type in the new column is a logical (lgl) data type. This is because the code is comparing the meanwgt column to 50 and returning a boolean value.

c. (2 points) **Explain, in your own words, what the following code is doing and why you get the answer that you do. What does the single number you get in your answer tell you about your data?**

```
species %>%
  mutate(is_small = (meanwgt < 50)) %>%
  summarize(x = mean(is_small))
```

```
## # A tibble: 1 × 1
##       x
##   <dbl>
## 1 0.762
```

This code creates a logical column that is TRUE if the meanwgt is less than 50 and FALSE otherwise. It then finds the mean of this logical column, which is the fraction of TRUE values. In this case, the mean is 0.762, which means that 76.2% of the species have a mean weight less than 50.

d. (2 points) **Explain, in your own words, why the code here gives the same result as the previous question.**

```
species %>%
  summarize(x = mean(meanwgt < 50))
```

```
## # A tibble: 1 × 1
##       x
##   <dbl>
## 1 0.762
```

This code skips mutating the data set. Instead it directly calculates the mean of the expression meanwgt < 50. This code is equivalent to the previous question because it is calculating the same thing.

e. (3 points) **Using the artists data set, write code to create a data set with one row and two columns, where the first column contains the number of artists in the data set who's race is not White, and the second column contains the fraction of artists in the data set that are not White.**

```
artists %>%
  summarize(num_non_white = sum(artist_race != "White"),
            frac_non_white = mean(artist_race != "White"))
```

```
## # A tibble: 1 × 2
##   num_non_white frac_non_white
##           <int>          <dbl>
## 1           226         0.0715
```

f. (3 points) **Use code to perform the same calculations in the previous part, but for each combination of year and book separately. What do you observe about the frequency of non-white artists appearing in textbooks over time?**

```
artists %>%
  group_by(book, year) %>%
  summarize(num_non_white = sum(artist_race != "White"),
            frac_non_white = mean(artist_race != "White"))
```

```
## `summarise()` has grouped output by 'book'. You can override using the `.groups` argu
ment.
```

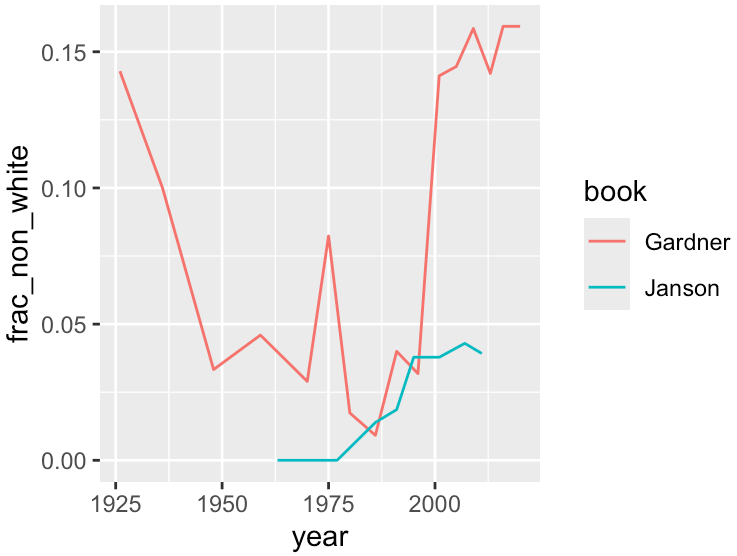
```
## # A tibble: 25 × 4
## # Groups:   book [2]
##    book     year num_non_white frac_non_white
##    <chr>   <dbl>         <int>          <dbl>
##  1 Gardner  1926             3        0.143
##  2 Gardner  1936             5        0.1
##  3 Gardner  1948             2        0.0333
##  4 Gardner  1959             4        0.0460
##  5 Gardner  1970             2        0.0290
##  6 Gardner  1975             7        0.0824
##  7 Gardner  1980             2        0.0174
##  8 Gardner  1986             1        0.00917
##  9 Gardner  1991             6        0.04
## 10 Gardner  1996             5        0.0318
## # ℹ 15 more rows
```

For Gardner, the franction of non-white artists seems to dip after the 1930s and then rise again. For Janson, it seems to increase more steadily over time.

> g. (3 points) **Make a line graph with year on the x-axis and the fraction of non-White artists on the y-axis. There should be two lines, one for each book. What do you observe in this plot?**

```
artists %>%
  group_by(book, year) %>%
  summarize(num_non_white = sum(artist_race != "White"),
            frac_non_white = mean(artist_race != "White")) %>%
  ggplot() +
  geom_line(mapping = aes(x = year, y = frac_non_white, color = book))
```

```
## `summarise()` has grouped output by 'book'. You can override using the `.groups` argu
ment.
```



The fraction of non-White artists in Gardner's book fluctuates significantly over time. There is a notable dip around the 1950s and 1960s, followed by a rise in the 1970s and a significant increase after the 1990s. The fraction of non-White artists in Janson's book starts to increase around the 1970s from zero and continues to rise steadily over time, though the increase is more gradual compared to Gardner's book. Both books remain under 20% by the end.

Question 9 (14 points)

> **This question concerns the pets_info.csv and pets_info2.csv files that were attached to the homework assignment.**

> a. (2 points) **Import the pets_info.csv file, without specifying any additional arguments inside the read_csv() function except the file name. Explain why the tibble you get is not what you want.**

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

```
## Warning: One or more parsing issues, call `problems()` on your data frame for detail
s, e.g.:
##   dat <- vroom(...)
##   problems(dat)
```

```
## Rows: 11 Columns: 3
## ── Column specification ────────────────────────────────────────────────────
──────────────
## Delimiter: ","
## chr (3): CSCI36, Spring 2024, Claremont McKenna College
##
## 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.
```

```
pets_info
```

```
## # A tibble: 11 × 3
##    CSCI36                                      `Spring 2024`              Claremo
nt McKenna Co…[1]
##    <chr>                                       <chr>                      <chr>
##  1 This file contains some information about pets <NA>                      <NA>
##  2 This data is artificial                     constructed for practice o… <NA>
##  3 Name                                        Age                        Specie
s,DateAdopted
##  4 Sparky                                      10                         Dog,3/
4/2021
##  5 Fluffy                                      4                          Cat,5/1
6/2020
##  6 Spot                                        3                          Dog,11/
23/2020
##  7 Buddy                                       7                          Dog,12/
3/2015
##  8 Fido                                        12                         Dog,4/
5/2016
##  9 Patches                                     3                          Cat,1/1
4/2019
## 10 Socks                                       6                          Cat,4/1
9/2018
## 11 Lassie                                      2                          Dog,7/1
7/2021
## # ℹ abbreviated name: [1]`Claremont McKenna College`
```

The tibble is not clean. The column names are not at the top of of the file and there is descriptive language that is messing with proper reading.

> b. (2 points) **Add a parameter to your import command to correctly import the data in pets_info.csv into a tibble, with the correct column names.**

```
pets_info <- read_csv("pets_info.csv", skip = 3)
```

```
## Rows: 8 Columns: 4
## ── Column specification ────────────────────────────────────────────────────
────────────────
## Delimiter: ","
## chr (3): Name, Species, DateAdopted
## dbl (1): Age
##
## 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.
```

```
pets_info
```

```
## # A tibble: 8 × 4
##   Name      Age Species DateAdopted
##   <chr>   <dbl> <chr>   <chr>
## 1 Sparky     10 Dog     3/4/2021
## 2 Fluffy      4 Cat     5/16/2020
## 3 Spot        3 Dog     11/23/2020
## 4 Buddy       7 Dog     12/3/2015
## 5 Fido       12 Dog     4/5/2016
## 6 Patches     3 Cat     1/14/2019
## 7 Socks       6 Cat     4/19/2018
## 8 Lassie      2 Dog     7/17/2021
```

c. (2 points) **Consider the pets_info2.csv file attached to the homework assignment; it does not have column names, but the columns are, in order, Name, Species, Age, and Date Adopted. Import the pets_info2.csv file and specify the column names as you do.**

```
pets_info2 <- read_csv("pets_info2.csv", col_names = c("Name", "Species", "Age", "Date A
dopted"))
```

```
## Rows: 8 Columns: 4
## ── Column specification ────────────────────────────────────────────────────
────────────────
## Delimiter: ","
## chr (3): Name, Species, Date Adopted
## dbl (1): Age
##
## 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.
```

```
pets_info2
```

```
## # A tibble: 8 × 4
##   Name    Species   Age `Date Adopted`
##   <chr>   <chr>   <dbl> <chr>
## 1 Sparky  Dog        10 3/4/2021
## 2 Fluffy  Cat         4 5/16/2020
## 3 Spot    Dog         3 11/23/2020
## 4 Buddy   Dog         7 12/3/2015
## 5 Fido    Dog        12 4/5/2016
## 6 Patches Cat         3 1/14/2019
## 7 Socks   Cat         6 4/19/2018
## 8 Lassie  Dog         2 7/17/2021
```

d. (2 points) **Explain, in your own words, the benefits of using read_csv() instead of read.csv().**

Using the read_csv() function is better because it is faster, creates a tibble instead of a data frame, and chooses the correct data type for the columns.

e. (3 points) **What happens when one row of a csv file has a different number of commas than other rows? Explain in your own words. You can refer to the following two examples in your explanation if you'd like, though make sure your explanation is more general than just these two examples. Be sure to mention any errors or warnings that might occur.**

```
read_csv("a,b,c,d
         1,2,3,4
         5,6,7
         8,9,10,11")
```

```
## Warning: One or more parsing issues, call `problems()` on your data frame for detail
s, e.g.:
##   dat <- vroom(...)
##   problems(dat)
```

```
## Rows: 3 Columns: 4
## ── Column specification ────────────────────────────────────────────────────────
──────────────
## Delimiter: ","
## dbl (4): a, b, c, d
##
## 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.
```

```
## # A tibble: 3 × 4
##       a     b     c     d
##   <dbl> <dbl> <dbl> <dbl>
## 1     1     2     3     4
## 2     5     6     7    NA
## 3     8     9    10    11
```

```
read_csv("a,b,c,d
         1,2,3,4
         5,6,7,8,9
         10,11,12,13")
```

```
## Warning: One or more parsing issues, call `problems()` on your data frame for detail
s, e.g.:
##   dat <- vroom(...)
##   problems(dat)
```

```
## Rows: 3 Columns: 4
## ── Column specification ────────────────────────────────────────────────────────
────────────────
## Delimiter: ","
## dbl (3): a, b, c
## num (1): d
##
## 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.
```

```
## # A tibble: 3 × 4
##       a     b     c     d
##   <dbl> <dbl> <dbl> <dbl>
## 1     1     2     3     4
## 2     5     6     7    89
## 3    10    11    12    13
```

When a row in a CSV file has a different number of commas than other rows, it can cause issues during the import process. The read_csv() function will try to parse each row based on the number of columns in the header row. If a row has fewer commas, it will result in NA for the missing columns. If a row has more commas, it will cause a warning and the extra values will be ignored.

> f. (3 points) **In 2-4 sentences, explain in your own words when importing data directly using a URL is a good idea, and when importing data directly using a URL is a bad idea.**

Importing data directly using a URL is a good idea if the data is publicly available. It is also beneficial if the URL is stable and won't be moved or deleted. It is important that the data won't change over time. This ensures that analysis remains reproducible and the data source remains accessible. If these conditions aren't met, it may be better to download the data first and then import it into R.

Question 10 (6 points)

a. **Suppose you want to parse the string "$67,000" to sixty-seven thousand. How would you do this?**

```
parse_number("$67,000")
```

```
## [1] 67000
```

b. **Suppose you want to parse the string "67.000,00" to mean sixty-seven thousand. How would you do this?**

```
parse_number("67.000,00", locale = locale(grouping_mark = "."))
```

```
## [1] 67000
```

c. **Explain the difference between parse_double() and parse_number(). Give an example of when you should use parse_number(), and when you should use parse_double().**

parse_double() is used when you want to parse a string that represents a number with a decimal point. parse_number() is used when you want to parse a string that represents a number with a comma.

Question 11 (6 points)

a. (3 points) **Import the attached data set "Monthly_amounts.txt". After importing, correct the data types of the columns largest_amount and average_amount to be what they should be. Explain any warnings you get.**

```
monthly_amounts <- read_csv("Monthly_amounts.txt", skip = 1) %>%
  mutate(largest_amount = parse_number(largest_amount),
         average_amount = parse_number(average_amount))
```

```
## Warning: One or more parsing issues, call `problems()` on your data frame for detail
s, e.g.:
##   dat <- vroom(...)
##   problems(dat)
```

```
## Rows: 20 Columns: 3
## ── Column specification ────────────────────────────────────────────────────
────────────────
## Delimiter: ","
## chr (3): Year_month, largest_amount, average_amount
##
## 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.
```

```
## Warning: There were 2 warnings in `mutate()`.
## The first warning was:
## ℹ In argument: `largest_amount = parse_number(largest_amount)`.
## Caused by warning:
## ! 1 parsing failure.
## row col expected actual
##  18  -- a number      *
## ℹ Run `dplyr::last_dplyr_warnings()` to see the 1 remaining warning.
```

```
monthly_amounts
```

```
## # A tibble: 20 × 3
##    Year_month largest_amount average_amount
##    <chr>               <dbl>          <dbl>
##  1 2023-1                 63           45.7
##  2 2023-2                 56           35.2
##  3 2023-3                 54           34.2
##  4 2023-4                 12           11.6
##  5 2023-5                 NA           35.2
##  6 2023-6                 87           56.9
##  7 2023-7                 82           45.7
##  8 2023-8                 36           17.9
##  9 2023-9                 98          543
## 10 2023-10                16            9.8
## 11 2023-11                78           50.2
## 12 2023-12                45           43.6
## 13 2024-1                 35           31.2
## 14 2024-2                 91           NA
## 15 2024-3                 45           34.5
## 16 2024-4                 67           34.6
## 17 2024-5                 34           21.9
## 18 2024-6                 NA           49.8
## 19 2024-7                 35           23.8
## 20 2024-8                 45           43
```

```
problems(monthly_amounts)
```

There are some warnings largest_amount because a value could not be parsed as a number because it contained a (*). For average_amount, a value could not be parsed as a number because it contained a (-).

b. (3 points) **For the same data set, specify additional option(s) while importing so that, after importing but before using any other functions, the largest_amount and average_amount columns are both a numeric data type. Then, correct the data type of the largest_amount column to be what it should be.**

```
monthly_amounts <- read_csv("Monthly_amounts.txt", skip = 1, col_types = cols(
  largest_amount = col_number(),
  average_amount = col_number()
))
```

```
## Warning: One or more parsing issues, call `problems()` on your data frame for detail
s, e.g.:
##   dat <- vroom(...)
##   problems(dat)
```

```
monthly_amounts <- monthly_amounts %>%
  mutate(largest_amount = parse_integer(as.character(largest_amount)))
# Note: These instructions were really confusing to me. I did my best to interpret them.
monthly_amounts
```

```
## # A tibble: 20 × 3
##    Year_month largest_amount average_amount
##    <chr>               <int>          <dbl>
##  1 2023-1                 63           45.7
##  2 2023-2                 56           35.2
##  3 2023-3                 54           34.2
##  4 2023-4                 12           11.6
##  5 2023-5                 NA           35.2
##  6 2023-6                 87           56.9
##  7 2023-7                 82           45.7
##  8 2023-8                 36           17.9
##  9 2023-9                 98          543
## 10 2023-10                16            9.8
## 11 2023-11                78           50.2
## 12 2023-12                45           43.6
## 13 2024-1                 35           31.2
## 14 2024-2                 91           NA
## 15 2024-3                 45           34.5
## 16 2024-4                 67           34.6
## 17 2024-5                 34           21.9
## 18 2024-6                 NA           49.8
## 19 2024-7                 35           23.8
## 20 2024-8                 45           43
```

Question 12 (9 points)

a. (2 points) **Explain, in your own words, what the fill() function does. Describe an example, different from any we talked about in class, of when you might want to use this function.**

The fill() function is used to fill missing values in a column with the last observed value. This is useful when you want to fill in missing values in a column with the last known value. Some data sets have missing values that are not unknown, but need to be cleaned by looking at the last known value.

Example: month, date, amount June,1,100 ,2,101 ,3,105 ,4,102

We can tell that it should be June.

b. (2 points) **Give some examples of data that is implicitly missing from this tibble.**

```
## # A tibble: 37 × 4
##    School Day_of_week TimeOfDay NumVisitors
##    <chr>  <chr>       <chr>           <dbl>
##  1 CMC    Monday      Morning            23
##  2 CMC    Monday      Afternoon          23
##  3 CMC    Monday      Evening            32
##  4 CMC    Tuesday     Morning            42
##  5 CMC    Tuesday     Afternoon          11
##  6 CMC    Tuesday     Evening            12
##  7 CMC    Wednesday   Evening             8
##  8 CMC    Thursday    Morning             3
##  9 CMC    Thursday    Afternoon           0
## 10 CMC    Thursday    Evening            14
## # ℹ 27 more rows
```

Some examples: CMC Wednesday Morning, CMC Wednesday Afternoon, CMC Friday Evening, etc.

c. **(3 points) Use the command we learned in class to add rows to this tibble that correspond to the missing data; how many rows do you get? Explain why you get this number.**

```
campus_visitors %>%
  complete(School, Day_of_week, TimeOfDay)
```

```
## # A tibble: 84 × 4
##    School Day_of_week TimeOfDay NumVisitors
##    <chr>  <chr>       <chr>           <dbl>
##  1 CMC    Friday      Afternoon          35
##  2 CMC    Friday      Evening            NA
##  3 CMC    Friday      Morning             3
##  4 CMC    Monday      Afternoon          23
##  5 CMC    Monday      Evening            32
##  6 CMC    Monday      Morning            23
##  7 CMC    Saturday    Afternoon          35
##  8 CMC    Saturday    Evening            NA
##  9 CMC    Saturday    Morning            NA
## 10 CMC    Sunday      Afternoon          NA
## # ℹ 74 more rows
```

Got 84 rows because there are 3 schools, 7 days of the week, and 4 times of day. So 3 * 7 * 4 = 84.

d. **(2 points) Do you think there is still data implicitly missing from the tibble you made in part b? Explain. Hint: Use what other knowledge you might have about the columns/values involved.**

It looks like Pomona has no data for any of the times of day.

Question 13 (9 points)

As we start importing data from more sources, it's important to have information about the data and how it was collected. Read the research paper "Datasheets for Datasets," available at https://arxiv.org/abs/1803.09010 (https://arxiv.org/abs/1803.09010).

a. **Explain why it's important to fully document the creation of data sets, and what ethical harms may result if data set are not thoroughly documented.**

Fully documenting the creation of data sets is important for ensuring transparency, reproducibility, and accountability. Without documentation, readers may misinterpret the data, leading to incorrect conclusions and potentially harmful decisions. Ethical harms include biased outcomes, privacy violations, and misuse of data, which can affect vulnerable groups.

b. **Of all the questions listed in Section 3, which one surprised you most? Explain why, despite being surprising to you, having an answer to that question in a datasheet is important.**

The question that surprised me most was “Were any ethical review processes conducted (e.g., by an institutional review board)?” This question is important because it ensures that the data collection process adhered to ethical standards, protecting the rights and well-being of individuals involved. It highlights the importance of considering ethical implications in data collection and usage. I know CMC has an IRB. I’m working with them right now on a study.

c. **For the artists data set from last week’s homework (See https://github.com/rfordatascience/tidytuesday/blob/master/data/2023/2023-01-17/readme.md (https://github.com/rfordatascience/tidytuesday/blob/master/data/2023/2023-01-17/readme.md) and the various other links included on that page, such as https://research.repository.duke.edu/concern/datasets/q811kk70n?locale=en (https://research.repository.duke.edu/concern/datasets/q811kk70n?locale=en)): how thorough was the documentation accompanying this data set? Give some examples of questions from Section 3 that you were able to find answers to, and give some examples of questions from Section 3 that you were not able to find answers to.**

The documentation for the artists data set was good, providing variable descriptions, data sources, and the purpose of the data collection.

Questions answered: What data does the dataset contain?, What is the source of the data?, What is the purpose of the dataset? Questions not answered: Were any ethical review processes conducted?, What preprocessing/cleaning/labeling was done?, Are there any known errors or uncertainties in the data?