Week 5 Assignment

Ellen Bledsoe

2024-02-19

Week 5 Assignment

Assignment Exercises

Set-up

Load the packages we will need. You can either load all of them individually (`readr`, `dplyr`, `ggplot2`) or load the `tidyverse` package.

```
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.4     v readr     2.1.5
## v forcats   1.0.0     v stringr   1.5.1
## v ggplot2   3.4.4     v tibble    3.2.1
## v lubridate 1.9.3     v tidyr     1.3.0
## v purrr     1.0.2
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

1. Acacia and Ants (20 pts)

Read in the acacia data frame by running the following code chunk.

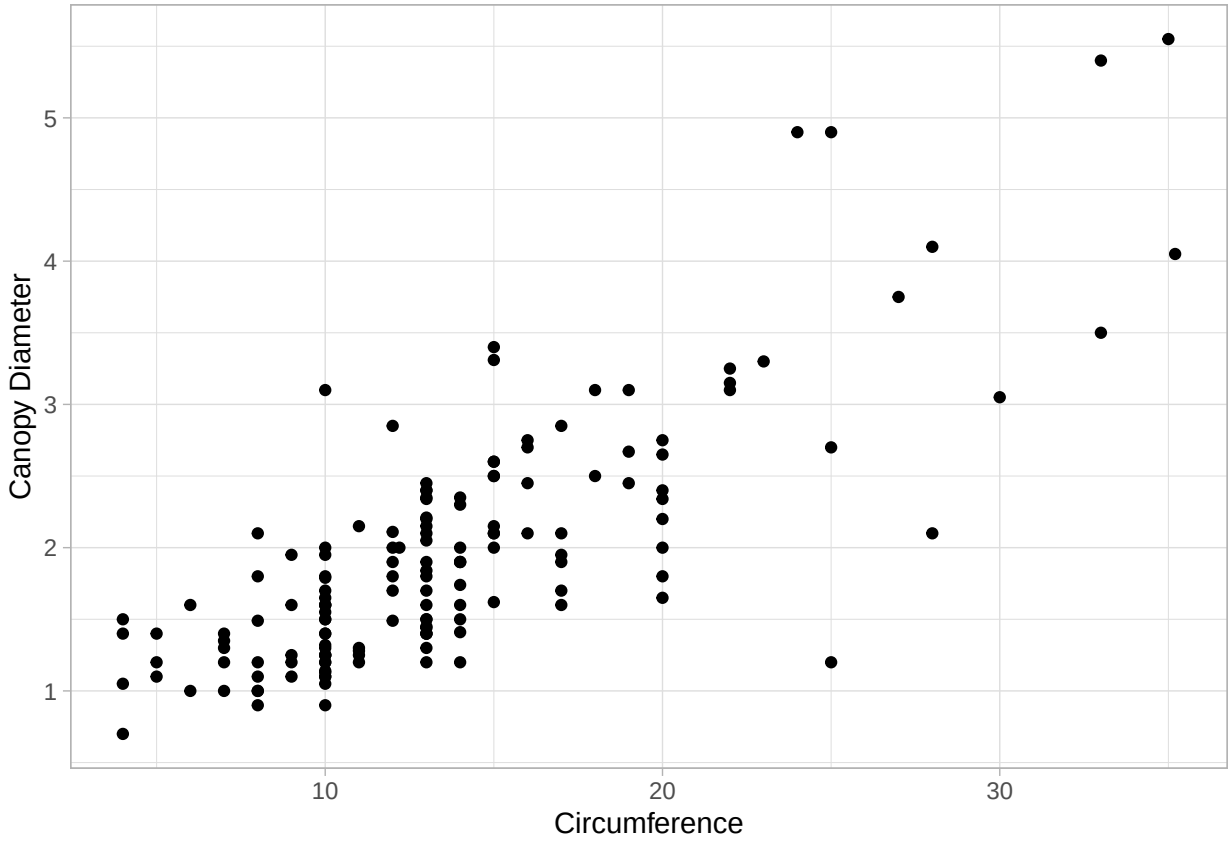
```
acacia <- read_tsv("ACACIA_DREPANOLOBIUM_SURVEY.txt", na = c("", "dead"))
```

```
## Rows: 157 Columns: 15
## -- Column specification --------------------------------------------------------
## Delimiter: "\t"
## chr  (4): SITE, TREATMENT, PLOT, ANT
## dbl (11): SURVEY, YEAR, BLOCK, ID, HEIGHT, AXIS1, AXIS2, CIRC, FLOWERS, BUDS...
##
## 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. Make a scatter plot with `CIRC` on the x axis and `AXIS1` (the maximum canopy width) on the y axis. Label the x axis "Circumference" and the y axis "Canopy Diameter".

```
ggplot(acacia, aes(x=CIRC, y=AXIS1))+
  geom_point()+
  labs(x="Circumference", y = "Canopy Diameter")+
  theme_light()
```

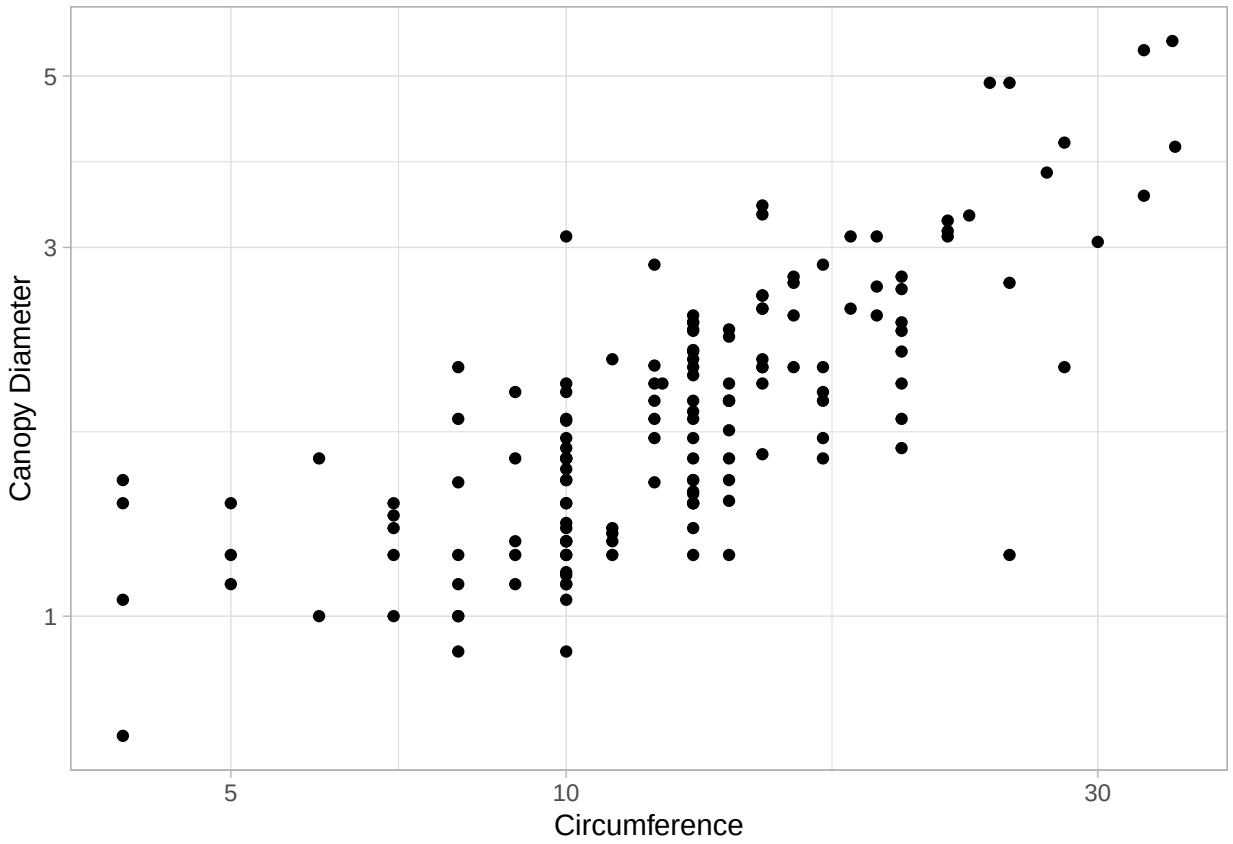
```
## Warning: Removed 4 rows containing missing values (`geom_point()`).
```



b. The same plot as (a), but with both axes scaled logarithmically (using `scale_x_log10` and `scale_y_log10`).

```
ggplot(acacia, aes(x=CIRC, y=AXIS1))+
  geom_point()+
  labs(x="Circumference", y = "Canopy Diameter")+
  scale_x_log10()+
  scale_y_log10()+
  theme_light()
```

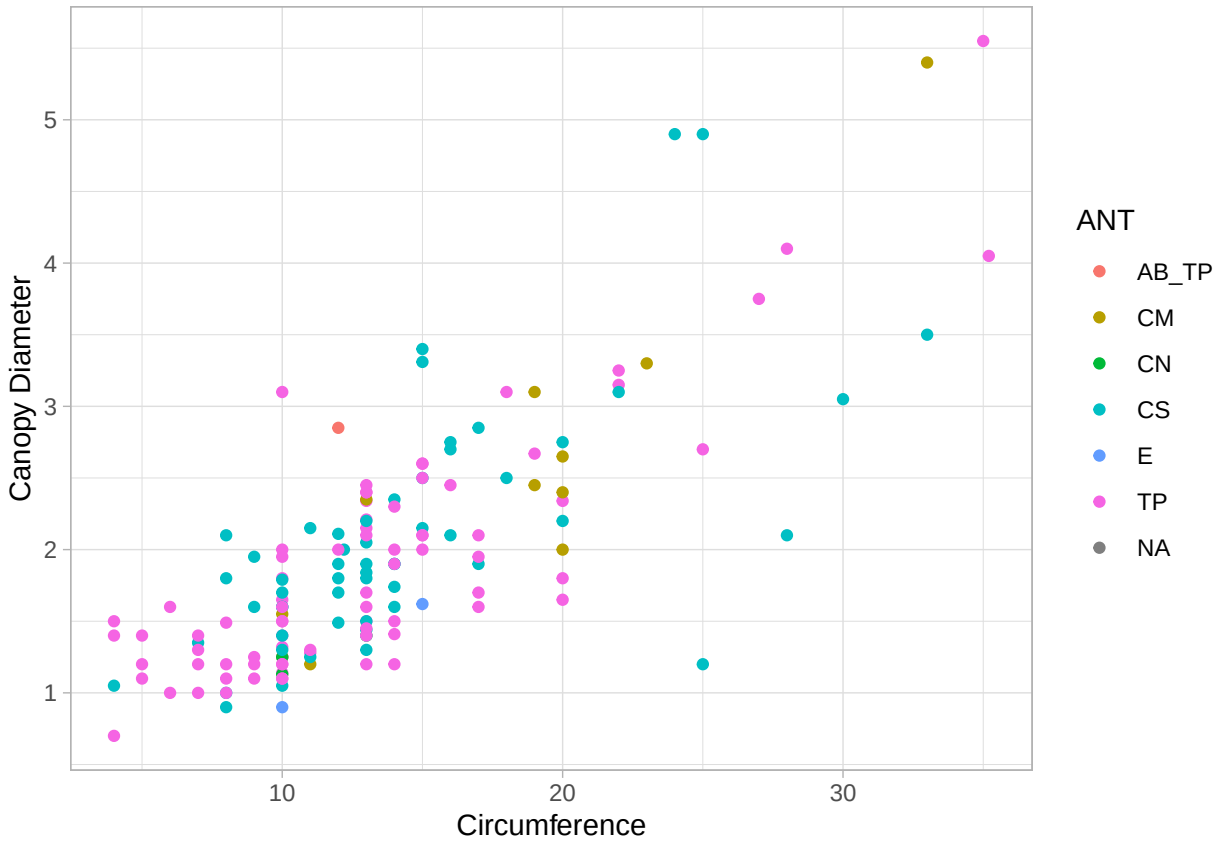
```
## Warning: Removed 4 rows containing missing values (`geom_point()`).
```



c. The same plot as (a), but with points colored based on the `ANT` column (the species of ant symbiont living with the acacia)

```
ggplot(acacia, aes(x=CIRC, y=AXIS1, color = ANT))+
  geom_point()+
  labs(x="Circumference", y = "Canopy Diameter")+
  theme_light()
```

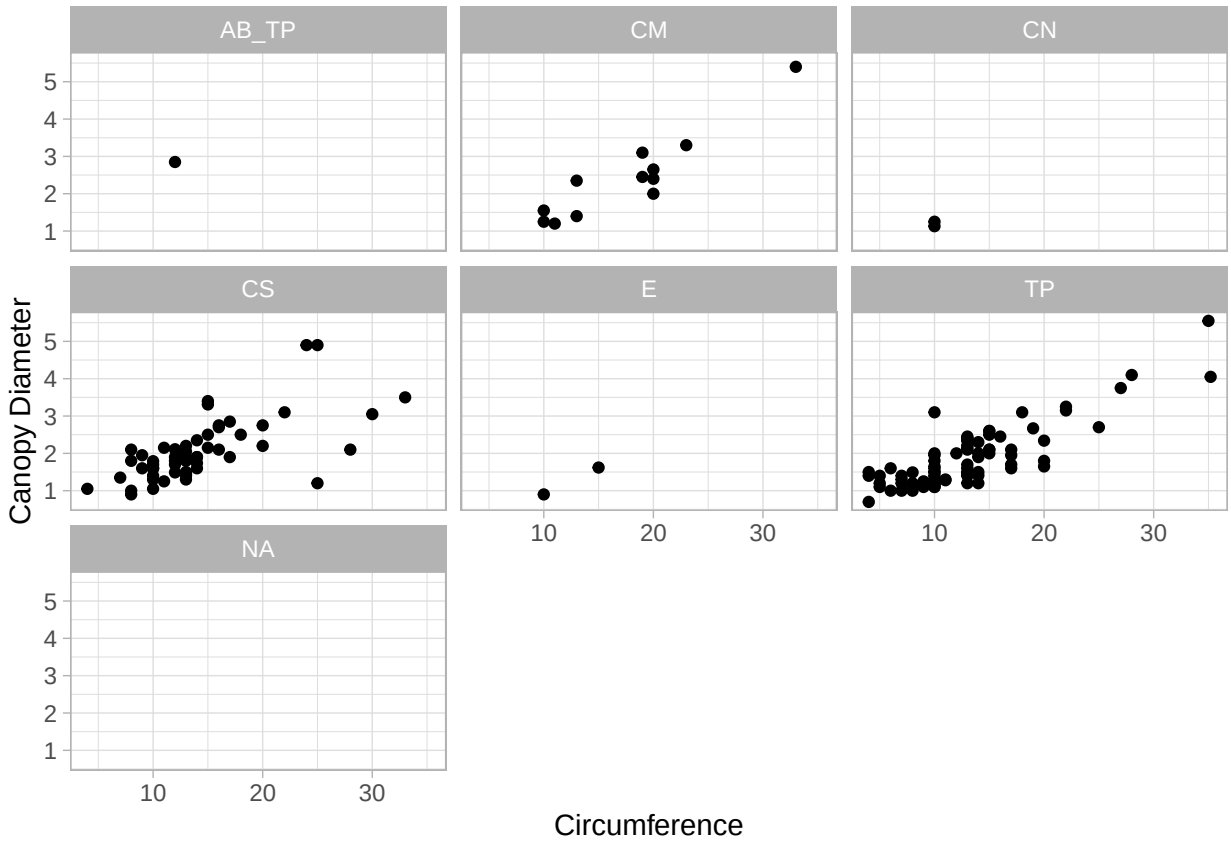
```
## Warning: Removed 4 rows containing missing values (`geom_point()`).
```



d. The same plot as (c), but instead of different colors show different species of ant (values of `ANT`) each in a separate subplot.

```
ggplot(acacia, aes(x=CIRC, y=AXIS1))+
  geom_point()+
  labs(x="Circumference", y = "Canopy Diameter")+
  facet_wrap(~ANT)+
  theme_light()
```

```
## Warning: Removed 4 rows containing missing values (`geom_point()`).
```



e. The same plot as (d) but add a simple model of the data by adding `geom_smooth`.

```
ggplot(acacia, aes(x=CIRC, y=AXIS1))+
  geom_point()+
  labs(x="Circumference", y = "Canopy Diameter")+
  facet_wrap(~ANT)+
  geom_smooth()+
  theme_light()
```

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

```
## Warning: Removed 4 rows containing non-finite values (`stat_smooth()`).
```

```
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = parametric,
## : span too small.  fewer data values than degrees of freedom.
```

```
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = parametric,
## : at 9.975
```

```
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = parametric,
## : radius 0.000625
```

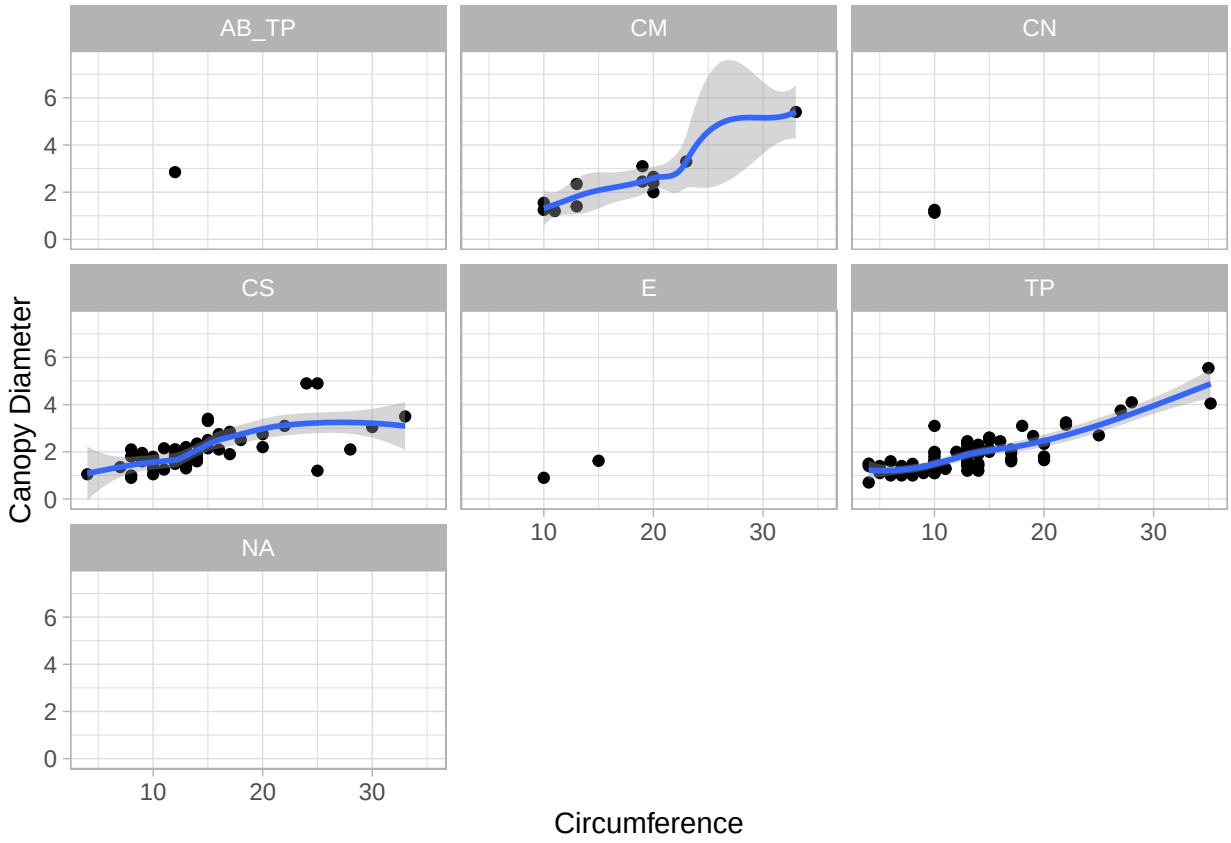
```
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = parametric,
## : all data on boundary of neighborhood. make span bigger
```

```
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = parametric,
## : pseudoinverse used at 9.975
```

```
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = parametric,
## : neighborhood radius 0.025
```

```
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = parametric,
```

```
## : reciprocal condition number 1

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = parametric,
## : at 15.025

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = parametric,
## : radius 0.000625

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = parametric,
## : all data on boundary of neighborhood. make span bigger

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = parametric,
## : There are other near singularities as well. 0.000625

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = parametric,
## : zero-width neighborhood. make span bigger

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = parametric,
## : zero-width neighborhood. make span bigger

## Warning: Computation failed in `stat_smooth()`
## Caused by error in `predLoess()`:
## ! NA/NaN/Inf in foreign function call (arg 5)

## Warning: Removed 4 rows containing missing values (`geom_point()`).
```



2. Mass vs. Metabolism (20 pts)

The relationship between the body size of an organism and its metabolic rate is one of the most well studied and still most controversial areas of organismal physiology. We want to graph this relationship in the

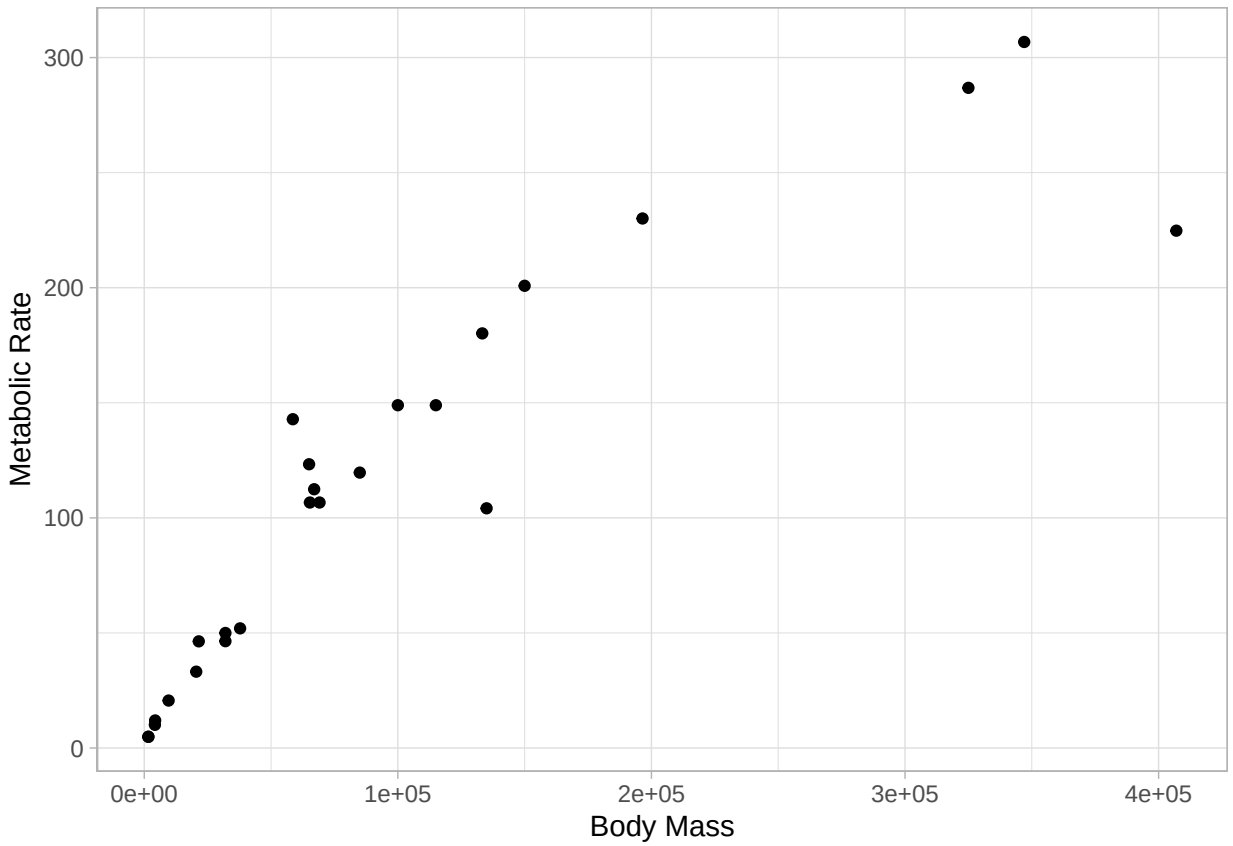
Artiodactyla using a subset of data from a large compilation of body size data (Savage et al. 2004). Run this code chunk to get started.

```
size_mr_data <- data.frame(
  body_mass = c(32000, 37800, 347000, 4200, 196500, 100000,
    4290, 32000, 65000, 69125, 9600, 133300, 150000, 407000,
    115000, 67000,325000, 21500, 58588, 65320, 85000, 135000,
    20500, 1613, 1618),
  metabolic_rate = c(49.984, 51.981, 306.770, 10.075, 230.073,
    148.949, 11.966, 46.414, 123.287, 106.663, 20.619, 180.150,
    200.830, 224.779, 148.940, 112.430, 286.847, 46.347,
    142.863, 106.670, 119.660, 104.150, 33.165, 4.900, 4.865),
  family = c("Antilocapridae", "Antilocapridae", "Bovidae",
    "Bovidae", "Bovidae", "Bovidae", "Bovidae", "Bovidae",
    "Bovidae", "Bovidae", "Bovidae", "Bovidae", "Bovidae",
    "Camelidae", "Camelidae", "Canidae", "Cervidae",
    "Cervidae", "Cervidae", "Cervidae", "Cervidae", "Suidae",
    "Tayassuidae", "Tragulidae", "Tragulidae"))
```

Make the following plots with appropriate axis labels:

a. A plot of body mass vs. metabolic rate

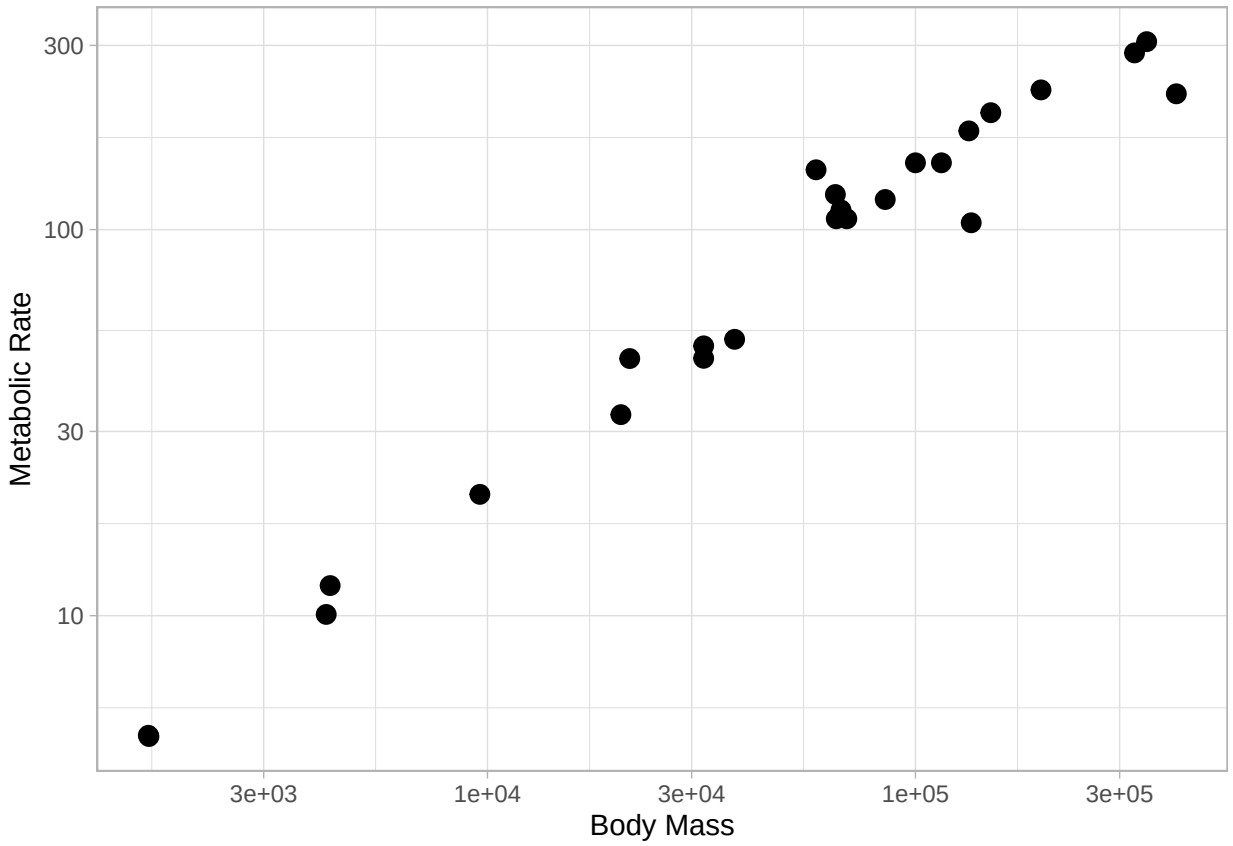
```
ggplot(data = size_mr_data, mapping = aes(x=body_mass, y = metabolic_rate))+
  geom_point()+
  labs(x= "Body Mass", y = "Metabolic Rate")+
  theme_light()
```



b. A plot of body mass vs. metabolic rate, with log10 scaled axes (this stretches the axis, but keeps the

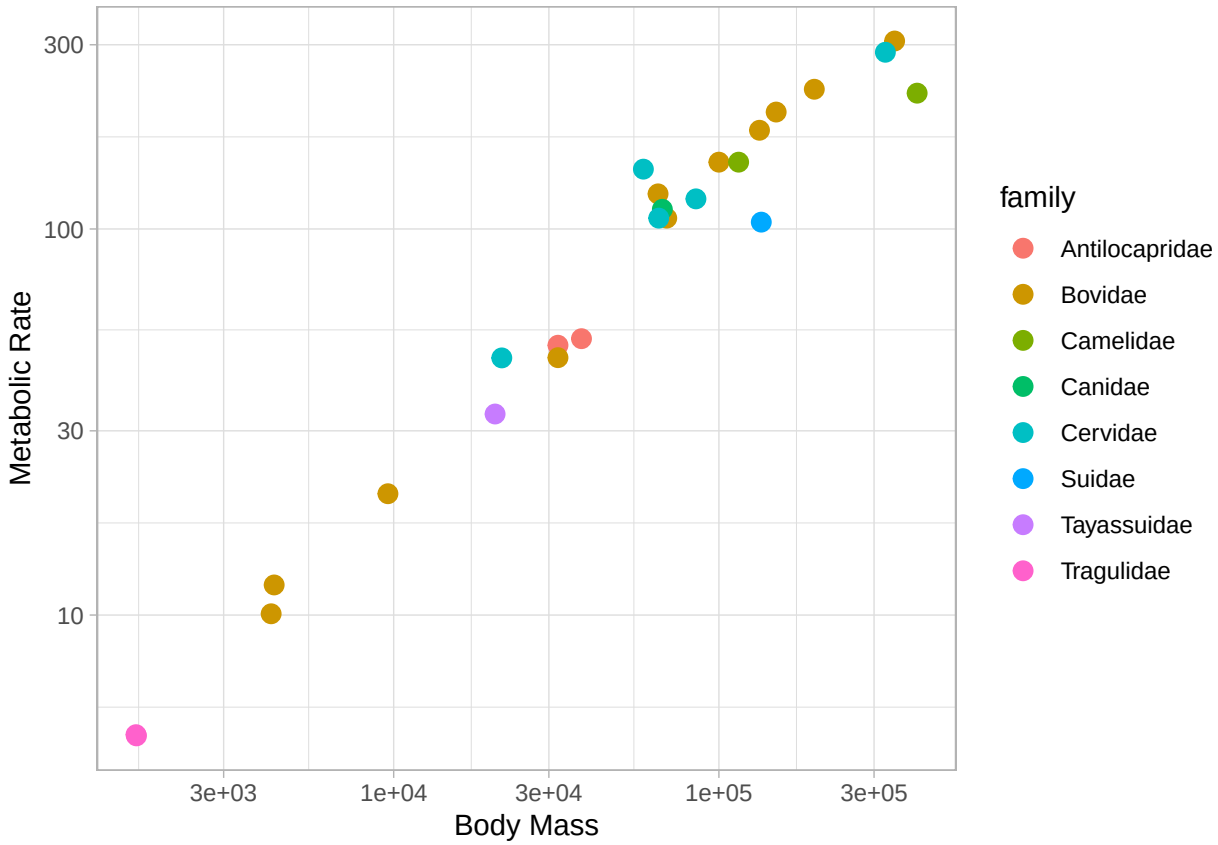
numbers on the original scale), and the point size set to 3.

```
ggplot(data = size_mr_data, mapping = aes(x=body_mass, y = metabolic_rate))+
  geom_point(size = 3)+
  scale_x_log10()+
  scale_y_log10()+
  labs(x= "Body Mass", y = "Metabolic Rate")+
  theme_light()
```



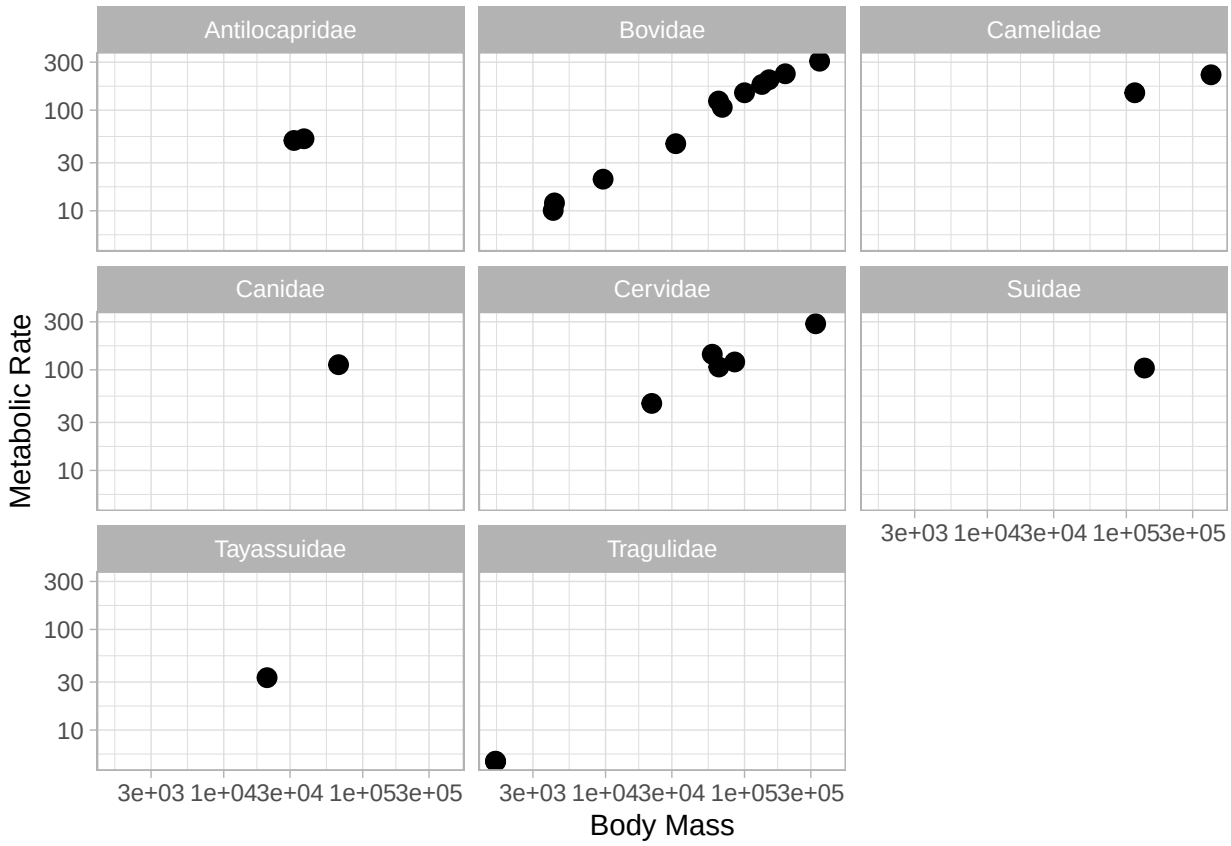
c. The same plot as (b), but with the different families indicated using color.

```
ggplot(data = size_mr_data, mapping = aes(x=body_mass, y = metabolic_rate, color = family))+
  geom_point(size = 3)+
  scale_x_log10()+
  scale_y_log10()+
  labs(x= "Body Mass", y = "Metabolic Rate")+
  theme_light()
```



d. The same plot as (b), but with the different families each in their own subplot.

```
ggplot(data = size_mr_data, mapping = aes(x=body_mass, y = metabolic_rate))+
  geom_point(size = 3)+
  scale_x_log10()+
  scale_y_log10()+
  facet_wrap(~family)+
  labs(x= "Body Mass", y = "Metabolic Rate")+
  theme_light()
```

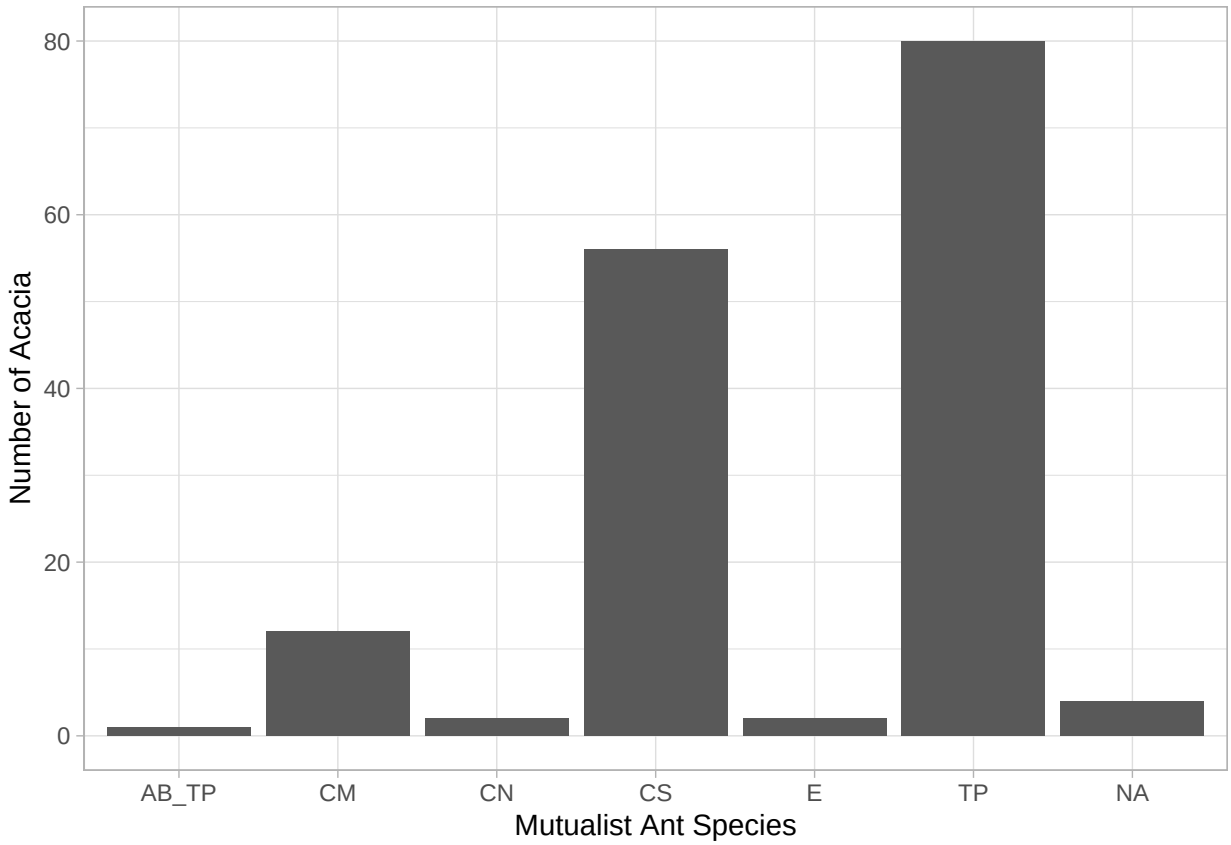


3. Acacia and Ants Histograms (20 pts)

In this exercise, we will be making a number of different histograms with the `acacia` dataset.

a. Make a bar plot of the number of acacia with each mutualist ant species (using the `ANT` column).

```
ggplot(data = acacia, mapping = aes(x=ANT))+
  geom_bar()+
  theme_light()+
  labs(x="Mutualist Ant Species", y= "Number of Acacia")
```

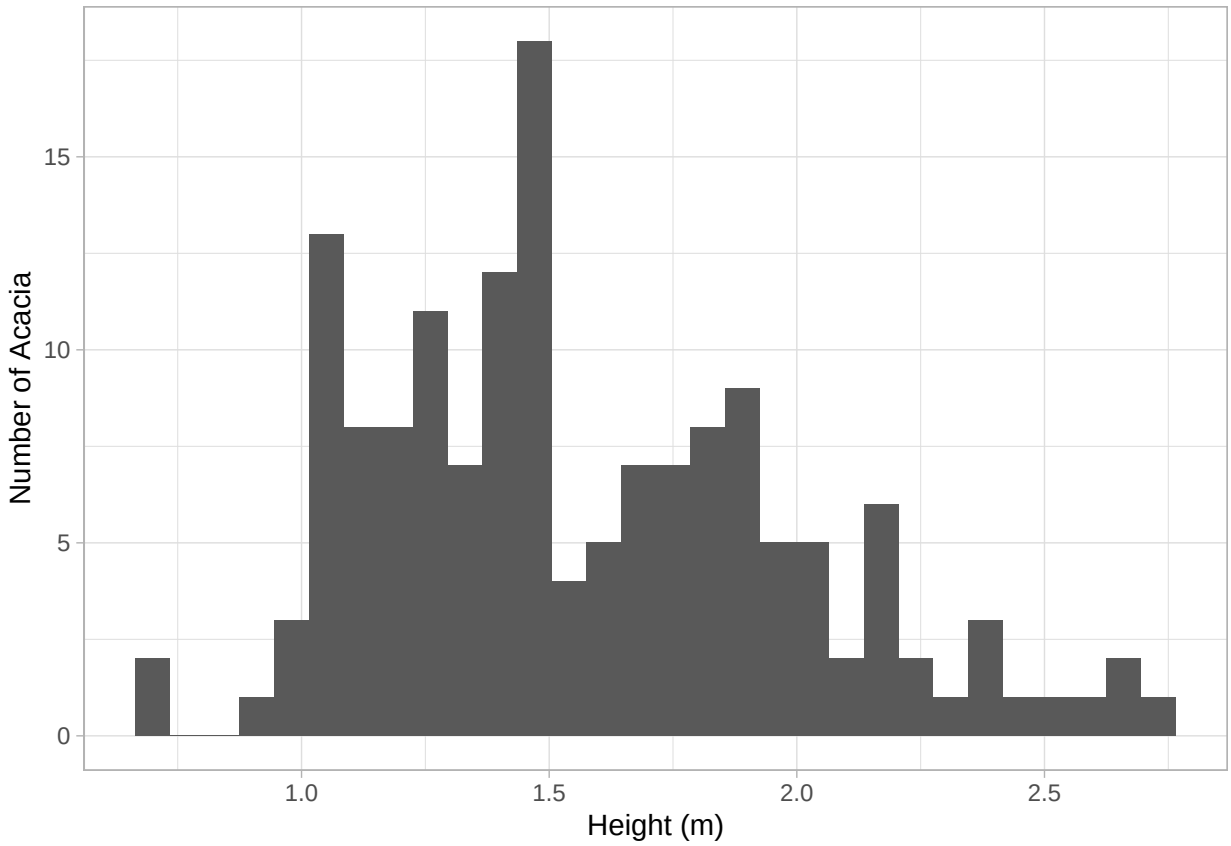


b. Make a histogram of the height of acacia (using the `HEIGHT` column). Label the x axis "Height (m)" and the y axis "Number of Acacia".

```
ggplot(data = acacia, mapping = aes(x=HEIGHT))+
  geom_histogram()+
  theme_light()+
  labs(x="Height (m)", y= "Number of Acacia")
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

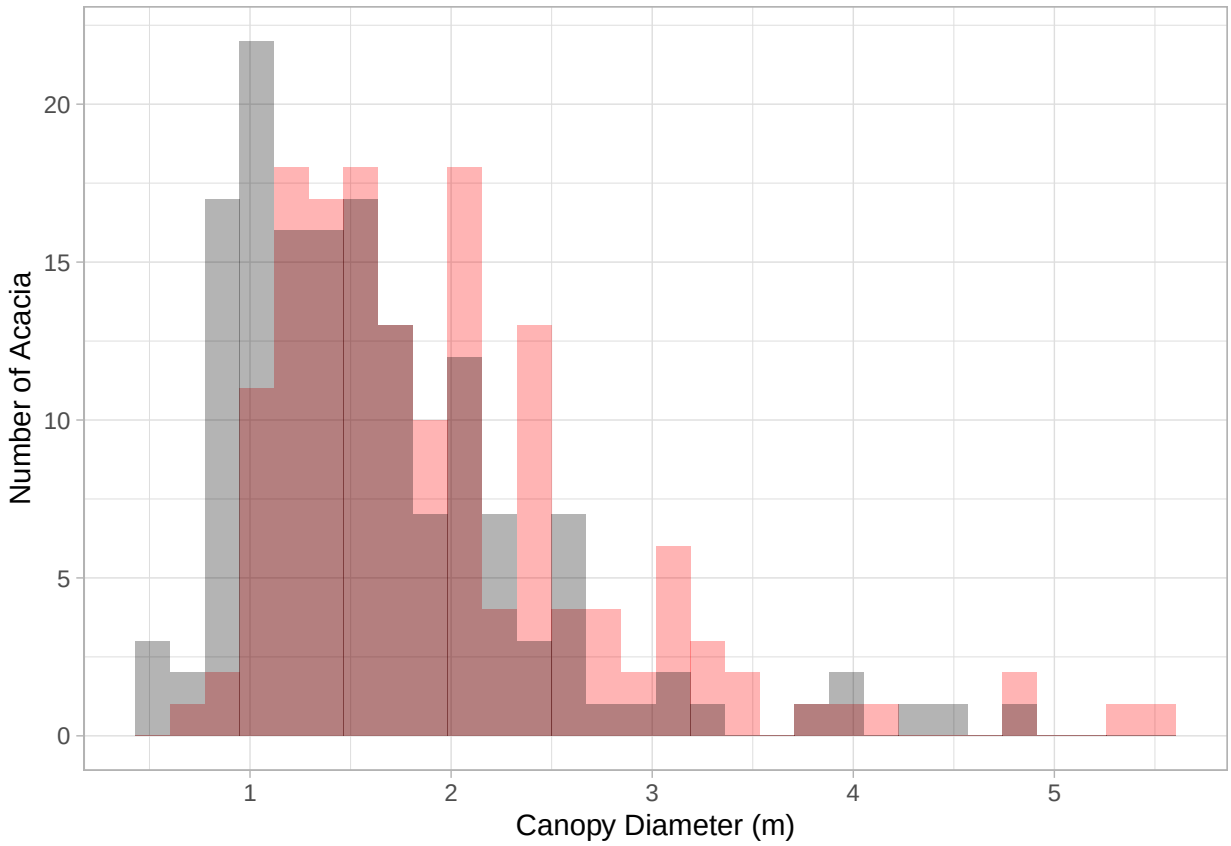
```
## Warning: Removed 4 rows containing non-finite values (`stat_bin()`).
```



c. Make a plot that shows histograms of both `AXIS1` and `AXIS2`. Due to the way the data is structured, you'll need to add a 2nd geom_histogram() layer that specifies a new aesthetic. To make it possible to see both sets of bars you'll need to make them transparent with the optional argument `alpha = 0.3`. Set the color for `AXIS1` to "red" and `AXIS2` to "black" using the `fill` argument. Label the x axis "Canopy Diameter (m)" and the y axis "Number of Acacia".

```
ggplot(acacia)+
  labs(x="Canopy Diameter (m)", y= "Number of Acacia")+
  geom_histogram(aes(x=AXIS1), alpha = 0.3, fill="red")+
  geom_histogram(aes(x=AXIS2), alpha = 0.3, fill="black")+
  theme_light()
```

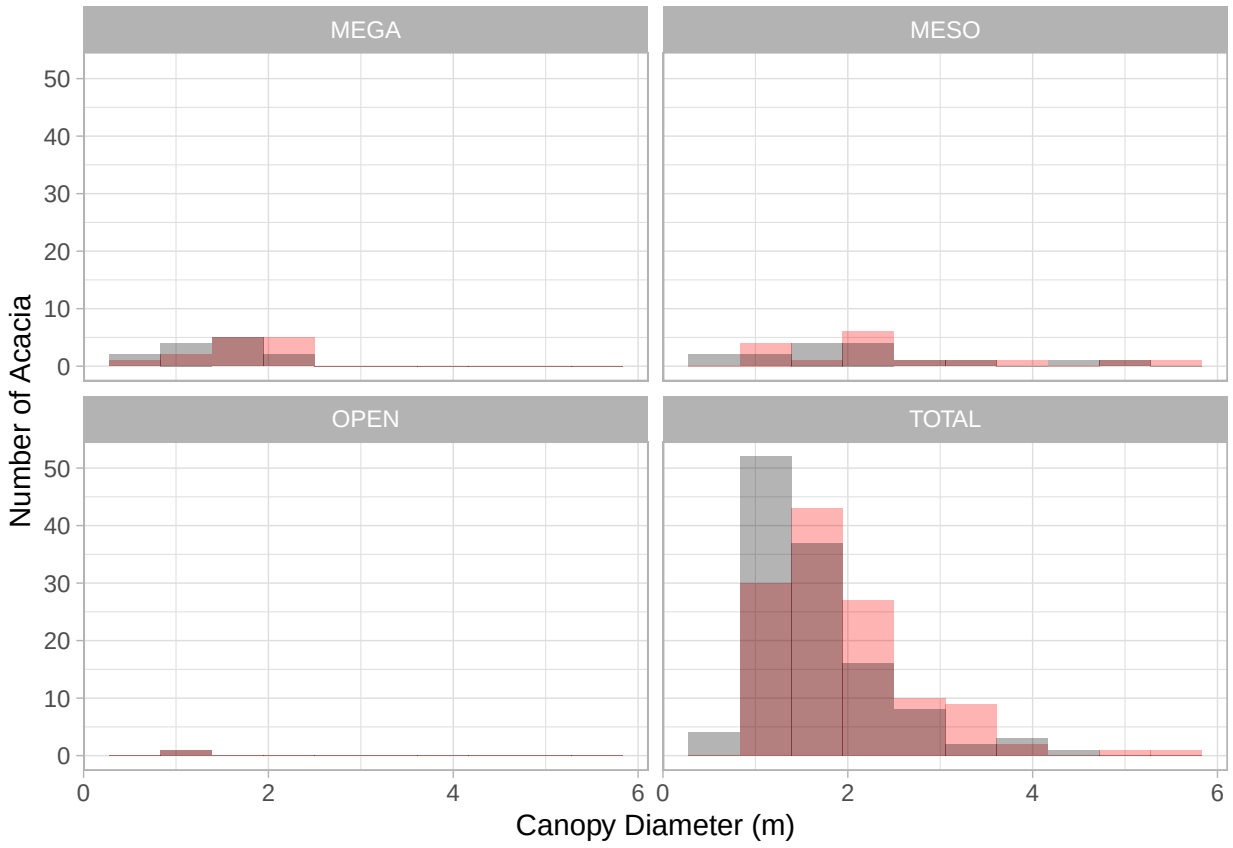
```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## Warning: Removed 4 rows containing non-finite values (`stat_bin()`).
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## Warning: Removed 4 rows containing non-finite values (`stat_bin()`).
```



d. Use `facet_wrap()` to make the same plot as (c) but with one subplot for each treatment. Set the number of bins in the histogram to 10.

```
ggplot(acacia)+
  labs(x="Canopy Diameter (m)", y= "Number of Acacia")+
  geom_histogram(bins =10, aes(x=AXIS1), alpha = 0.3, fill="red")+
  geom_histogram(bins = 10, aes(x=AXIS2), alpha = 0.3, fill="black")+
  theme_light()+
  facet_wrap(~TREATMENT)
```

```
## Warning: Removed 4 rows containing non-finite values (`stat_bin()`).
## Removed 4 rows containing non-finite values (`stat_bin()`).
```



4. Acacia and Ants Data Manipulation (20 pts)

Run the following line of code to use `read_tsv` from the `readr` package to read in the data from "TREE.txt". This line of code is using the `col_types` argument to specify the the `HEIGHT` and `AXIS_2` columns should have their data read is as the data class "double," which is like "numeric."

```
trees <- read_tsv("TREE.txt",
                  col_types = list(HEIGHT = col_double(),
                                   AXIS_2 = col_double()))
```

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

```
glimpse(trees)
```

```
## Rows: 7,508
## Columns: 16
## $ SURVEY       <dbl> 1, 2, 3, 4, 5, 1, 2, 3, 4, 5, 1, 2, 3, 4, 5, 1, 2, 3, 4, ~
## $ YEAR         <dbl> 2009, 2010, 2011, 2012, 2013, 2009, 2010, 2011, 2012, 201~
## $ SITE         <chr> "SOUTH", "SOUTH", "SOUTH", "SOUTH", "SOUTH", "SOUTH", "SO~
## $ TREATMENT    <chr> "TOTAL", "TOTAL", "TOTAL", "TOTAL", "TOTAL", "TOTAL", "TO~
## $ BLOCK        <dbl> 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, ~
## $ PLOT         <chr> "S2TOTAL", "S2TOTAL", "S2TOTAL", "S2TOTAL", "S2TOTAL", "S~
## $ SPECIES      <chr> "Acacia_etbaica", "Acacia_etbaica", "Acacia_etbaica", "Ac~
## $ ORIGINAL_TAG <dbl> 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 3, 3, 3, 3, 3, 4, 4, 4, 4, ~
## $ NEW_TAG      <dbl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, N~
```

```
## $ DEAD         <chr> "N", "N", "N", "N", "N", "N", "N", "N", "Y", "N", "N", "N~
## $ HEIGHT       <dbl> 3.40, 3.32, 3.65, 3.74, 3.59, 2.30, 2.32, 2.75, NA, 2.86,~
## $ AXIS_1       <dbl> 6.10, 8.25, 8.85, 5.50, 5.00, 2.20, 2.75, 3.30, NA, 3.70,~
## $ AXIS_2       <dbl> 5.00, 8.45, 9.00, 7.10, 8.15, 2.80, 2.65, 3.80, NA, 2.60,~
## $ CIRC         <dbl> 37.8, 18.8, 57.0, 60.0, 55.0, 14.2, 18.4, 25.0, NA, 31.0,~
## $ MEASUREMENT  <chr> "D", "D", "C", "C", "C", "D", "D", "C", NA, "D", "D", "D"~
## $ STEMS        <chr> "1", "1", "1", "1", "1", "1", "1", "1", NA, "1", "2", "3"~
```

Now that you have the `trees` data frame, do the following:

a. Update the `trees` data frame with a new column named `canopy_area` that contains the estimated canopy area calculated as the value in the `AXIS_1` column times the value in the `AXIS_2` column. Show output of the `trees` data frame with just the `SURVEY`, `YEAR`, `SITE`, and `canopy_area` columns.

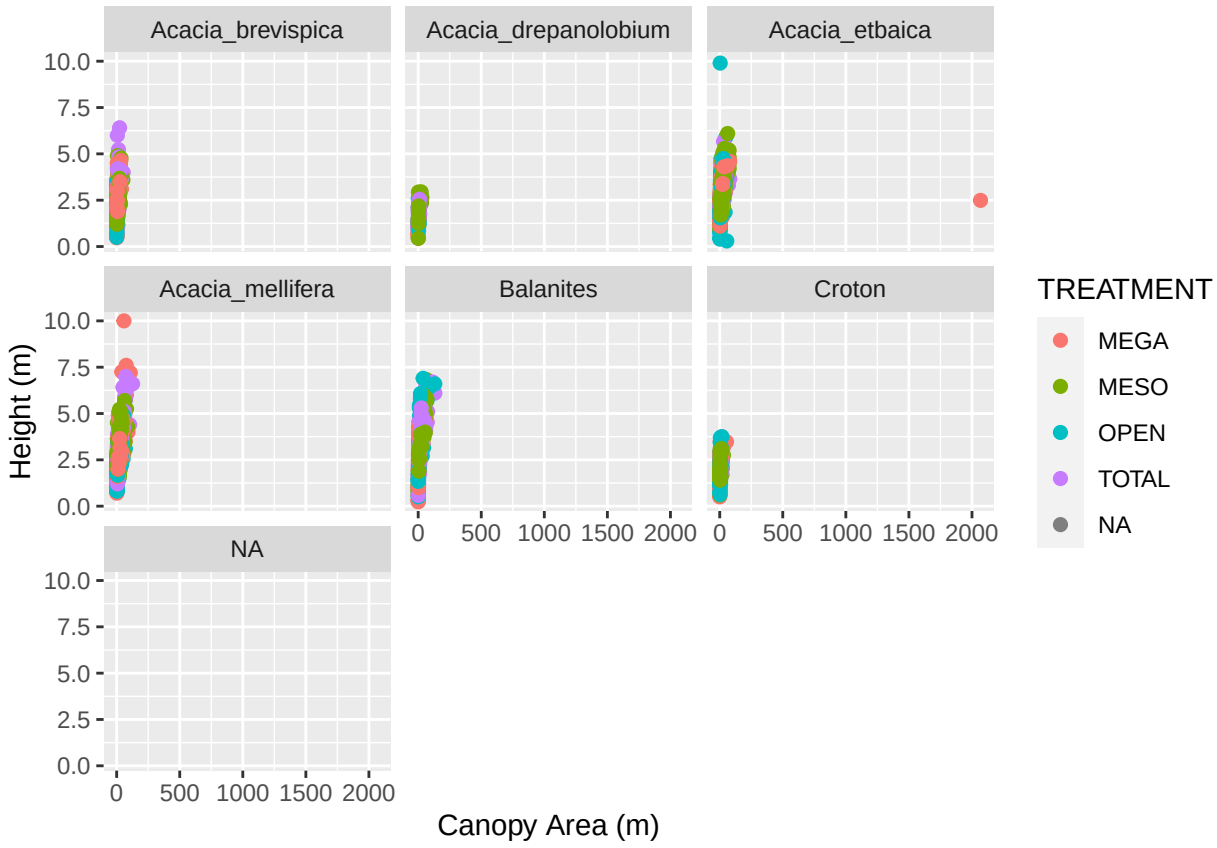
```
trees <- trees %>%
  mutate(canopy_area = AXIS_1*AXIS_2)

trees %>%
  select(SURVEY, YEAR, SITE, canopy_area, SPECIES)
```

```
## # A tibble: 7,508 x 5
##    SURVEY  YEAR SITE  canopy_area SPECIES
##     <dbl> <dbl> <chr>       <dbl> <chr>
##  1      1  2009 SOUTH       30.5  Acacia_etbaica
##  2      2  2010 SOUTH       69.7  Acacia_etbaica
##  3      3  2011 SOUTH       79.6  Acacia_etbaica
##  4      4  2012 SOUTH       39.0  Acacia_etbaica
##  5      5  2013 SOUTH       40.8  Acacia_etbaica
##  6      1  2009 SOUTH        6.16 Acacia_etbaica
##  7      2  2010 SOUTH        7.29 Acacia_etbaica
##  8      3  2011 SOUTH       12.5  Acacia_etbaica
##  9      4  2012 SOUTH       NA    Acacia_etbaica
## 10      5  2013 SOUTH        9.62 Acacia_etbaica
## # i 7,498 more rows
```

b. Make a scatter plot with `canopy_area` on the x axis and `HEIGHT` on the y axis. Color the points by `TREATMENT` and plot the points for each value in the `SPECIES` column in a separate subplot. Label the x axis "Canopy Area (m)" and the y axis "Height (m)". Make the point size 2.

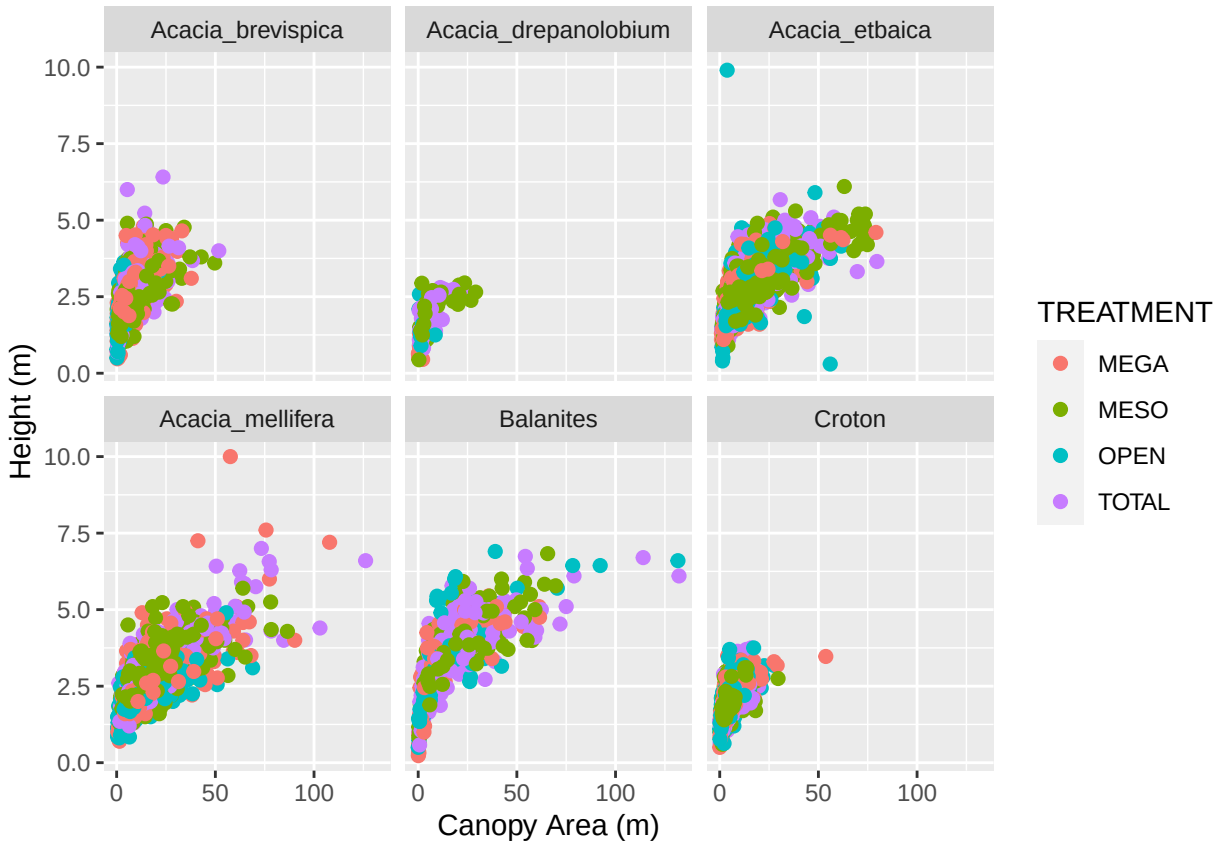
```
ggplot()+
  geom_point(trees, mapping=aes(x=canopy_area, y=HEIGHT, color=TREATMENT), size =2)+
  facet_wrap(~SPECIES)+
  labs(x="Canopy Area (m)", y="Height (m)")
```

```
## Warning: Removed 215 rows containing missing values (`geom_point()`).
```



c. That's a big outlier in the plot from (b). 50 by 50 meters is a little too big for a real Acacia, so filter the data to remove any values for `AXIS_1` and `AXIS_2` that are over 20 and update the data frame. Then remake the graph.

```
trees <- trees %>%
  filter(AXIS_1 < 20,
         AXIS_2 < 20)

trees%>%
  ggplot()+
  geom_point(trees, mapping=aes(x=canopy_area, y=HEIGHT, color=TREATMENT), size =2)+
  facet_wrap(~SPECIES)+
  labs(x="Canopy Area (m)", y="Height (m)")
```



d. Using the data without the outlier (i.e., the data generated in (c)), find out how the abundance of each species has been changing through time. Use `group_by`, `summarize`, and `n` to make a data frame with `YEAR`, `SPECIES`, and an `abundance` column that has the number of individuals in each species in each year. Print out this data frame.

```
question_4d <- trees%>%
  group_by(YEAR, SPECIES) %>%
  summarize(abundance = n())
```

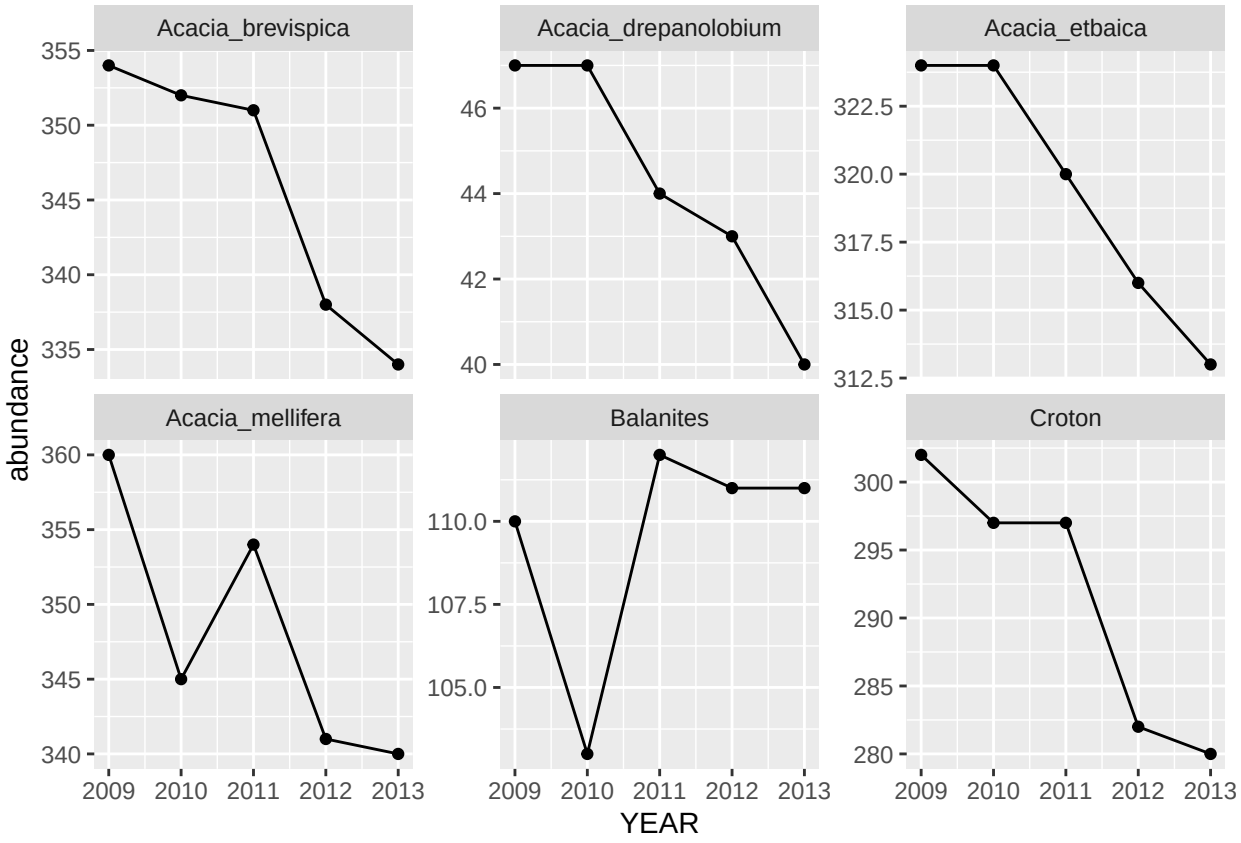
```
## `summarise()` has grouped output by 'YEAR'. You can override using the
## `.groups` argument.
```

```
question_4d
```

```
## # A tibble: 30 x 3
## # Groups:   YEAR [5]
##     YEAR SPECIES               abundance
##    <dbl> <chr>                     <int>
##  1  2009 Acacia_brevispica           354
##  2  2009 Acacia_drepanolobium         47
##  3  2009 Acacia_etbaica              324
##  4  2009 Acacia_mellifera            360
##  5  2009 Balanites                   110
##  6  2009 Croton                      302
##  7  2010 Acacia_brevispica           352
##  8  2010 Acacia_drepanolobium         47
##  9  2010 Acacia_etbaica              324
## 10  2010 Acacia_mellifera            345
## # i 20 more rows
```

e. Using the data frame generated in (d), make a line plot with points (by using `geom_line` in addition to `geom_point`) with `YEAR` on the x axis and `abundance` on the y axis with one subplot per species. To let you seen each trend clearly let the scale for the y axis vary among plots by adding `scales = "free_y"` as an optional argument to `facet_wrap`.

```
question_4d %>%
  ggplot()+
  geom_line(aes(x=YEAR, y=abundance))+
  geom_point(aes(x=YEAR, y=abundance))+
  facet_wrap(~SPECIES, scales= "free_y")
```



5. Adult vs. Newborn Size (20 pts)

Larger organisms have larger offspring. We want to explore the form of this relationship in mammals. First, read in the data frame. This code uses a handy function called `rename` from the `tidyverse` to rename columns.

```
# download the file
download.file("https://esapubs.org/archive/ecol/E084/093/Mammal_lifehistories_v2.txt",
              "Mammal_lifehistories_v2.txt")
# read the file
mammal_histories <- read_tsv("Mammal_lifehistories_v2.txt",
                             na = c("-999", "-999.00")) %>%
  # rename columns names for easier use
  rename(mass_g = `mass(g)`,
         gestation_mo = `gestation(mo)`,
         newborn_g = `newborn(g)`,
         weaning_mo = `weaning(mo)`,
```

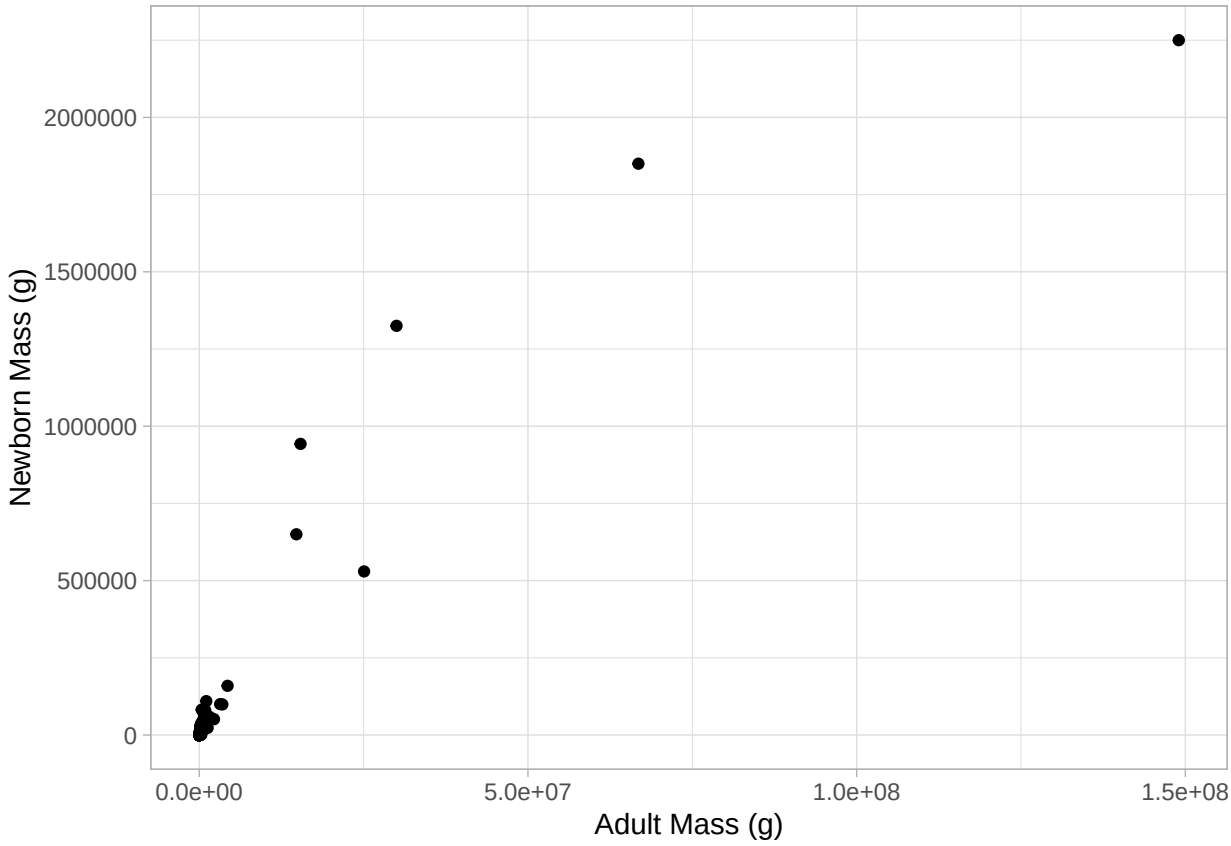
```
        wean_mass_g = `wean mass(g)`,
        AFR_mo = `AFR(mo)`,
        max_life_mo = `max. life(mo)`,
        litter_size = `litter size`,
        litters_per_year = `litters/year`)
```

```
## Rows: 1440 Columns: 14
## -- Column specification --------------------------------------------------------
## Delimiter: "\t"
## chr (4): order, family, Genus, species
## dbl (9): mass(g), gestation(mo), newborn(g), weaning(mo), wean mass(g), AFR(...
## num (1): refs
##
## 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. Graph adult mass vs. newborn mass. Label the axes with clearer labels than the column names.

```
ggplot(mammal_histories)+
  geom_point(aes(x=mass_g, y=newborn_g))+
  labs(x="Adult Mass (g)", y="Newborn Mass (g)")+
  theme_light()
```

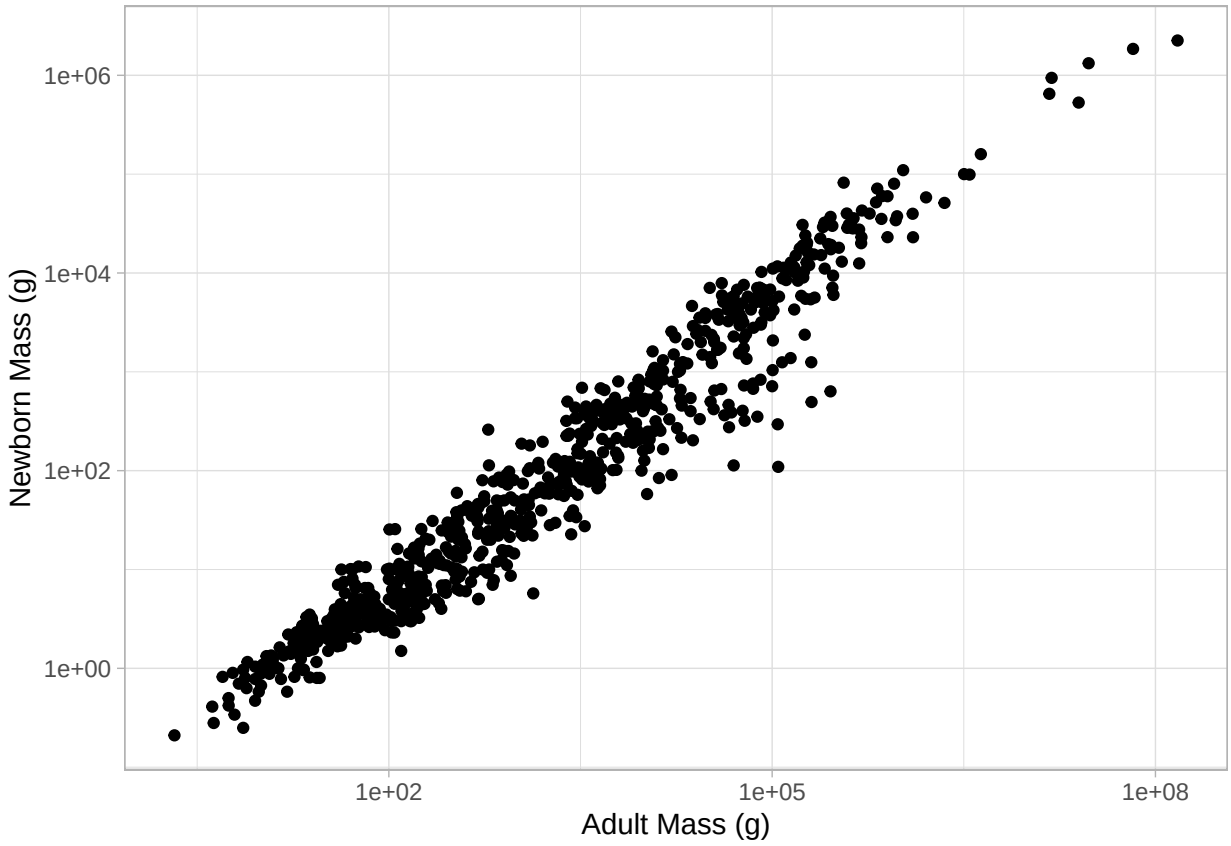
```
## Warning: Removed 624 rows containing missing values (`geom_point()`).
```



b. It looks like there's a regular pattern here, but it's definitely not linear. Let's see if log-transformation straightens it out. Graph adult mass vs. newborn mass, with both axes scaled logarithmically. Label the axes.

```
ggplot(mammal_histories)+
  geom_point(aes(x=mass_g, y=newborn_g))+
  labs(x="Adult Mass (g)", y="Newborn Mass (g)")+
  theme_light()+
  scale_x_log10()+
  scale_y_log10()
```

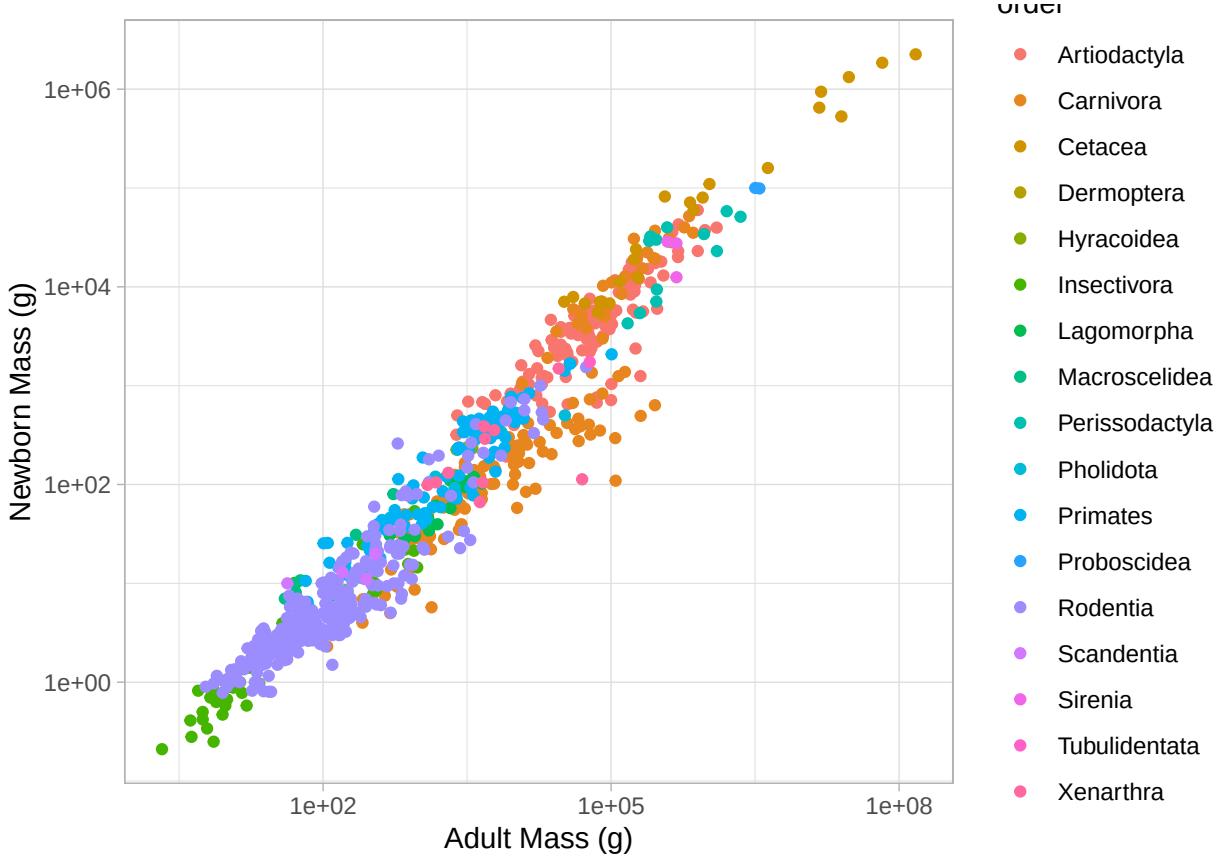
```
## Warning: Removed 624 rows containing missing values (`geom_point()`).
```



c. This looks like a pretty regular pattern, so you wonder if it varies among different groups. Graph adult mass vs. newborn mass, with both axes scaled logarithmically, and the data points colored by order. Label the axes.

```
ggplot(mammal_histories)+
  geom_point(aes(x=mass_g, y=newborn_g, color=order))+
  labs(x="Adult Mass (g)", y="Newborn Mass (g)")+
  theme_light()+
  scale_x_log10()+
  scale_y_log10()
```

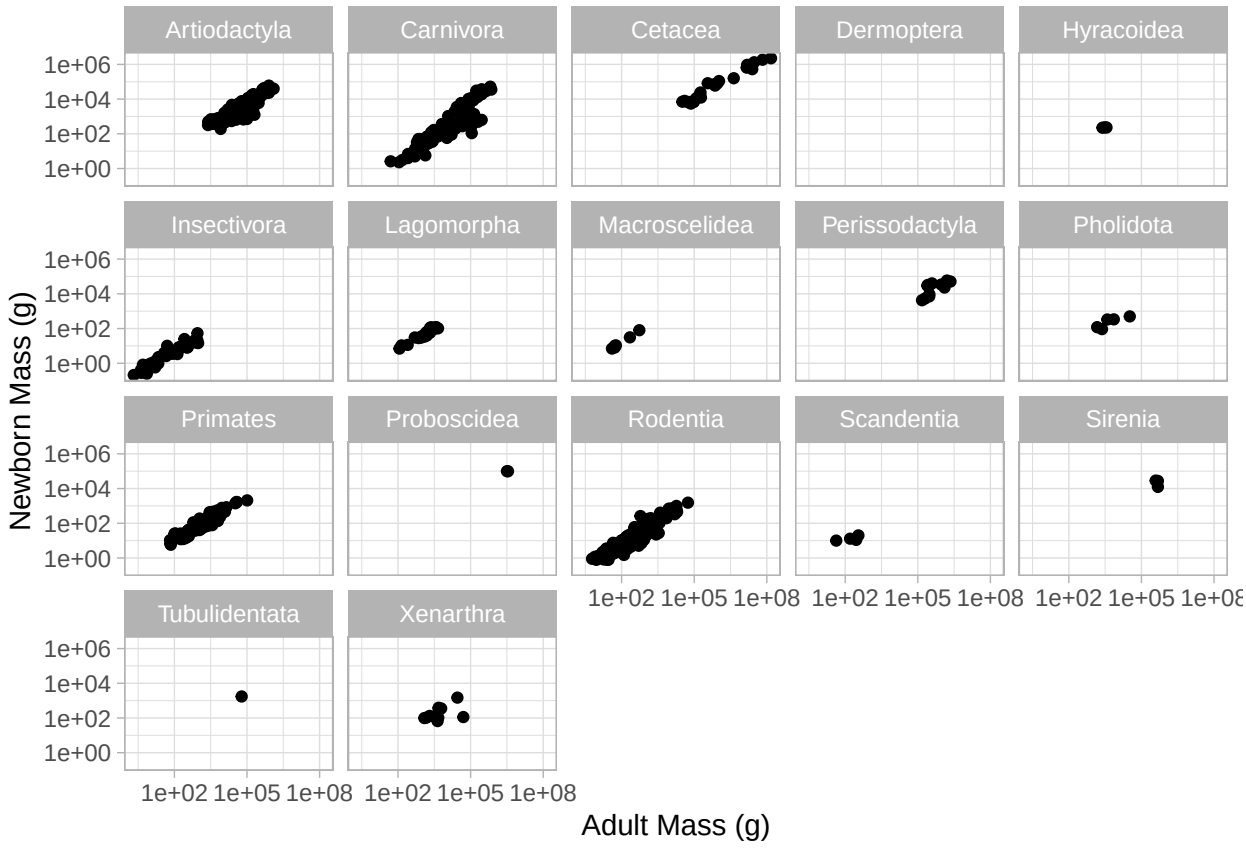
```
## Warning: Removed 624 rows containing missing values (`geom_point()`).
```



d. Coloring the points was useful, but there are a lot of points and it's kind of hard to see what's going on with all of the orders. Use `facet_wrap` to create a subplot for each order.

```
ggplot(mammal_histories)+
  geom_point(aes(x=mass_g, y=newborn_g))+
  labs(x="Adult Mass (g)", y="Newborn Mass (g)")+
  theme_light()+
  scale_x_log10()+
  scale_y_log10()+
  facet_wrap(~order)
```

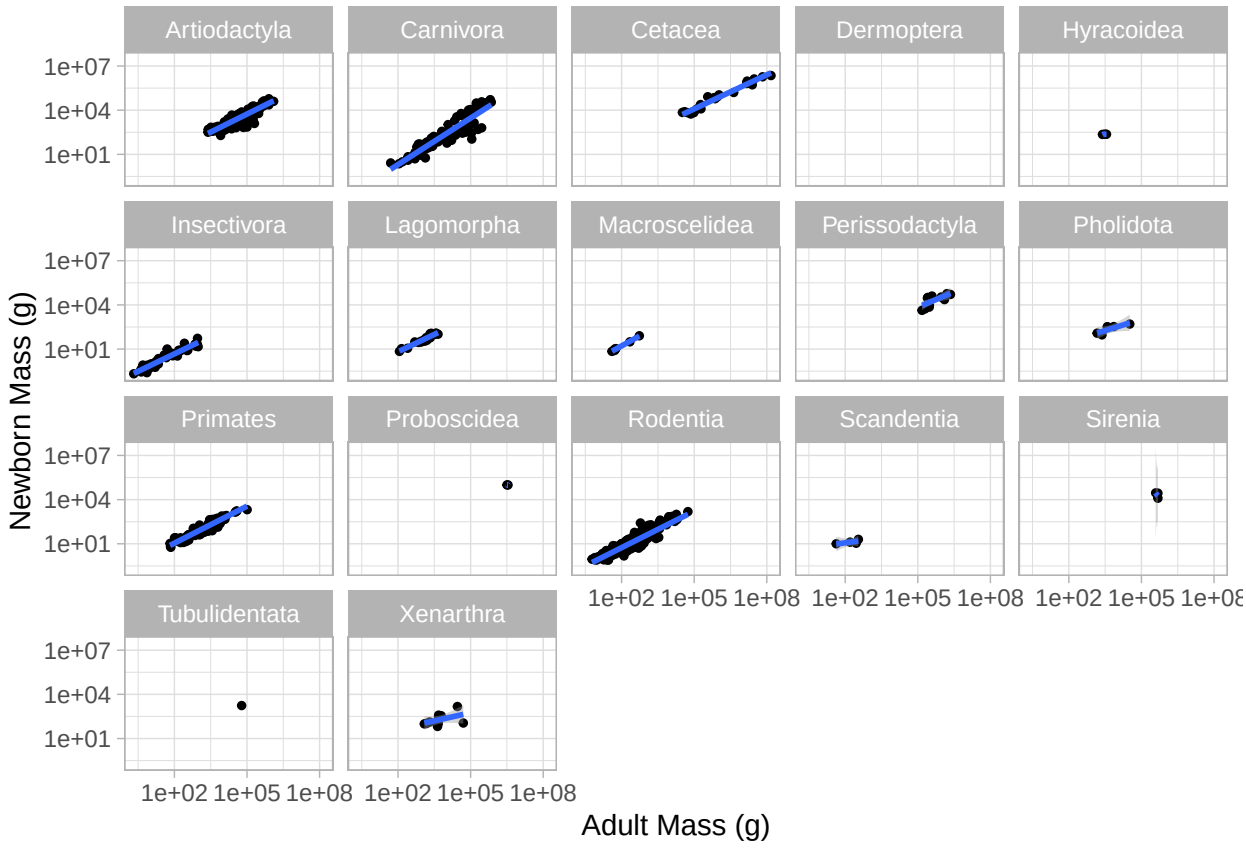
```
## Warning: Removed 624 rows containing missing values (`geom_point()`).
```



e. Now let's visualize the relationships between the variables using a simple linear model. Create a new graph like your faceted plot, but using `geom_smooth` to fit a linear model to each order. You can do this using the optional argument `method = "lm"` in `geom_smooth`.

```
ggplot(mammal_histories)+
  geom_point(aes(x=mass_g, y=newborn_g), size=1)+
  labs(x="Adult Mass (g)", y="Newborn Mass (g)")+
  theme_light()+
  scale_x_log10()+
  scale_y_log10()+
  facet_wrap(~order)+
  geom_smooth(aes(x=mass_g, y=newborn_g),method=lm)
```

```
## `geom_smooth()` using formula = 'y ~ x'
## Warning: Removed 624 rows containing non-finite values (`stat_smooth()`).
## Warning in qt((1 - level)/2, df): NaNs produced
## Warning: Removed 624 rows containing missing values (`geom_point()`).
## Warning in max(ids, na.rm = TRUE): no non-missing arguments to max; returning
## -Inf
```



6. Reflection (25 pts)

Reflections are graded for completion only.

Write about 5 sentences addressing *at least* one of the following questions.

- What has worked well for you in this course for you so far?
- What has been particularly challenging for you so far?
- Is there anything that I can do to help your learning in the course?
- Other reflections about the content of the module that you would like to share.

Answer: This course has been great so far, and I've found that I am learning a lot more about the mechanics of each function we're using than I knew previously. The in-class lectures with built in time to work on the assignment are also really nice and make it easy to ask questions and get answers versus using something like Slack to communicate those while working on the assignment at home. There haven't been any big challenges in the course so far, besides time management to watch the lectures and keep up with the module when I miss class for things. Along those lines, I really appreciate the recorded lectures, they're a great resource for the days when I have out of town events for band and can't be in class in person. Thanks Dr. Bledsoe!