# Week 5 Assignment

Ellen Bledsoe

2025-02-18

## 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   3.5.1     v tibble    3.2.1
## v lubridate 1.9.4     v tidyr     1.3.1
## v purrr     1.0.4
## -- 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".

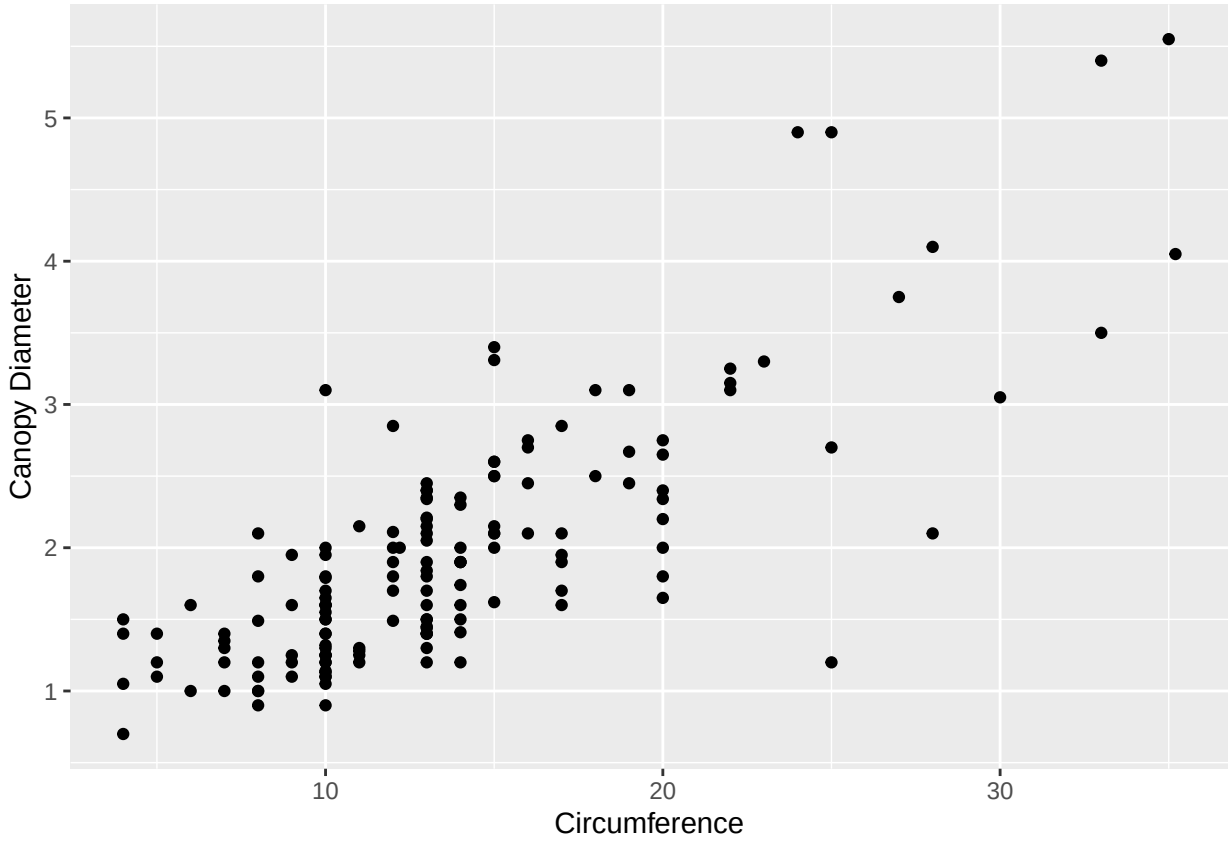
b. The same plot as (a), but with both axes scaled logarithmically (using `scale_x_log10` and `scale_y_log10`).
c. The same plot as (a), but with points colored based on the `ANT` column (the species of ant symbiont living with the acacia)
d. The same plot as (c), but instead of different colors show different species of ant (values of `ANT`) each in a separate subplot.
e. The same plot as (d) but add a simple model of the data by adding `geom_smooth`.

```
# 1a
print("1a")
```

```
## [1] "1a"
```

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

```
## Warning: Removed 4 rows containing missing values or values outside the scale range
## (`geom_point()`).
```

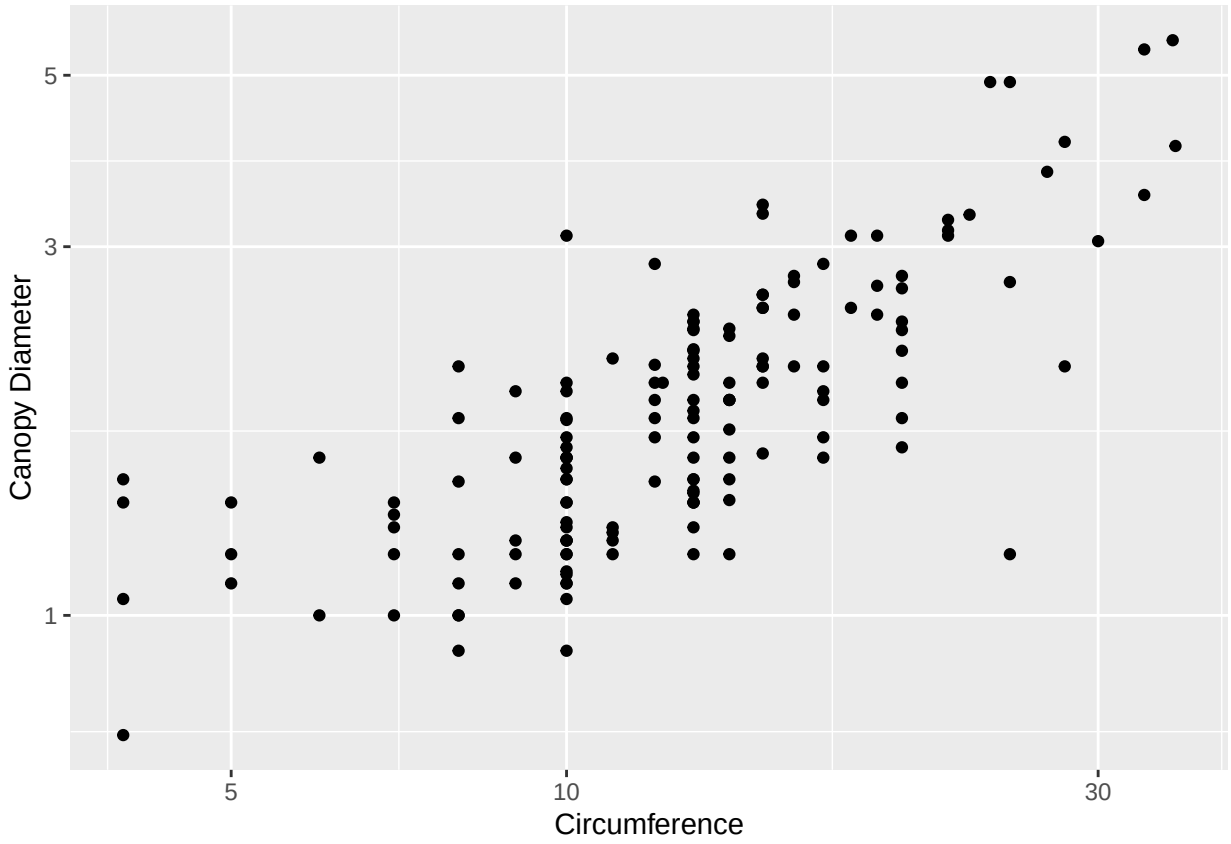



```
#ggsave("Graphing-acacia-ants-R-1.jpeg")

# 1b
print("1b")
```

```
## [1] "1b"
```

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

```
## Warning: Removed 4 rows containing missing values or values outside the scale range
## (`geom_point()`).
```

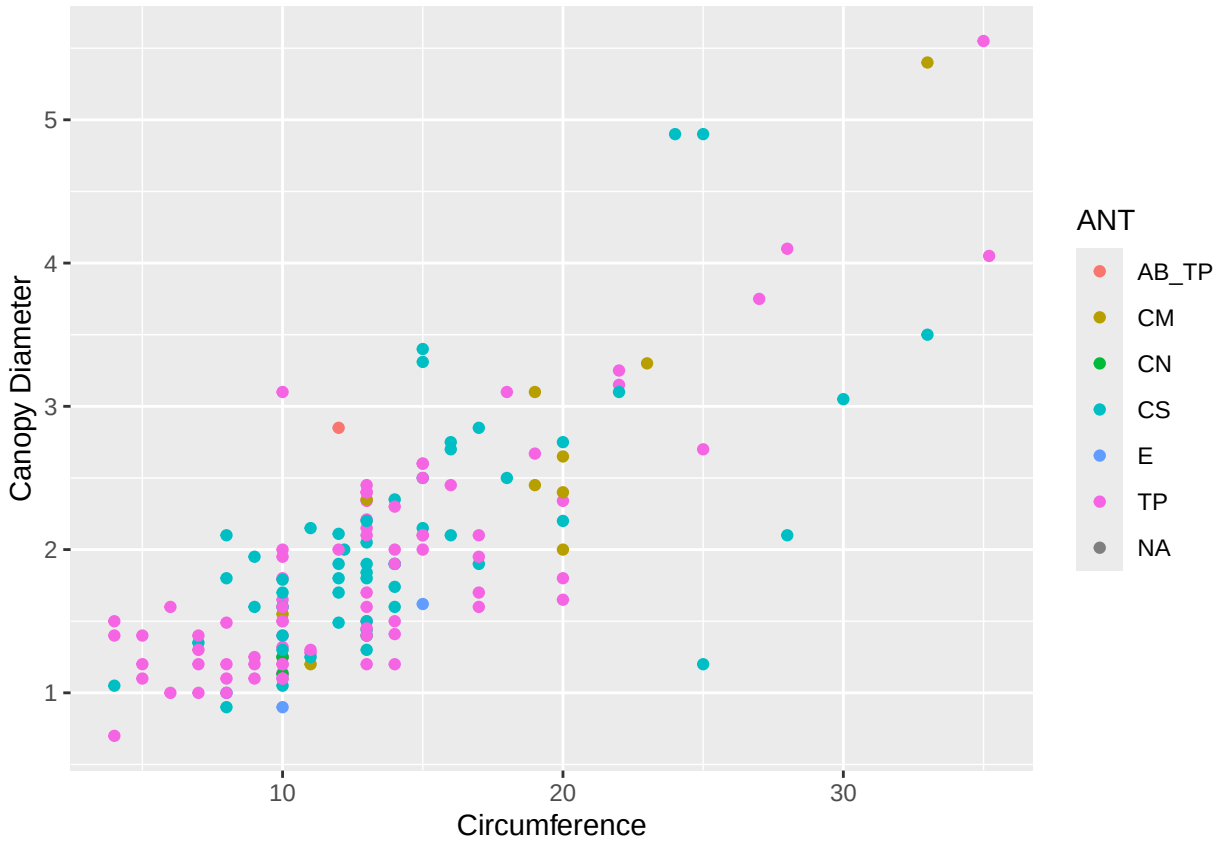



```
#ggsave("Graphing-acacia-ants-R-2.jpeg")

# 1c
print("1c")
```

```
## [1] "1c"
```

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

```
## Warning: Removed 4 rows containing missing values or values outside the scale range
## (`geom_point()`).
```

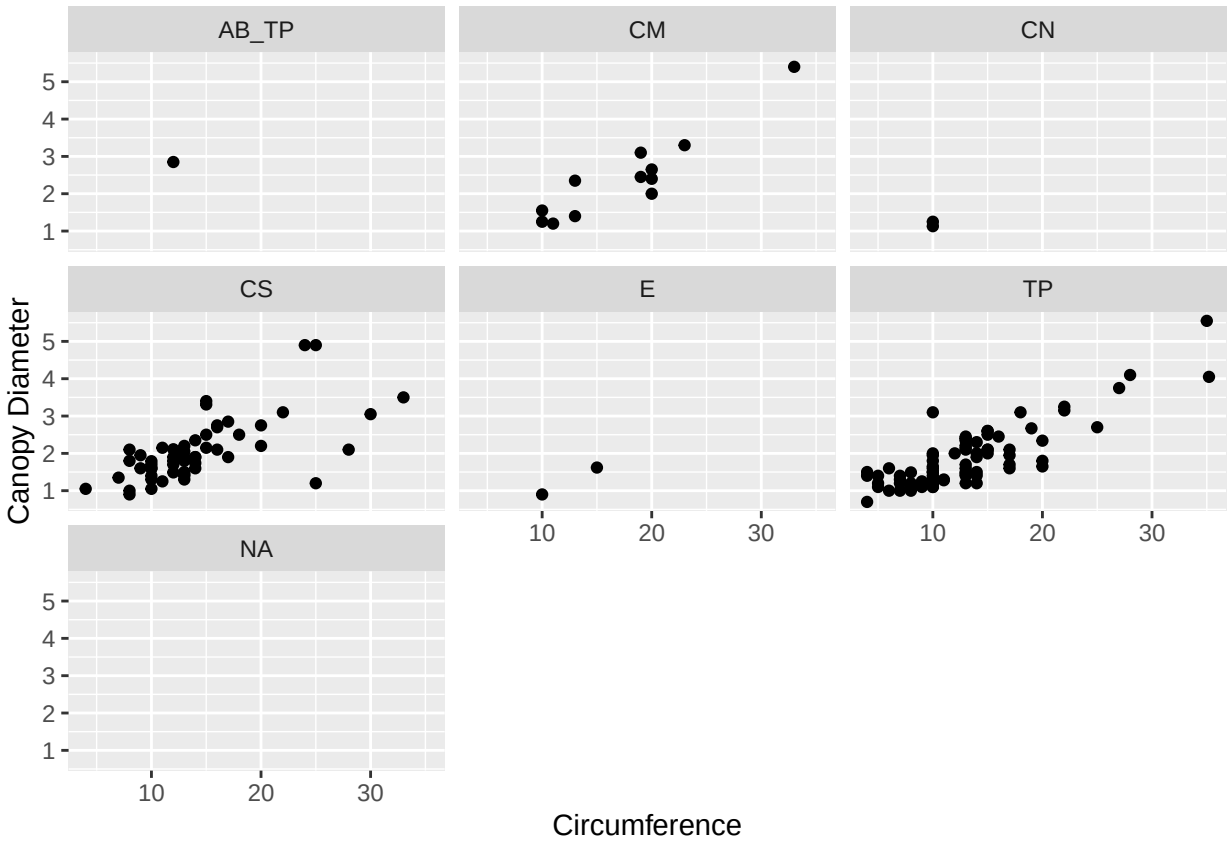



```
#ggsave("Graphing-acacia-ants-R-3.jpeg")

# 1d
print("1d")
```

```
## [1] "1d"
```

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

```
## Warning: Removed 4 rows containing missing values or values outside the scale range
## (`geom_point()`).
```



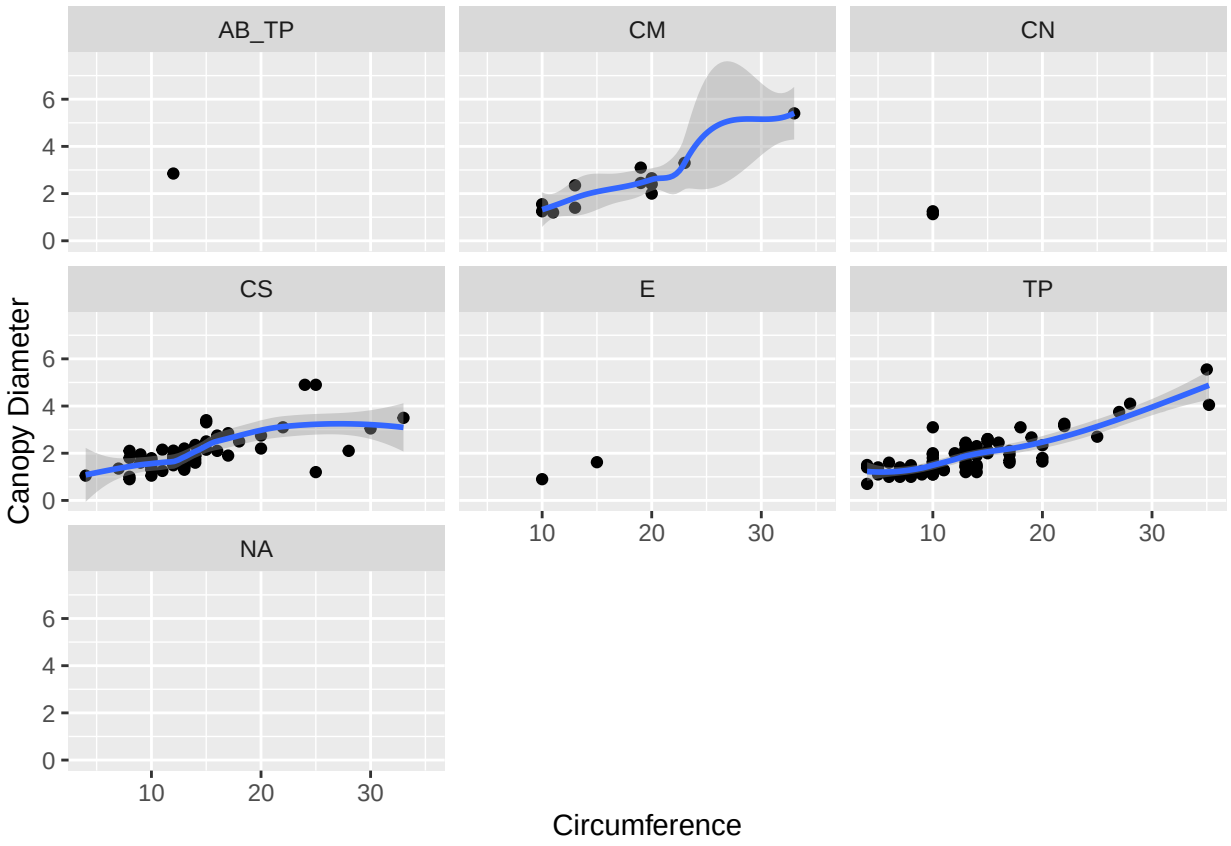

```
#ggsave("Graphing-acacia-ants-R-4.jpeg")

# 1e
print("1e")
```

```
## [1] "1e"
```

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

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

## Warning: Removed 4 rows containing non-finite outside the scale range
## (`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: Failed to fit group -1.
## Caused by error in `predLoess()`:
## ! NA/NaN/Inf in foreign function call (arg 5)

## Warning: Removed 4 rows containing missing values or values outside the scale range
## (`geom_point()`).
```




```
#ggsave("Graphing-acacia-ants-R-5.jpeg")
```

**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.

```
# create a data frame with 3 columns: body_mass, metabolic_rate, and family
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"))
```

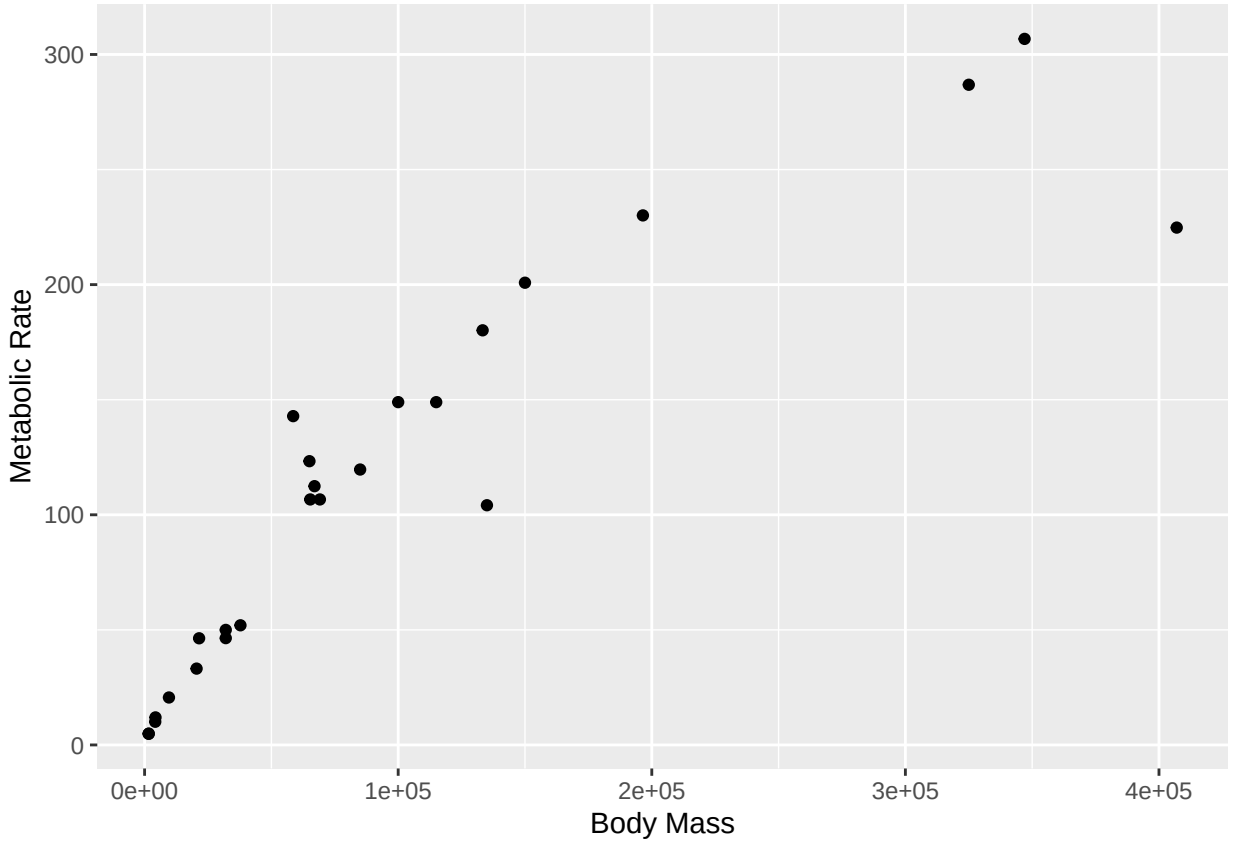
Make the following plots with appropriate axis labels:

a. A plot of body mass vs. metabolic rate
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.
c. The same plot as (b), but with the different families indicated using color.
d. The same plot as (b), but with the different families each in their own subplot.

```
# 2a.  A graph of body mass vs. metabolic rate
print("2a")
```

```
## [1] "2a"
```

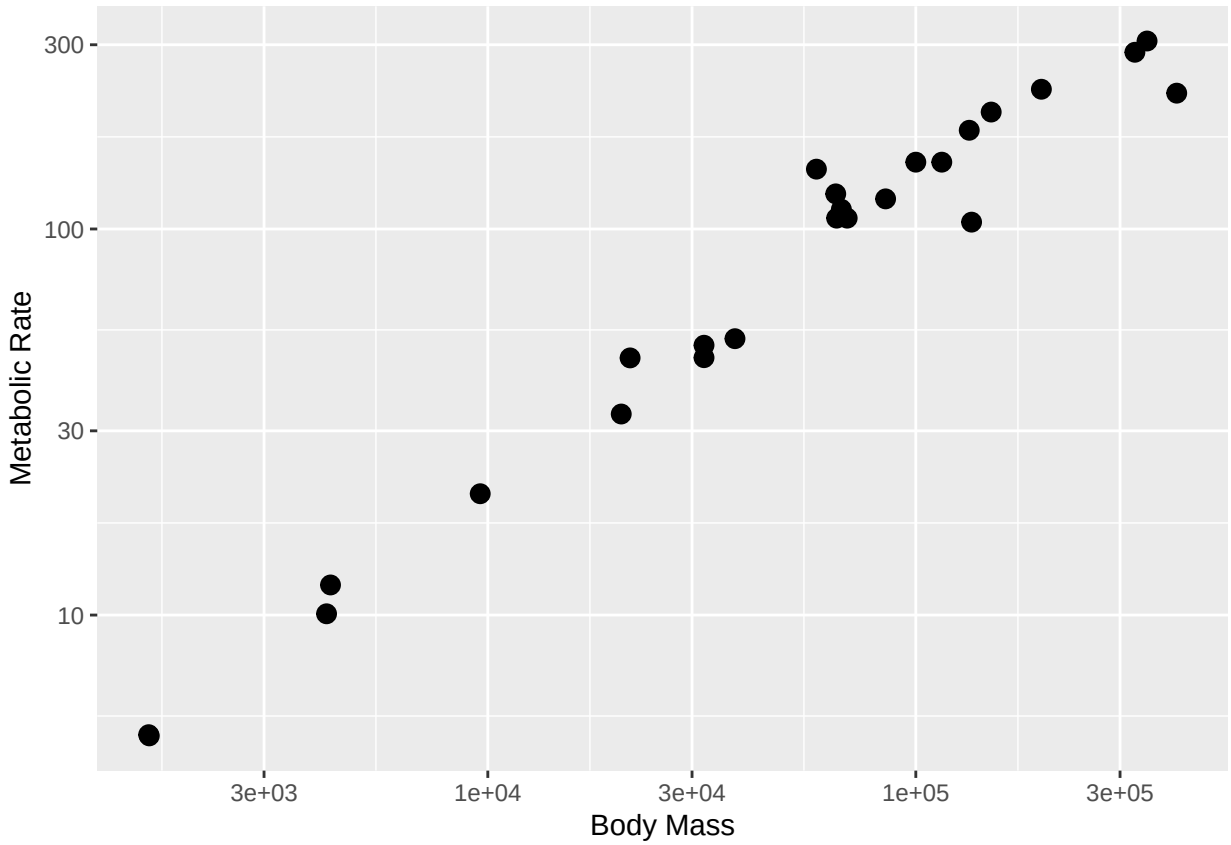
```
ggplot(size_mr_data, aes(body_mass, metabolic_rate)) + geom_point() +
  labs(x = "Body Mass", y = "Metabolic Rate")
```




```
# 2b.  A graph of body mass vs. metabolic rate, log scaled, with pt size 5.
print("2b")
```

```
## [1] "2b"
```

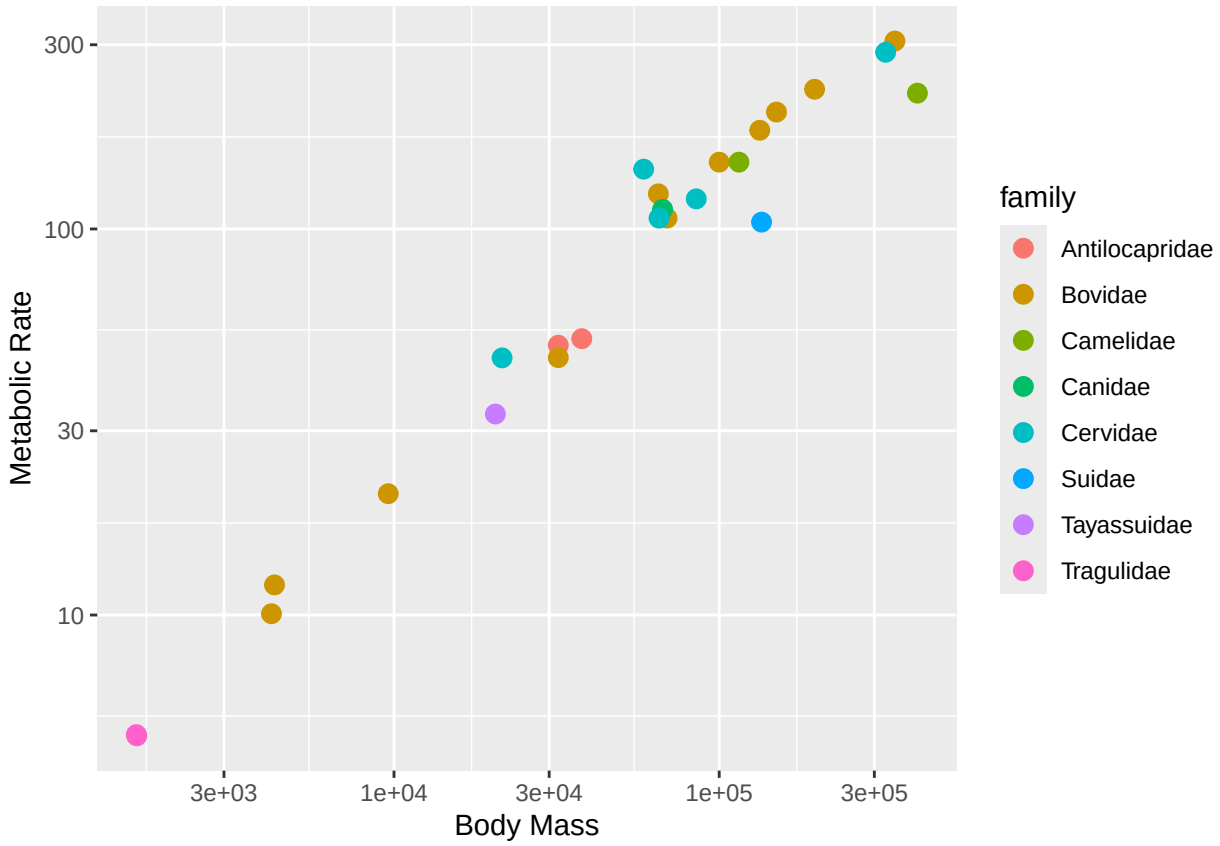
```
ggplot(data = size_mr_data, mapping = aes(x = body_mass, y = metabolic_rate)) +
  geom_point(size=3) +
  labs(x = "Body Mass", y = "Metabolic Rate") +
  scale_x_log10() + scale_y_log10()
```




```
# c. The same plot as (2), but with the different families indicated using color.
print("2c")
```

```
## [1] "2c"
```

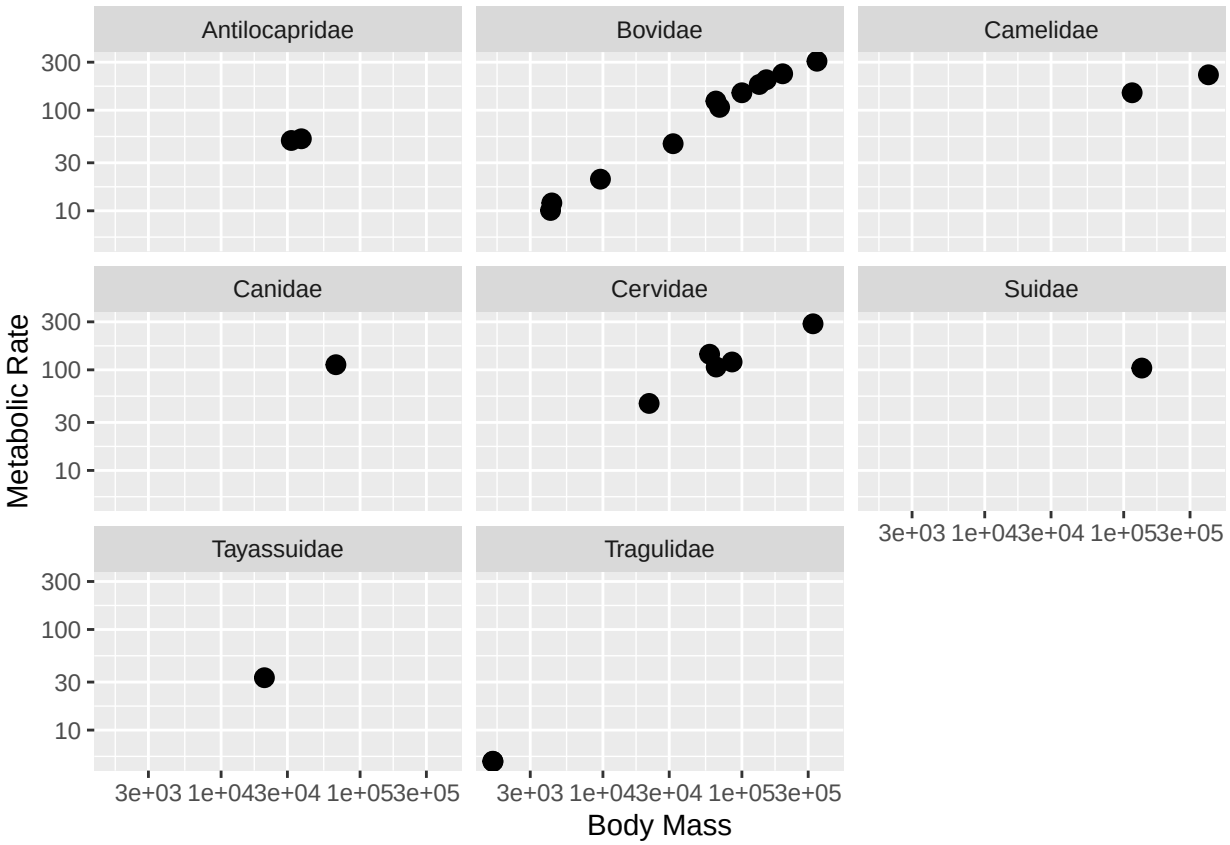
```
ggplot(size_mr_data, aes(x = body_mass, y = metabolic_rate, color = family)) +
  geom_point(size = 3) +
  scale_x_log10() +
  scale_y_log10() +
  labs(x = "Body Mass", y = "Metabolic Rate")
```




```
# d. The same plot as (2), but with the different families each in their own subplot.
print("2d")
```

```
## [1] "2d"
```

```
ggplot(size_mr_data, 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")
```




### 3. Acacia and Ants Histograms (20 pts)

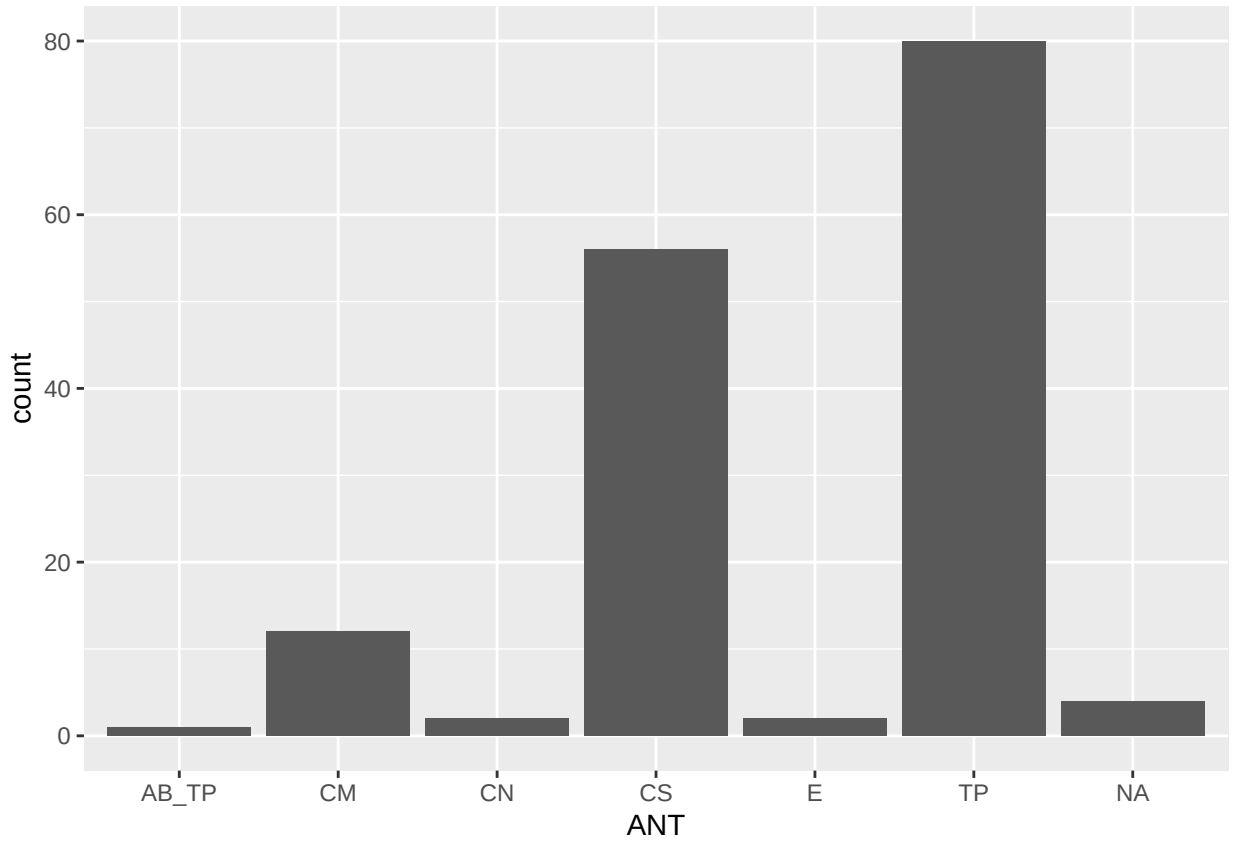
In this exercise, we will be making a number of different histograms with the `acacia` dataset.

a. Make a bar plot of the number of acacia with each mutualist ant species (using the `ANT` column).
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".
c. Make a plot that shows histograms of both `AXIS1` and `AXIS2`. Due to the way the data are 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".
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.

```
# 3a
print("3a")
```

```
## [1] "3a"
```

```
ggplot(data = acacia, mapping = aes(x = ANT)) +
  geom_bar()
```



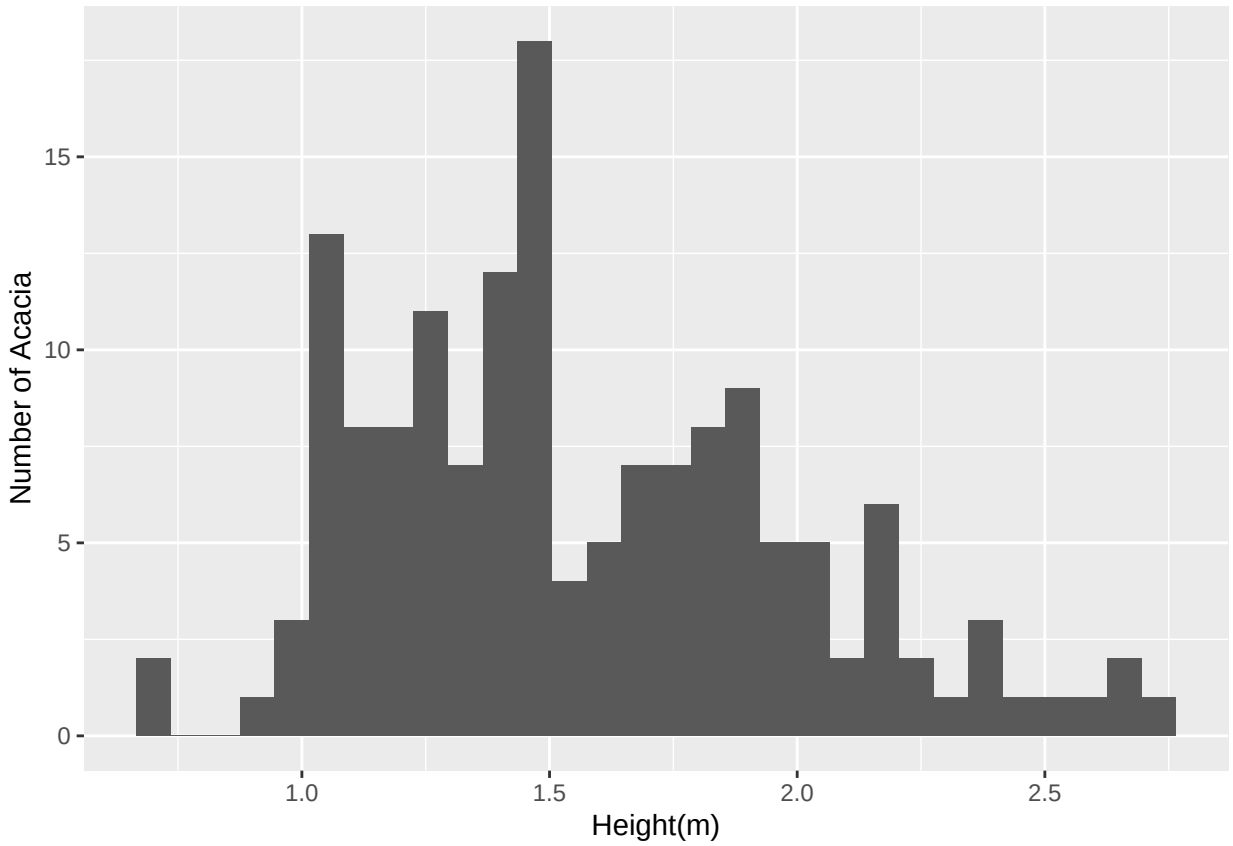

```
#ggsave("Graphing-acacia-ants-histograms-R-1.jpeg")

# 3b
print("3b")
```

```
## [1] "3b"
```

```
ggplot(data = acacia, mapping = aes(x = HEIGHT)) +
  geom_histogram() +
  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 outside the scale range
## (`stat_bin()`).
```

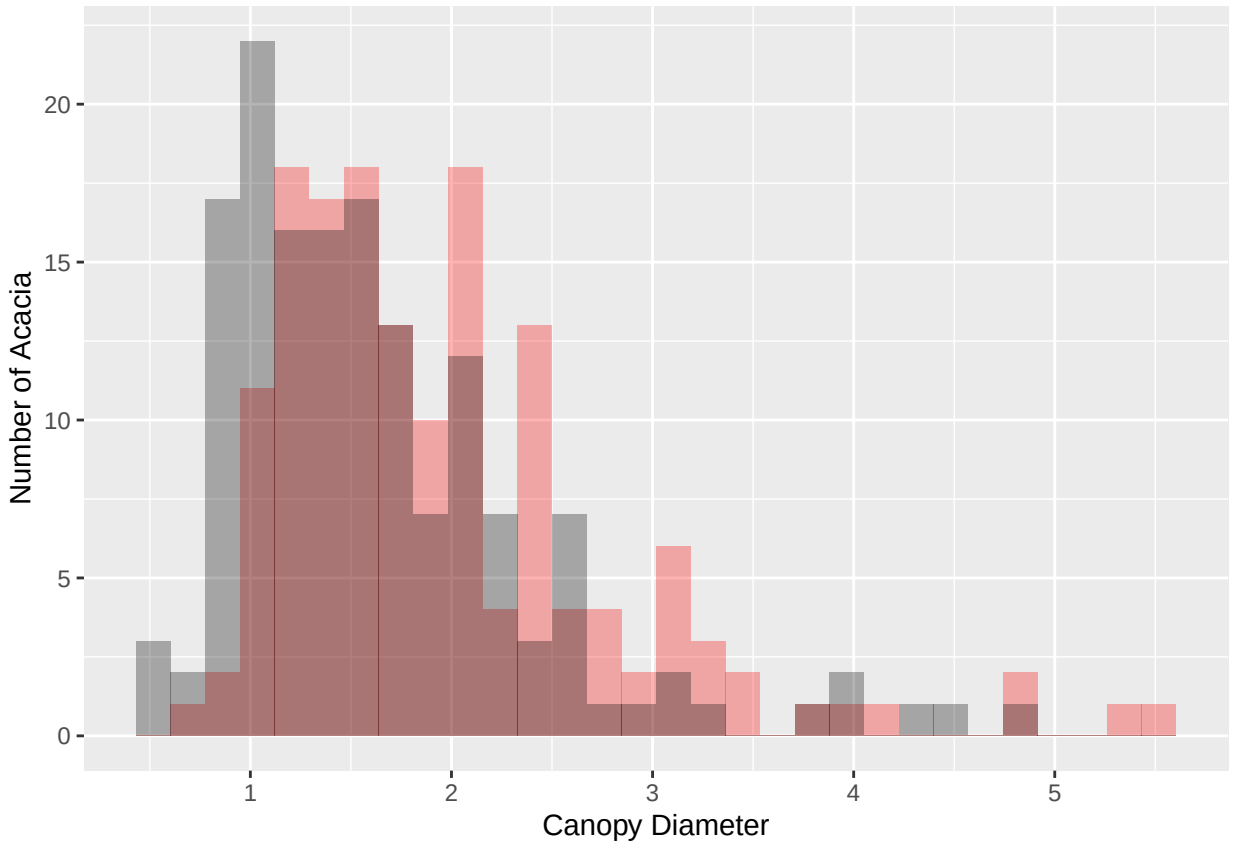



```
#ggsave("Graphing-acacia-ants-histograms-R-2.jpeg")

# 3c
print("3c")
```

```
## [1] "3c"
```

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

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

## Warning: Removed 4 rows containing non-finite outside the scale range
## (`stat_bin()`).

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

## Warning: Removed 4 rows containing non-finite outside the scale range
## (`stat_bin()`).
```

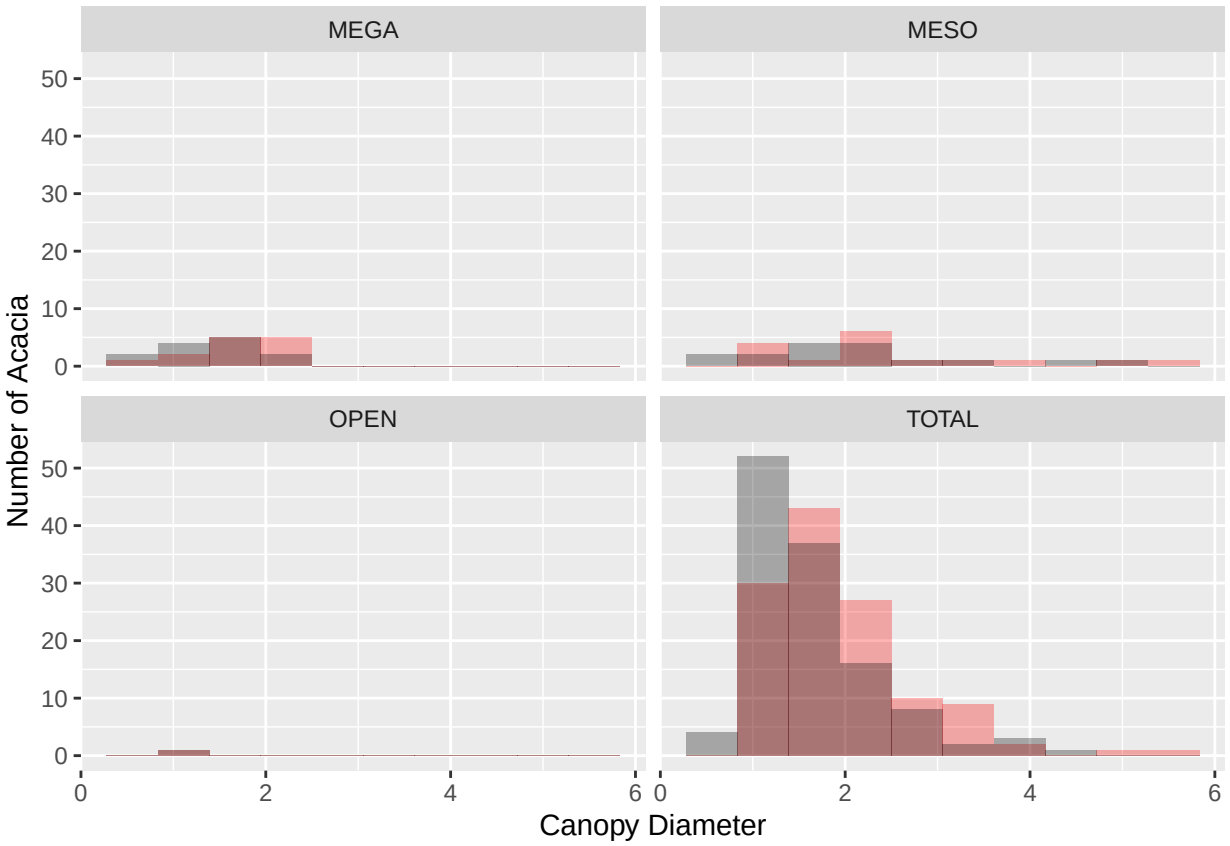



```
#ggsave("Graphing-acacia-ants-histograms-R-3.jpeg")

# 3d
print("3d")
```

```
## [1] "3d"
```

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

```
## Warning: Removed 4 rows containing non-finite outside the scale range (‘stat_bin()‘).
## Removed 4 rows containing non-finite outside the scale range (‘stat_bin()‘).
```




```
#ggsave("Graphing-acacia-ants-histograms-R-4.jpeg")
```

### 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."

You'll see a warning when you read in the data. You can ignore it.

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

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

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.

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.
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.
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.
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`.

```
# 4a
print("4a")
```

```
## [1] "4a"
```

```
trees <- mutate(trees, canopy_area = AXIS_1 * AXIS_2)
select(trees, SURVEY, YEAR, SITE, SPECIES, canopy_area)
```

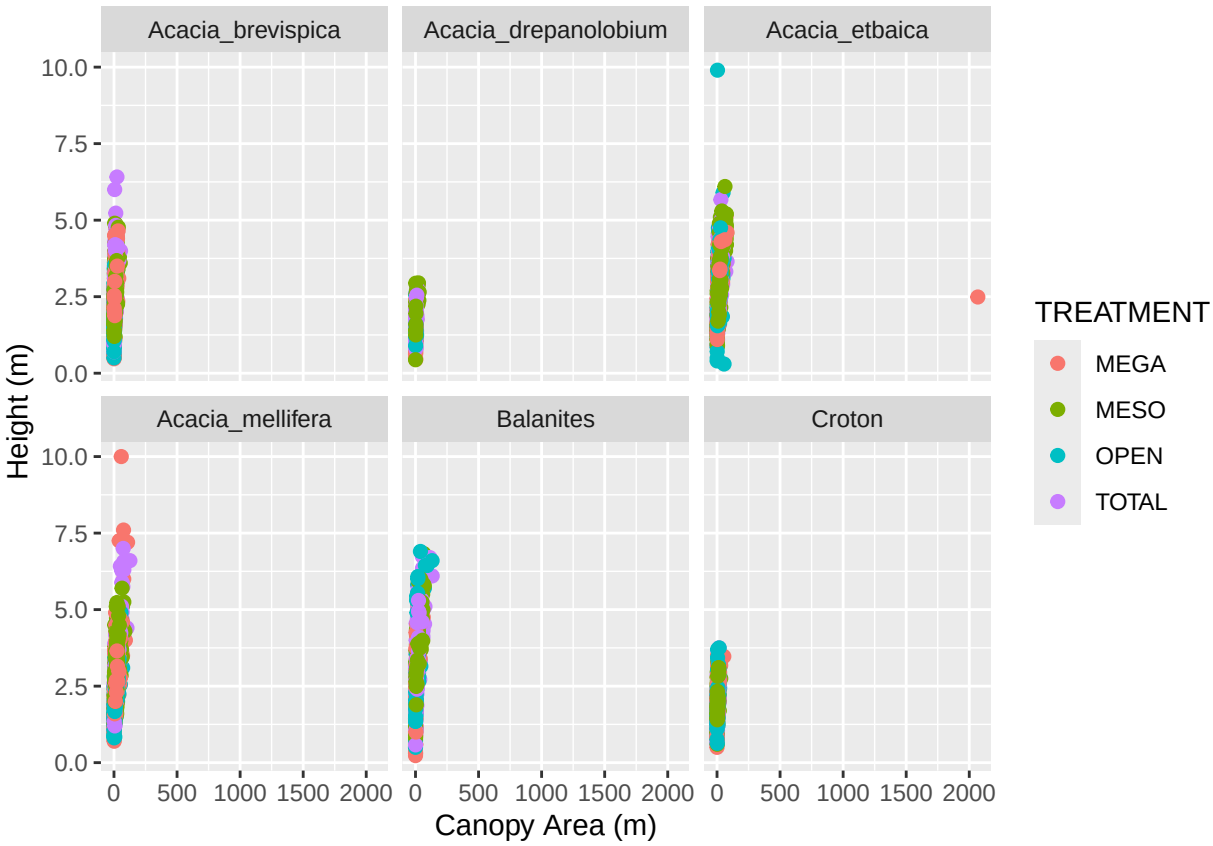
```
## # A tibble: 7,501 x 5
##    SURVEY  YEAR SITE  SPECIES        canopy_area
##     <dbl> <dbl> <chr> <chr>                <dbl>
##  1      1  2009 SOUTH Acacia_etbaica       30.5
##  2      2  2010 SOUTH Acacia_etbaica       69.7
##  3      3  2011 SOUTH Acacia_etbaica       79.6
##  4      4  2012 SOUTH Acacia_etbaica       39.0
##  5      5  2013 SOUTH Acacia_etbaica       40.8
##  6      1  2009 SOUTH Acacia_etbaica        6.16
##  7      2  2010 SOUTH Acacia_etbaica        7.29
##  8      3  2011 SOUTH Acacia_etbaica       12.5
##  9      4  2012 SOUTH Acacia_etbaica       NA
## 10      5  2013 SOUTH Acacia_etbaica        9.62
## # i 7,491 more rows
```

```
# 4b
print("4b")
```

```
## [1] "4b"
```

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

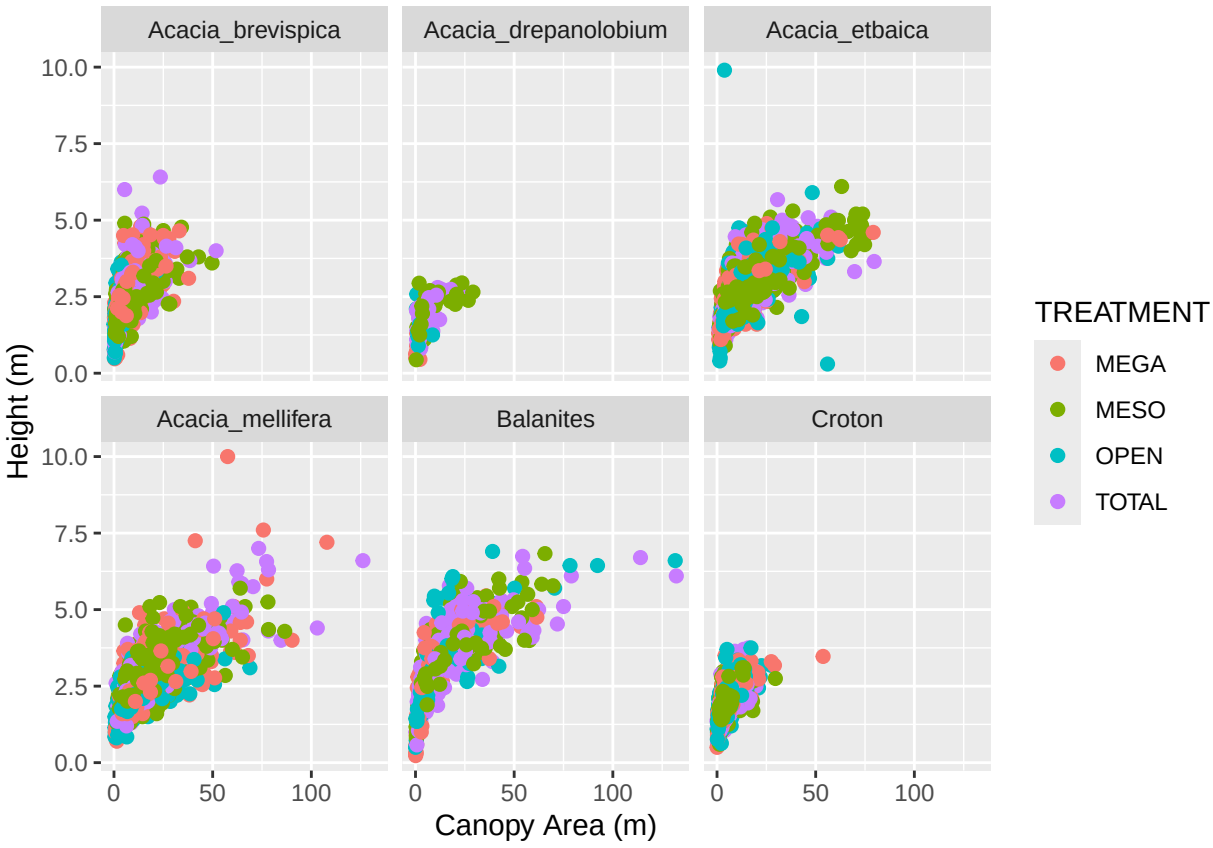
```
## Warning: Removed 208 rows containing missing values or values outside the scale range
## (`geom_point()`).
```




```
#ggsave("Graphing-acacia-ants-data-manip-R-2.jpeg")

# 4c
print("4c")
```

```
## [1] "4c"
```

```
trees <- filter(trees, AXIS_1 < 20, AXIS_2 < 20)
ggplot(data = trees, mapping = aes(x = canopy_area, y = HEIGHT, color = TREATMENT)) +
  geom_point(size = 2) +
  labs(x = "Canopy Area (m)", y = "Height (m)") +
  facet_wrap(~SPECIES)
```




```
#ggsave("Graphing-acacia-ants-data-manip-R-3.jpeg")

# 4d
print("4d")
```

```
## [1] "4d"
```

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

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

```
abundance_time
```

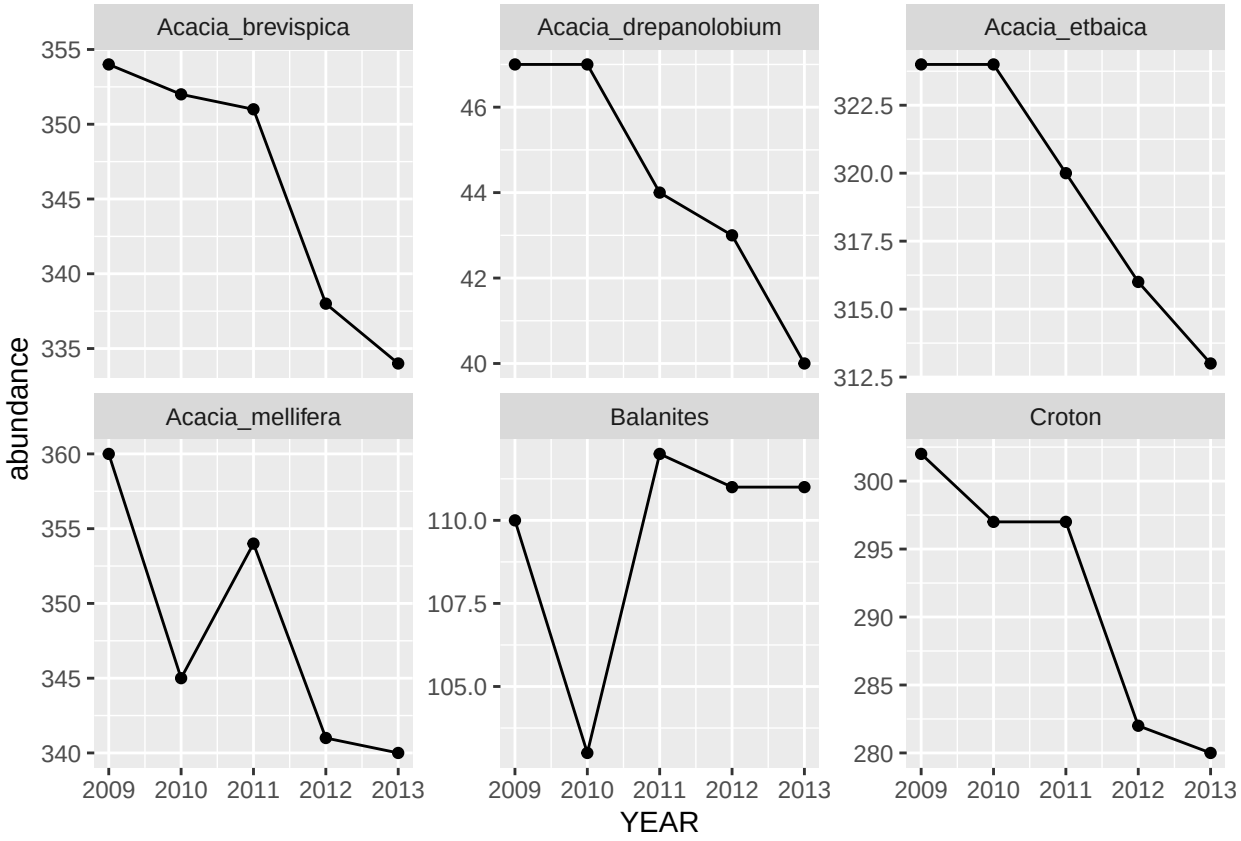
```
## # A tibble: 30 x 3
## # Groups:   YEAR [5]
##     YEAR SPECIES               abundance
##    <dbl> <chr>                     <int>
##  1  2009 Acacia_brevispica           354
##  2  2009 Acacia_drepanolobium         47
##  3  2009 Acacia_etbaica              324
##  4  2009 Acacia_mellifera            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
```

```
# 4e
print("4e")
```

```
## [1] "4e"
```

```
ggplot(data = abundance_time, mapping = aes(x = YEAR, y = abundance)) +
  geom_point() +
  geom_line() +
  facet_wrap(~SPECIES, scales = "free_y")
```




```
#ggsave("Graphing-acacia-ants-data-manip-R-5.jpeg")
```

### 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 with the code below and take a look at the dataframe. Do you see any issues that need to be addressed?

```
read_tsv("Mammal_lifehistories_v2.txt")
```

```
## 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 tibble: 1,440 x 14
##    order        family      Genus species `mass(g)` `gestation(mo)` `newborn(g)`
##    <chr>        <chr>       <chr> <chr>       <dbl>           <dbl>        <dbl>
##  1 Artiodactyla Antilocapr~ Anti~ americ~    45375            8.13        3246.
##  2 Artiodactyla Bovidae     Addax nasoma~   182375            9.39        5480
##  3 Artiodactyla Bovidae     Aepy~ melamp~    41480            6.35        5093
##  4 Artiodactyla Bovidae     Alce~ busela~   150000            7.9        10167.
##  5 Artiodactyla Bovidae     Ammo~ clarkei    28500            6.8         -999
##  6 Artiodactyla Bovidae     Ammo~ lervia     55500            5.08        3810
##  7 Artiodactyla Bovidae     Anti~ marsup~    30000            5.72        3910
##  8 Artiodactyla Bovidae     Anti~ cervic~    37500            5.5         3846
##  9 Artiodactyla Bovidae     Bison bison     497667.           8.93       20000
## 10 Artiodactyla Bovidae     Bison bonasus   500000            9.14       23000.
## # i 1,430 more rows
## # i 7 more variables: `weaning(mo)` <dbl>, `wean mass(g)` <dbl>,
## #   `AFR(mo)` <dbl>, `max. life(mo)` <dbl>, `litter size` <dbl>,
## #   `litters/year` <dbl>, refs <dbl>
```

The code below will read in the dataset and save it as the object `mammal_histories`. You should recognize the `na` argument in the `read_tsv()` function; this will handle our -999 values and convert them to NAs. This code also uses a handy function called `rename` from the `tidyverse` to rename columns, removing any special characters.

```
mammal_histories <- read_tsv("Mammal_lifehistories_v2.txt",
                             na = c("-999", "-999.00")) %>%
  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.
```

```
mammal_histories
```

```
## # A tibble: 1,440 x 14
##    order        family    Genus species mass_g gestation_mo newborn_g weaning_mo
##    <chr>        <chr>     <chr> <chr>    <dbl>        <dbl>     <dbl>      <dbl>
##  1 Artiodactyla Antiloca~ Anti~ americ~ 4.54e4         8.13     3246.       3
##  2 Artiodactyla Bovidae   Addax nasoma~ 1.82e5         9.39     5480        6.5
##  3 Artiodactyla Bovidae   Aepy~ melamp~ 4.15e4         6.35     5093        5.63
##  4 Artiodactyla Bovidae   Alce~ busela~ 1.5 e5         7.9     10167.       6.5
##  5 Artiodactyla Bovidae   Ammo~ clarkei 2.85e4         6.8        NA       NA
##  6 Artiodactyla Bovidae   Ammo~ lervia  5.55e4         5.08     3810        4
##  7 Artiodactyla Bovidae   Anti~ marsup~ 3   e4         5.72     3910        4.04
##  8 Artiodactyla Bovidae   Anti~ cervic~ 3.75e4         5.5      3846        2.13
##  9 Artiodactyla Bovidae   Bison bison   4.98e5         8.93    20000       10.7
## 10 Artiodactyla Bovidae   Bison bonasus 5   e5         9.14    23000.       6.6
## # i 1,430 more rows
## # i 6 more variables: wean_mass_g <dbl>, AFR_mo <dbl>, max_life_mo <dbl>,
## #   litter_size <dbl>, litters_per_year <dbl>, refs <dbl>
```

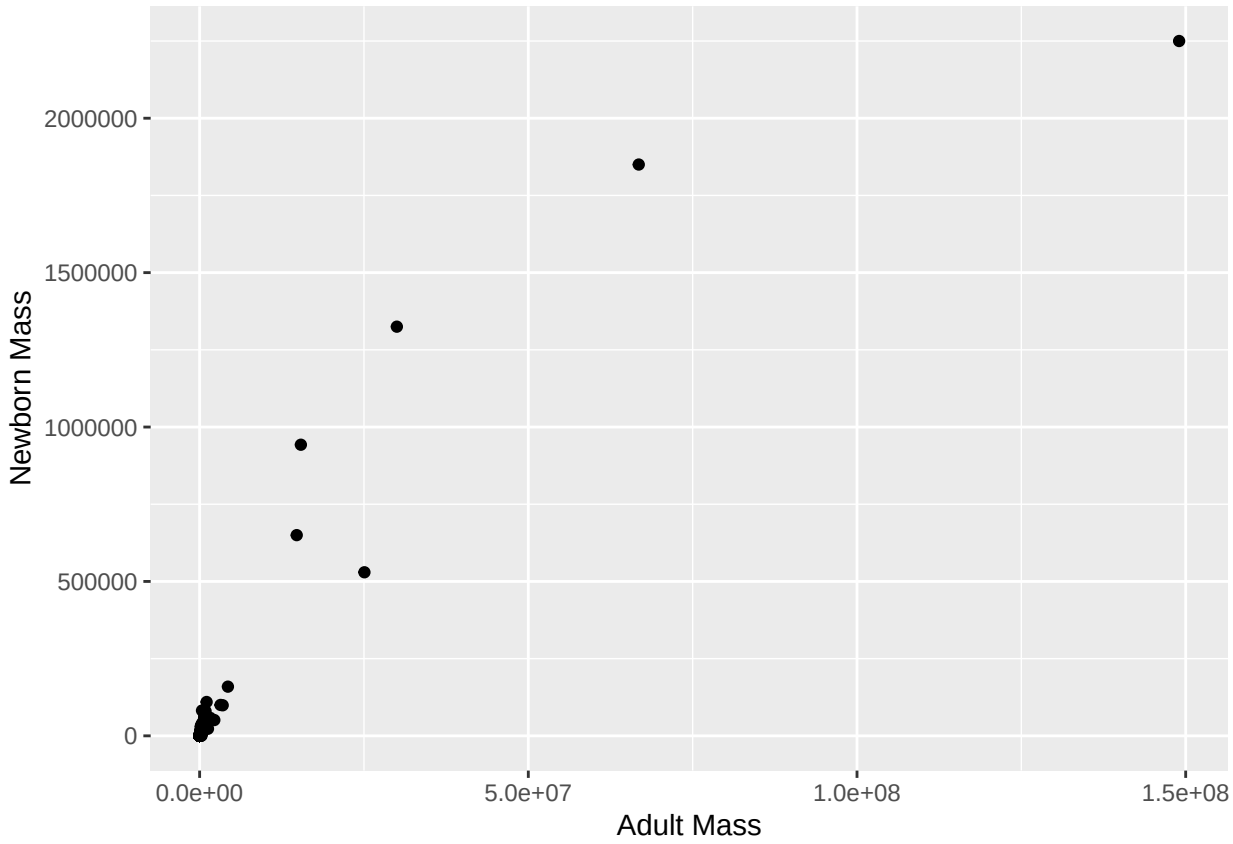
a. Graph adult mass vs. newborn mass. Label the axes with clearer labels than the column names.
b. It looks like there's a regular pattern here, but it's definitely not linear. Let's see if log-transformation straightens it out. Graph adult mass vs. newborn mass, with both axes scaled logarithmically.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

```
# a. Graph adult mass vs. newborn mass.
print("5a")
```

```
## [1] "5a"
```

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

```
## Warning: Removed 624 rows containing missing values or values outside the scale range
## (`geom_point()`).
```

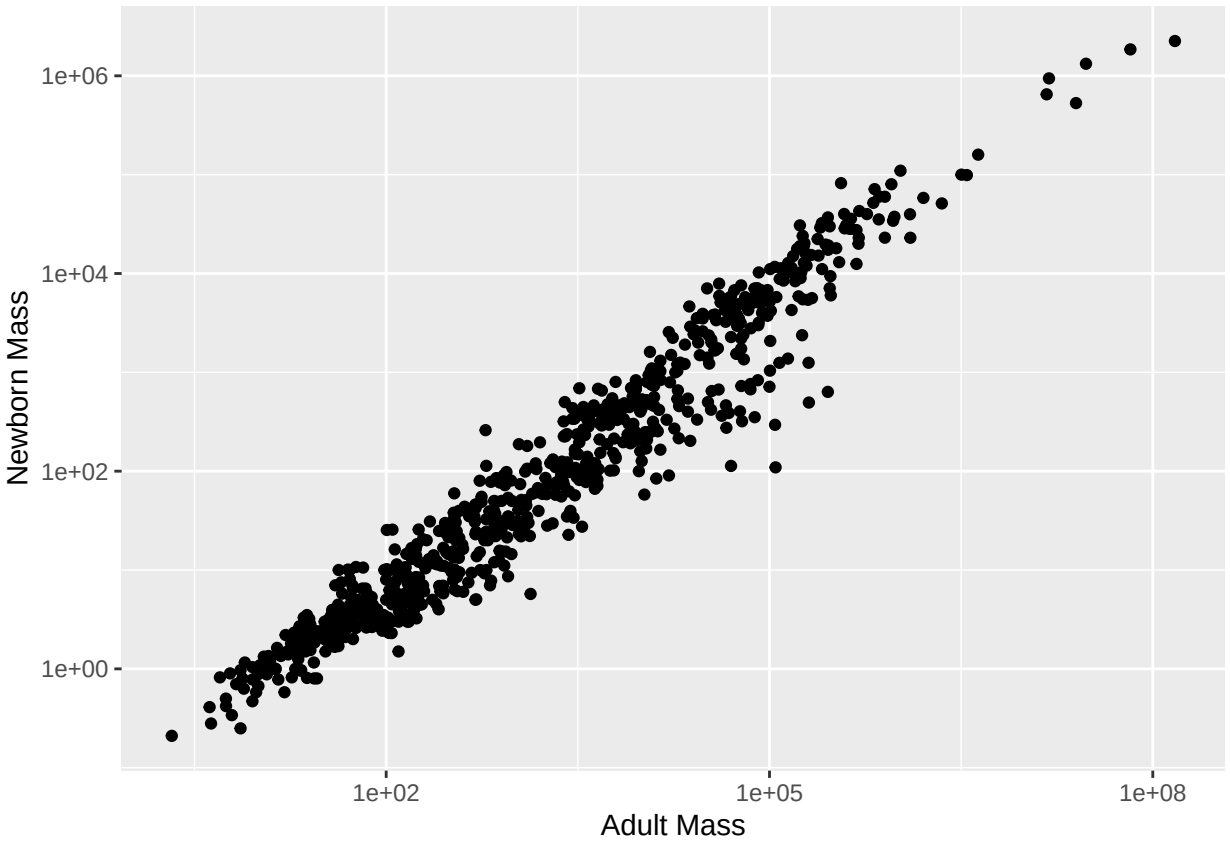



```
# b. Graph adult mass vs. newborn mass, with both axes scaled logarithmically.
print("5b")
```

```
## [1] "5b"
```

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

```
## Warning: Removed 624 rows containing missing values or values outside the scale range
## (`geom_point()`).
```

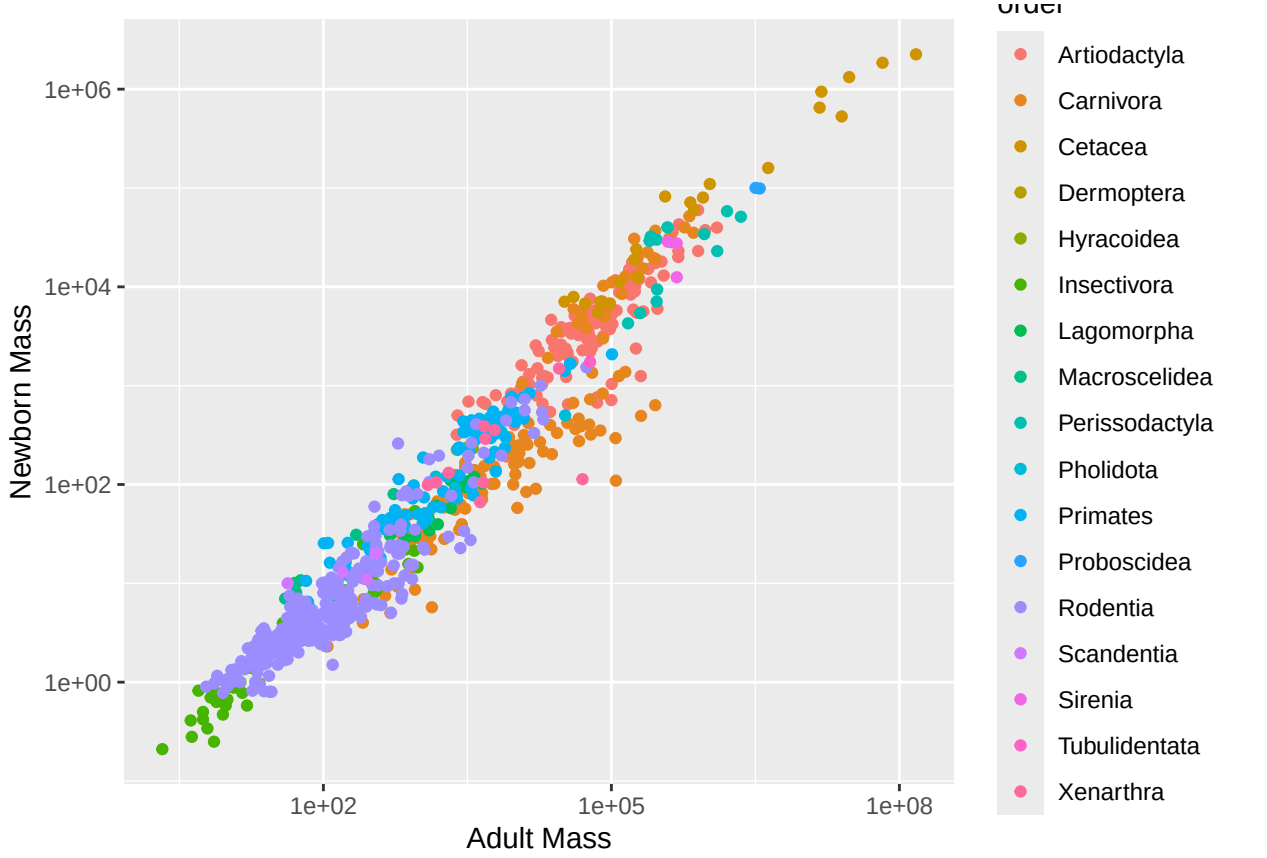



```
# c. Graph adult mass vs. newborn mass, log-scaled, with data colored by order.
print("5c")
```

```
## [1] "5c"
```

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

```
## Warning: Removed 624 rows containing missing values or values outside the scale range
## (`geom_point()`).
```

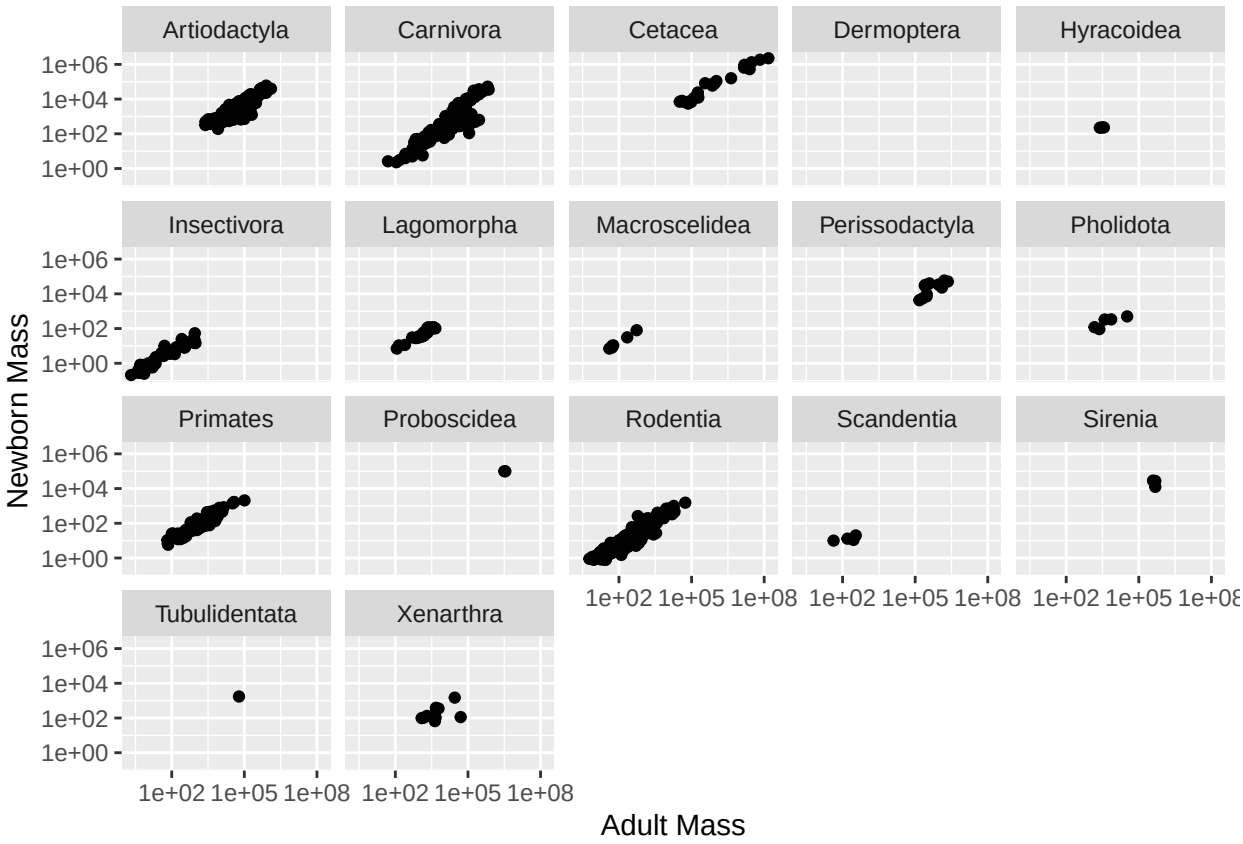



```
# d. Use `facet_wrap` to create subplot for each order.
print("5d")
```

```
## [1] "5d"
```

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

```
## Warning: Removed 624 rows containing missing values or values outside the scale range
## (`geom_point()`).
```

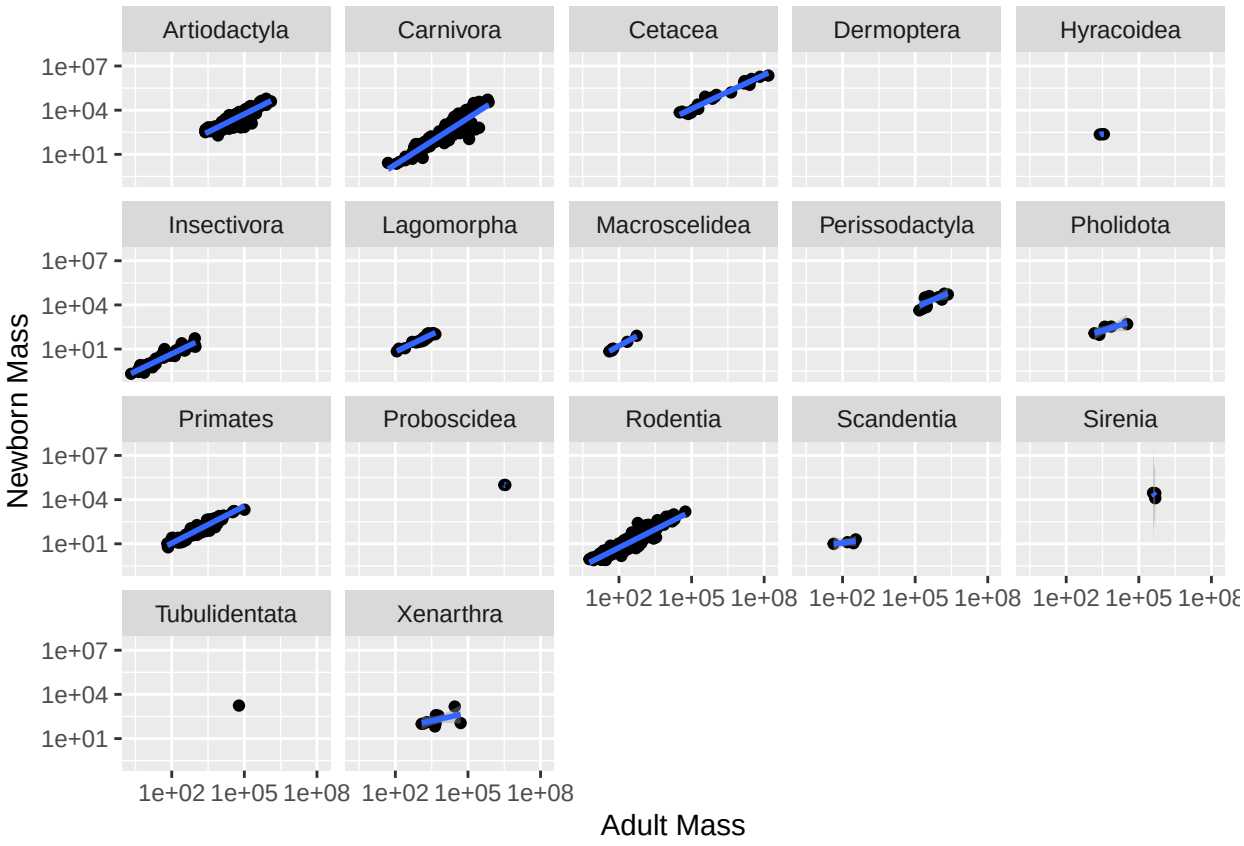



```
# e. use `geom_smooth` to fit a linear model to each order.
print("5e")
```

```
## [1] "5e"
```

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

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

## Warning: Removed 624 rows containing non-finite outside the scale range
## (`stat_smooth()`).

## Warning in qt((1 - level)/2, df): NaNs produced

## Warning: Removed 624 rows containing missing values or values outside the scale range
## (`geom_point()`).

## Warning in max(ids, na.rm = TRUE): no non-missing arguments to max; returning
## -Inf
```




### 6. Check That Your Code Runs

Sometimes we think our code runs, but it only actually works because of something else we did previously.

To make sure it actually runs, you should save your work and then run it in a clean (read: empty) environment.

I have actually be forcing the issue by having you "knit" your file before submitting it. To successfully knit, every RMarkdown file must be "self-contained," meaning every line of code that is necessary to run anything in the .Rmd file must be written in the .Rmd file. The file doesn't recognize things already in the environment.

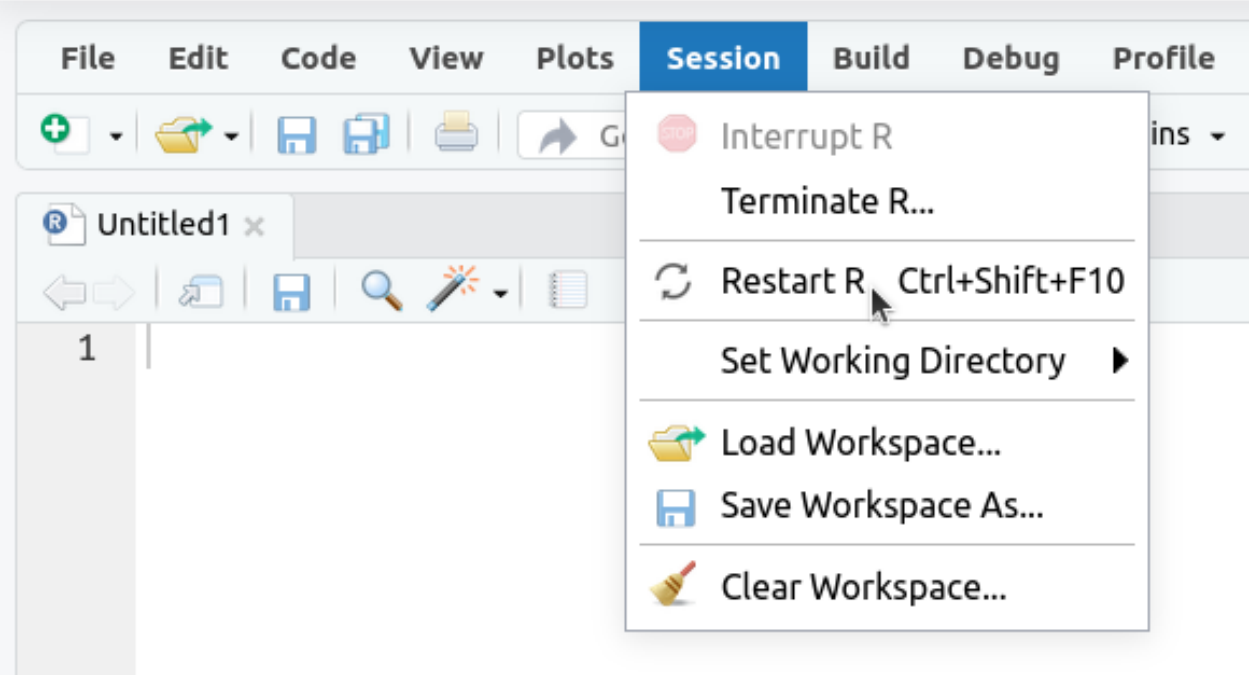
For example, if you use a function from the `readr` package (e.g., `read_csv`) anywhere in your .Rmd file, you will need to have `library(readr)` in a code chunk in your .Rmd file before any line with `read_csv`.

Similarly, if you have a line of code referencing an object (like a data frame), you need to have a line of code beforehand in your .Rmd file that creates the object.

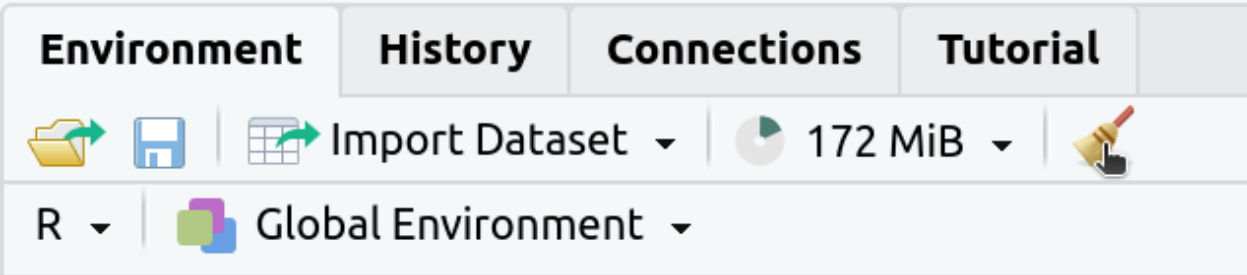
While attempting to knit a document is one way to learn if your code actually runs the way you think it should, there are other (and maybe better) ways to test this out before you ever try to knit a document.

Follow these steps in RStudio to make sure your code really runs:

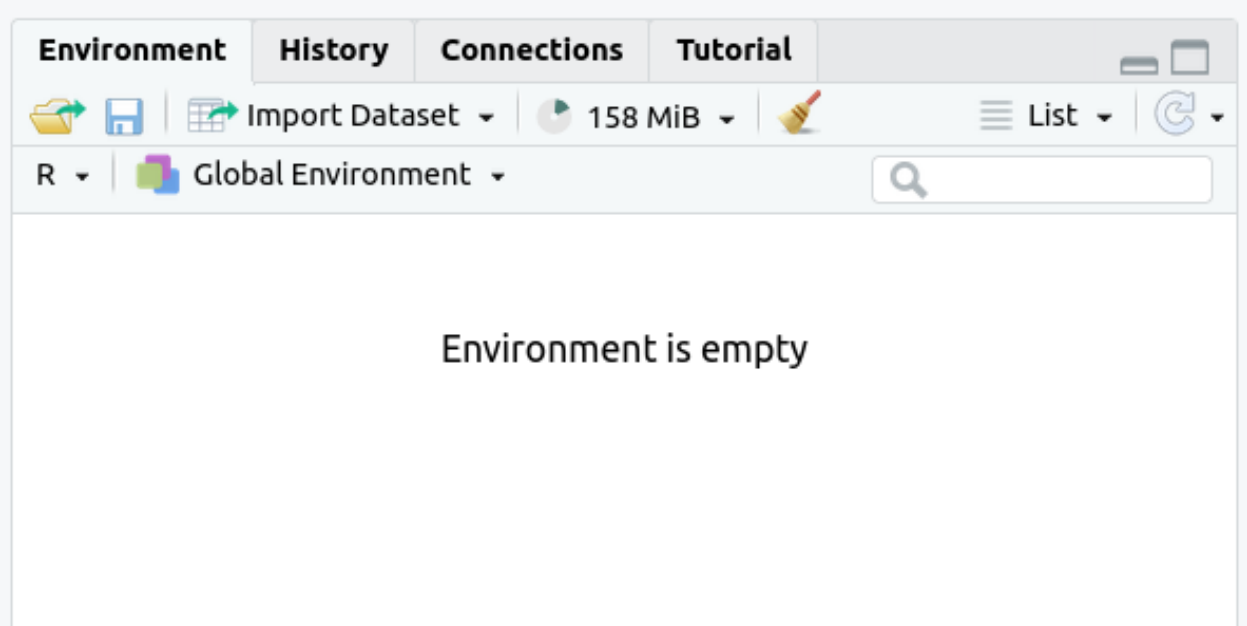
1. Restart R by clicking `Session` in the menu bar and selecting `Restart R`.



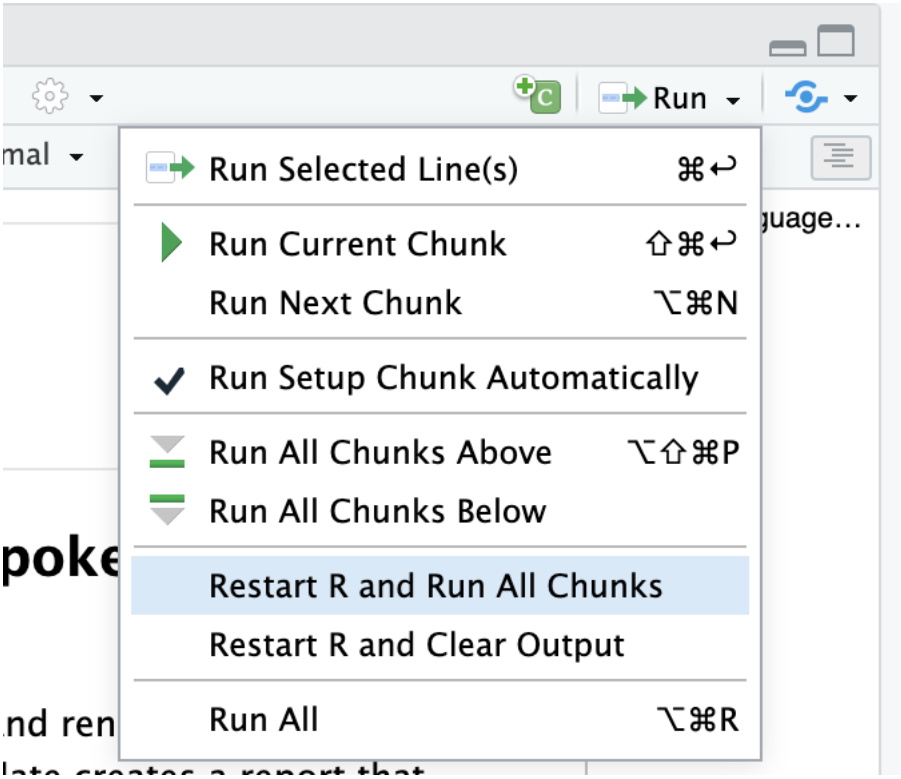

2. If the `Environment` tab isn't empty, click on the broom icon to clear it.




The `Envirnment` tab should now say "Environment is empty."



3. Run all of the code you have written in your assignment so far by selecting `Restart R and Run All Chunks`,`Run All`, or using the keyboard shortcut (`Ctrl`+`Alt`+`R` or `Cmd`+`Opt`+`R`)




If your code does *not* run all the way through, check the error messages. This will helpful you isolate the potential issue.

No need to type any answers here. If you are able to submit a PDF, you're good to go.

### 7. Graphing Data from Multiple Tables (*Optional*)

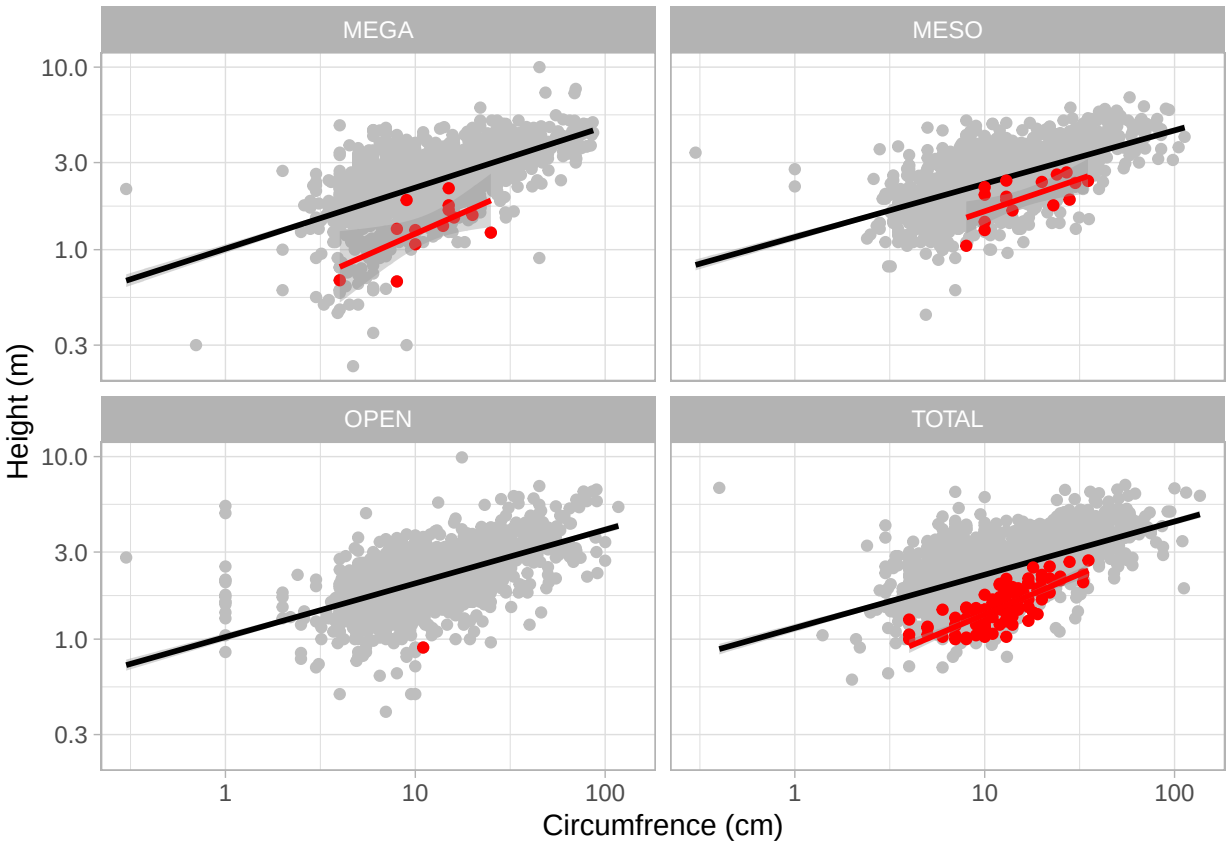
This questions uses the `acacia` and `trees` data frames.

We want to compare the circumference to height relationship in acacia on different treatments in the context of the same relationship for trees in the region.

Make a graph with the relationship between `CIRC` and `HEIGHT` for the trees as gray points in the background and the same relationship for acacia as red points plotted on top of the tree points.

There should be one subplot for each treatment. Scale the both axes logarithmically. Include linear models for both sets of data. Provide clear labels for the axes. Add a theme.

```
ggplot() +
  geom_point(data = trees, aes(x = CIRC, y = HEIGHT), color = "gray") +
  geom_smooth(data = trees, aes(x = CIRC, y = HEIGHT), color = "black", method = "lm") +
  geom_point(data = acacia, aes(x = CIRC, y = HEIGHT), color = "red") +
  geom_smooth(data = acacia, aes(x = CIRC, y = HEIGHT), color = "red", method = "lm") +
  facet_wrap(~ TREATMENT) +
  labs(x = "Circumfrence (cm)",
       y = "Height (m)") +
  scale_x_log10() +
  scale_y_log10() +
  theme_light()
```


MEGA
MESO
OPEN
TOTAL
Height (m)
10.0
3.0
1.0
0.3
1
10
100
Circumfrence (cm)