HW 10: Data Visualization I

Graphical Analysis of Biological Data

By the end of this assignment, you should demonstrate an ability to visually analyze data for

- outliers,
- normality,
- relationships among variables, and
- interactions.

Click on any blue text to visit the external website.

Note: If you contact me for help or (better yet) open an issue in the public discussion forum, please include the code that is not working and also tell me what you have tried.

Preparation

- Open your `.Rproj` project file in RStudio.
- Create an `hw10` folder inside the same folder as your project file.
- Create a new notebook file called `<lastname>_10.Rmd`. Edit the YAML file as you have for previous assignments. Save this inside your `hw10` folder.
- Right-click and save sparrows.txt to your data folder.
- Install the `GGally` and the `patchwork` packages. **Do not include the installation code in your notebook.** Install from the console or the `Tools` menu.
- Add a code chunk to load the `tidyverse`, `here`, `ggally`, and `patchwork` libraries.
- Commit early. Commit often. Push regularly but at least push your completed assignment.
- Download and open A protocol for data exploration to avoid common statistical problems (free access) by Zuur et al. 2010.

```
library(tidyverse)
library(here)
library(GGally)
library(patchwork)
```

The Zuur et al. 2010. paper outlines a series of steps that should be performed *always* on a data set before your real analysis begins. You must have a feel for the "structure* of the data, identify outliers, and so on. The purpose of this exercise is to help you learn how to do these essential steps in R.

You will recreate Figures 2, 3, 5, 10, and 11 from the Zuur et al. 2010. paper. You will learn some `ggplot` tricks along the way. I will show you most of the code, which I expect you to type in to the R Notebook you just created and run it. Study the code because you will have to apply it in the next assignment. In a few cases, I will not give you the code because you should know what to do by now. I will just tell you what to do. Like this.

Import and wrangle

- Import `sparrow.txt`. I will leave it to you to figure out if the data are comma-delimited or tab-delimited to choose the proper import function. I imported the raw data into a variable called `sparrow_raw`, which I leave unmodified. Reemeber to use `here` to import the data from your data folder.
- A good idea is to `View()` the imported data to be sure the data look like they were properly imported. Run this from the console or click the Environment tab (upper right pane) and then click on the data set name.

The `sparrows.txt` data set has morphometric measurements for 1295 Saltmarsh Sparrows, gathered by Gjerdrum et al. 2008

The first step is to create two new columns in the sparrow dataset. The `box_group` column contains a single arbitrary value. We'll use "Sparrows" because it will make sense when we plot. We have to do this step because `geom_boxplot` requires that data be mapped to the x-axis. If we used something like `Sex` or `SpeciesCode`, we would end up with more than one boxplot. In most circumstances, we would do that but, because the goal is to mimic Zuur's figure 2, we have to use a dummy variable.

The `row_order = 1:nrow(.)` creates another dummy variable that we need. The `nrow()` function returns the number of rows in a data frame. The `.` tells `nrow()` to use the `sparrows_raw` dataframe that was piped in. The result is that `row_order` column will have a sequence of numbers from 1 to 1265, the number of rows in the sparrows data set. We'll save the result in a new tibble called `sparrows`. I use this tibble for the remaining steps. If we screw up, we can start fresh from the `sparrows_raw` data.

```
sparrows <- sparrows_raw %>%
  mutate(box_group = "Sparrows",
         row_order = 1:nrow(.))
```

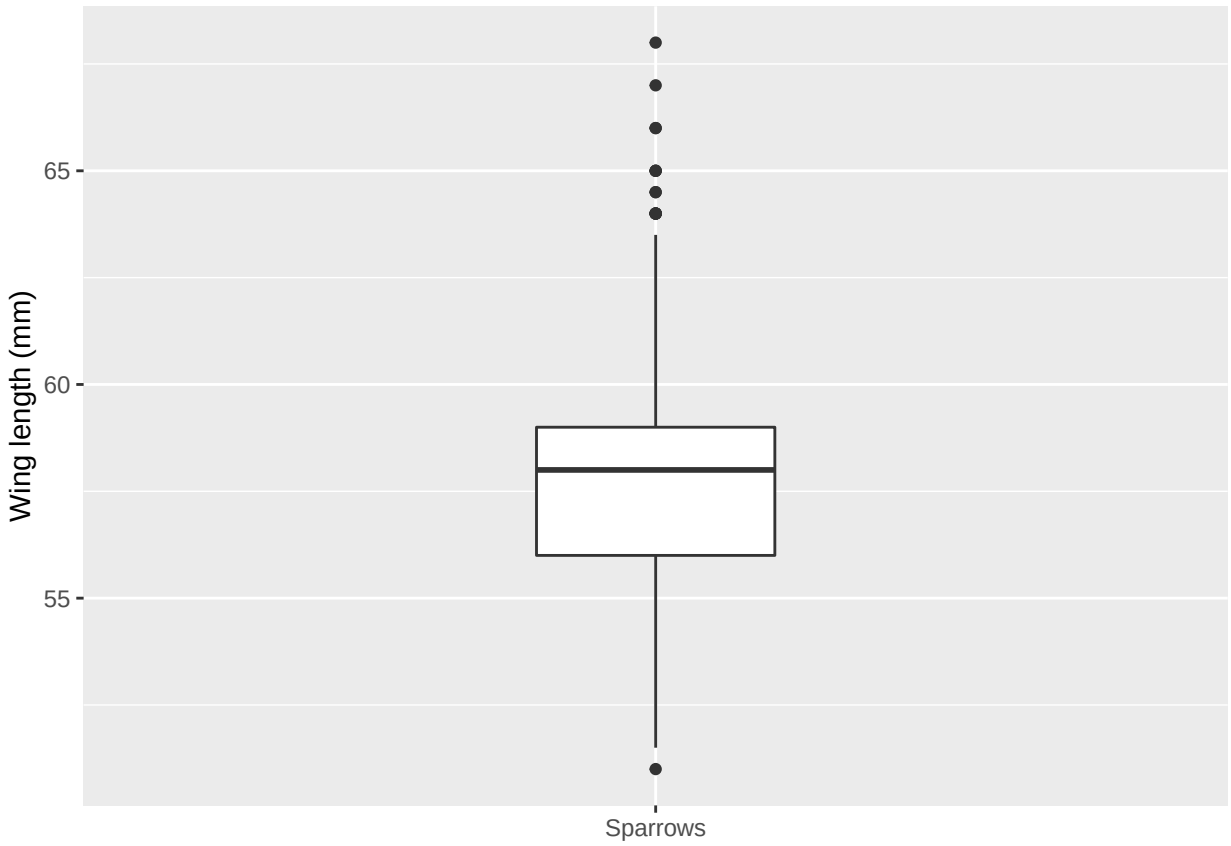
You should run `View(sparrows)` to ensure that you have both columns.

Figure 2: outliers

From here on, all references to figures will be to the Zuur et al. paper that I told you to download and open. You do have that open, right?

The left panel of Figure 2 shows a box plot of wing length (mm) for the entire data set. `wingcrd` is the column with wing length. Enter and run this code to generate the boxplot. Notice our code is using the `box_group` dummy variable. Our plot is identical to the Zuur plot except for some details.

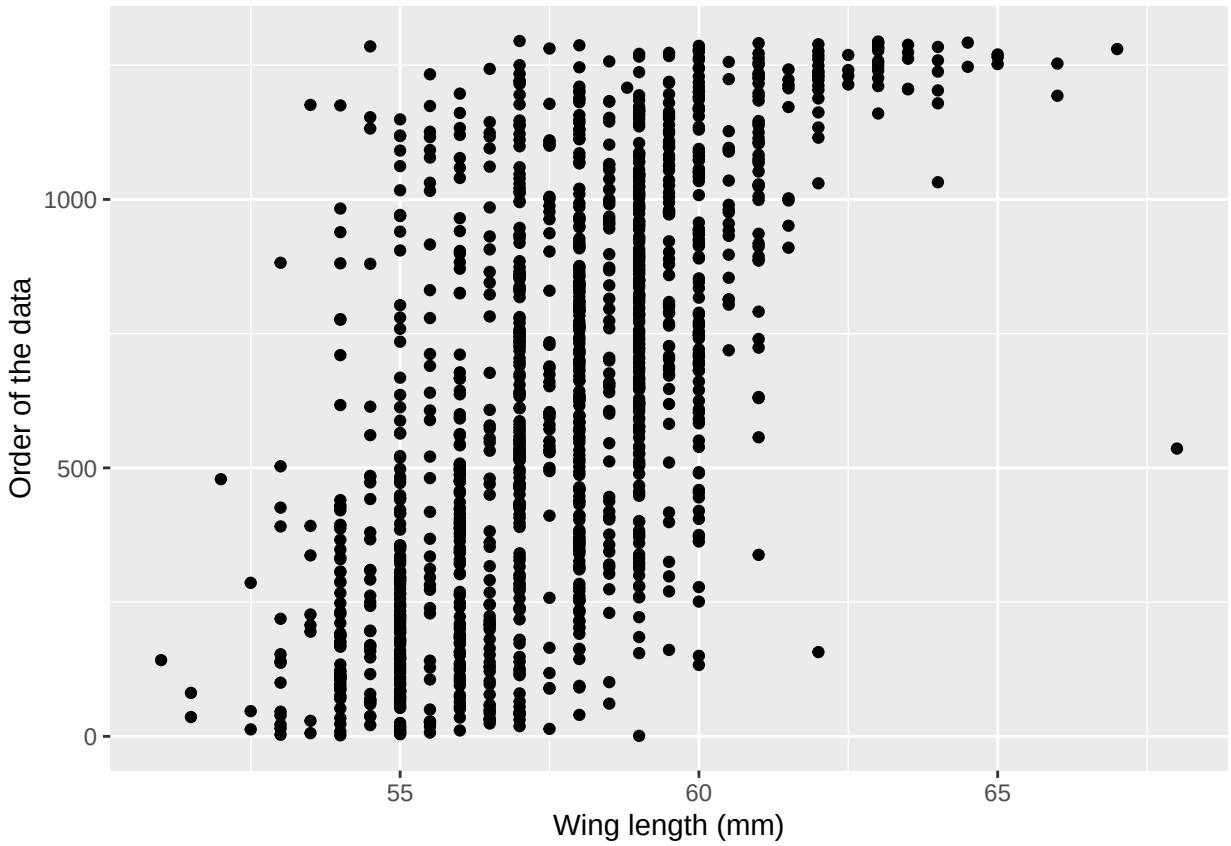
```
sparrows %>%
  ggplot(aes(x = box_group,
             y = wingcrd)) +
  geom_boxplot(width = 0.25) +
  labs(x = NULL,
       y = "Wing length (mm)")
```



The box plot identified *possible* outliers because the data points fall outside of the 1.5 * interquartile range. Outliers usually have values *very* different from most other values. In this boxplot, the outlier points are just beyond the whiskers, suggesting that the "outliers" may not be that different from the other values.

The full range of values can be assessed with a Cleveland dot plot, as shown in the right panel.

```
sparrows %>%
  ggplot(aes(x = wingcrd,
             y = row_order)) +
  geom_point() +
  labs(x = "Wing length (mm)",
       y = "Order of the data")
```



To finish recreating the figure, we need to store our two plots as separate objects, then take advantage of `patchwork` to put our two plots together. Plots made by `ggplot` are actually R list structures, which means plots can be stored in variables.

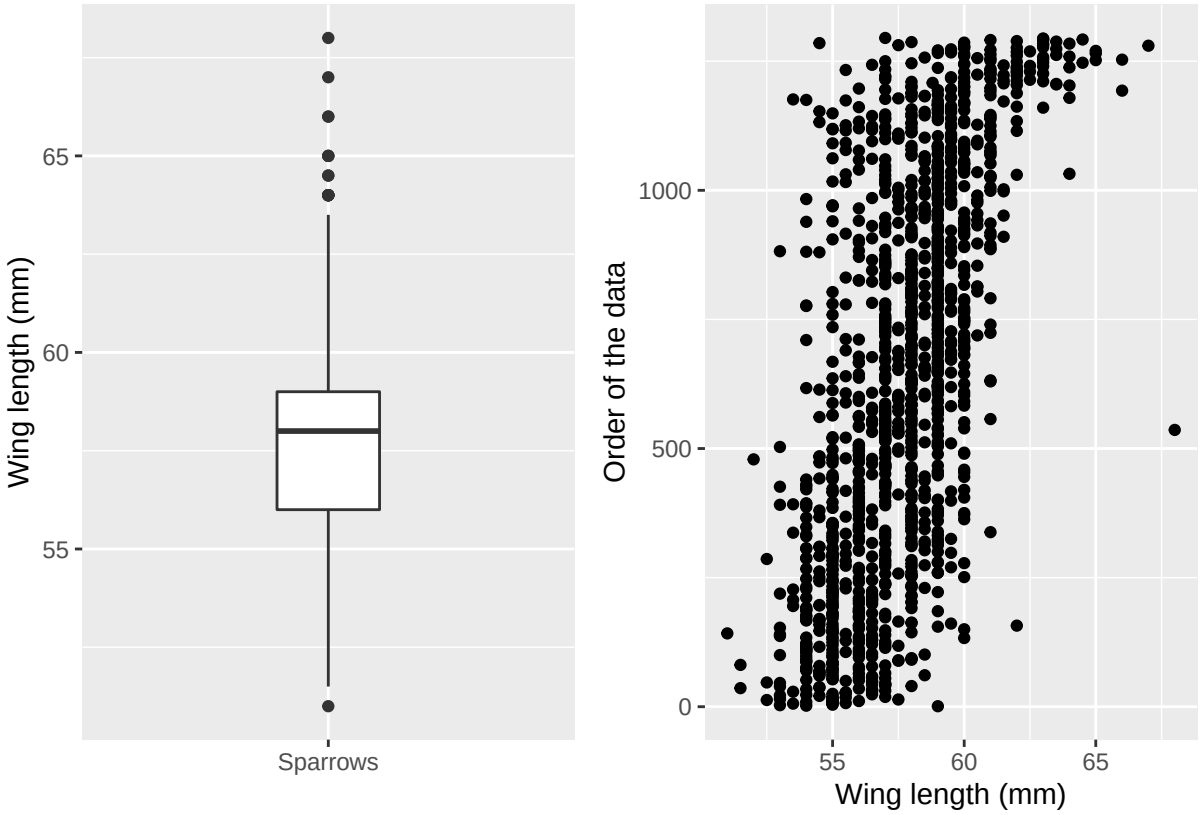
We want to store our first two plots in separate variables. For simplicity, we'll call them `plot_box` and `plot_cleveland`. Go back to the previous code chunks so that they store the results in their respective variables. When you are finished, they should look this this.

When you run the code, you will not see the plots but they are stored in their variables. You can see that by entering `plot_box` or `plot_cleveland` on lines by themselves.

Before you move further, look at the examples on the patchwork GitHub page (scroll down). The first example shows what we want to do: put two plots side-by-side. In the example, the two plots were stored as `p1` and `p2`, respectively. To plot them side-by-side, they used `p1 + p2`.

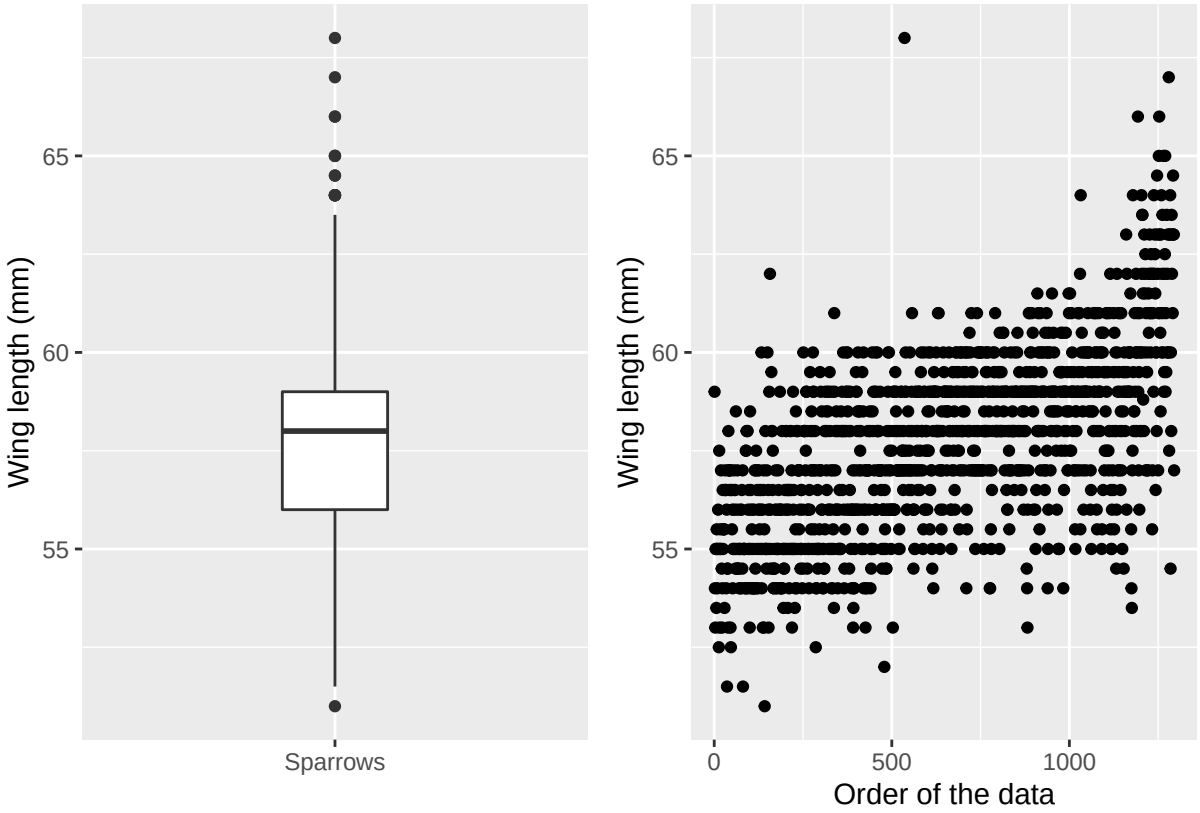
We just substitute our two plot names in the same fashion.

```
plot_box + plot_cleveland
```



Bingo! We've recreated Figure 2. Unfortunately, the wing lengths are shown on the y-axis of the boxplot but the x-axis of the Cleveland plot. The authors note in the figure legend that it can be helpful to have the axes aligned. We will take advantage of `ggplot` layers to make a very simple improvement, by adding a `coord_flip` layer.

```
plot_box + plot_cleveland + coord_flip()
```



Comparison shows that the longest wing (row 536) is not very different at all from the other long wings. Similarly, the shortest wing outlier is not very different from the other short wings. That suggests these data are okay to include in subsequent analyses.

Figure 3: outliers

A good scientist such as yourself will look at every variable in the data set to search for outliers. Figure 3 shows six more variables from the `sparrows` data set.

Before we begin, look again at the patchwork examples. You can add a `plot_layout(ncol = )` layer to specify the number of columns. (You could also use `nrow`.) Figure 3 has three columns in two rows. For example, if we wanted Figure 2 to place one graph above the other, we have run `plot_box + plot_cleveland + plot_layout(ncol = 1)`

An assignment

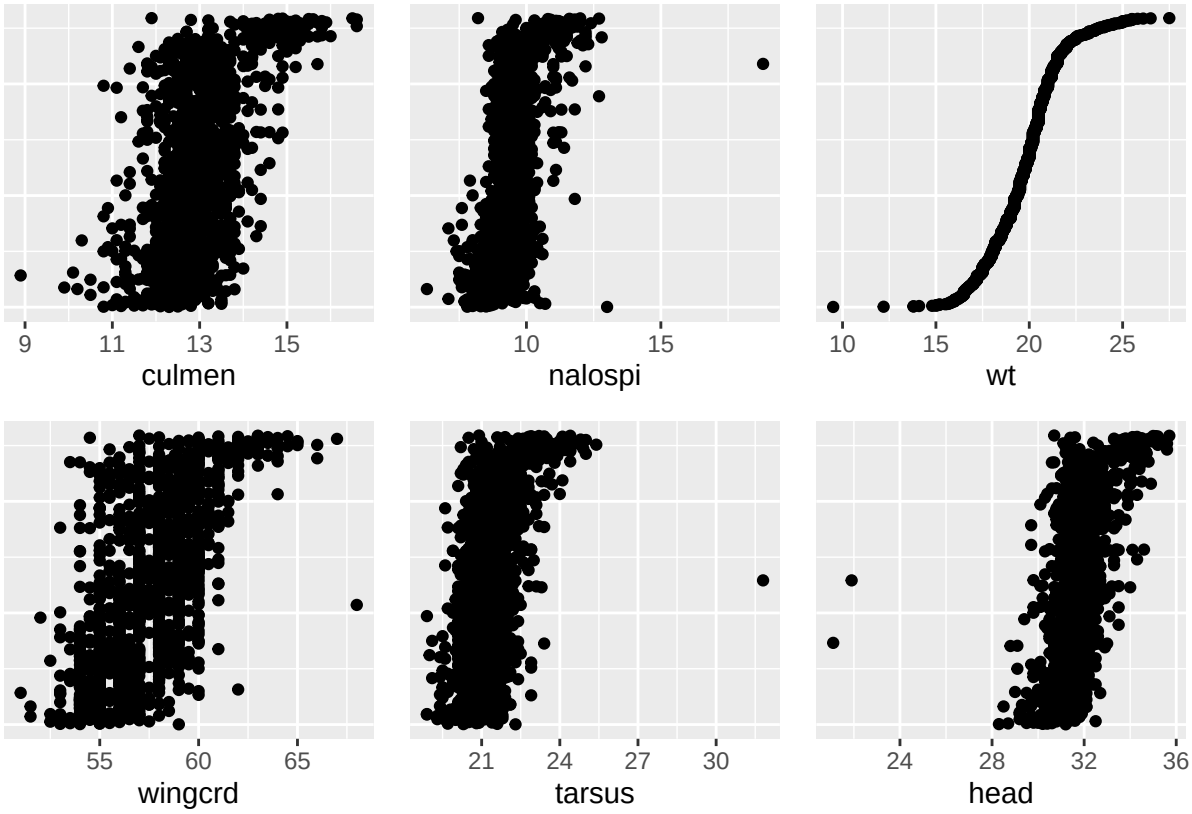
You are to write a single chunk of code that makes the six graphs shown in Figure 3 and lays them out as shown. I will start you off with the code for the upper left panel (Culmen length). You can adjust the code accordingly. Include but do not worry about the `theme()` lines. They remove the y-axis information which we do not need. Notice that I use `p_<variable>` to indicate the name of each saved plot. For example, the first variable is `culmen` so I saved the plot as `p_culmen`.

While I encourage you to type every example, you have my permission to copy and paste, and then adjust the variables as necessary. I suggest you run each plot independently first to be sure it matches the corresponding panel in Figure 3 before assembling them into the final figure.

```
p_culmen <- sparrows %>%
  ggplot(aes(y = row_order)) +
  geom_point(aes(x = culmen)) +
  theme(axis.title.y=element_blank(),
```

```
        axis.text.y=element_blank(),
        axis.ticks.y=element_blank())
```

The other five variables are `nalospi`, `wt`, `wingcrd`, `tarsus`, and `head`. **Make a new code chunk and include the code above plus the code to produce the other five graphs, and then lay them out as shown in the figure.** If successful, your plot should look like this.



Question: Which three variables appear to have outlier data?

Figure 5: distribution

The histogram in left panel of Figure 5 shows the number of indivdiuals in each "bin", or weight class of birds. In their figure, each bin is 0.5 grams wide. For example, notice there are four bins (or columns) between 18 and 20. The first bin includes weights from 18.1-18.5 g. The second bin includes weights from 18.5-19.0 g. The third bin includes weights from 191.-19.5 g. The fourth bin includes weights from 19.6-20.0 g.

The number you use for the `binwidth` argument depends on the precision of your measurements. If you have weight measurements in 100s of grams, then binwidths of 250 or 500 or even larger may be appropriate. If your units of measurements are around 0.01, then bin widths of 0.05 or 0.1 or 0.25 might be more effective. You usually have to play around with the `binwidth` to find bins that make sense.

Note: I used the `scale_x_continuous()` layer to manually set the x-axis scale to mimic the figure. `seq(14, 28, by = 2)` generates a sequence of numbers from 14 to 28 in increments of 2; i.e., 14, 16, 18, . . . , 28. You are not requierd to use this option in your plots. If you do, the first number in the `seq()` funtion will be the left most number on the x-axis. The second number will be the right-most number, and `by =` will be the incremement.

```
# Histogram
h1 <- sparrows %>%
```

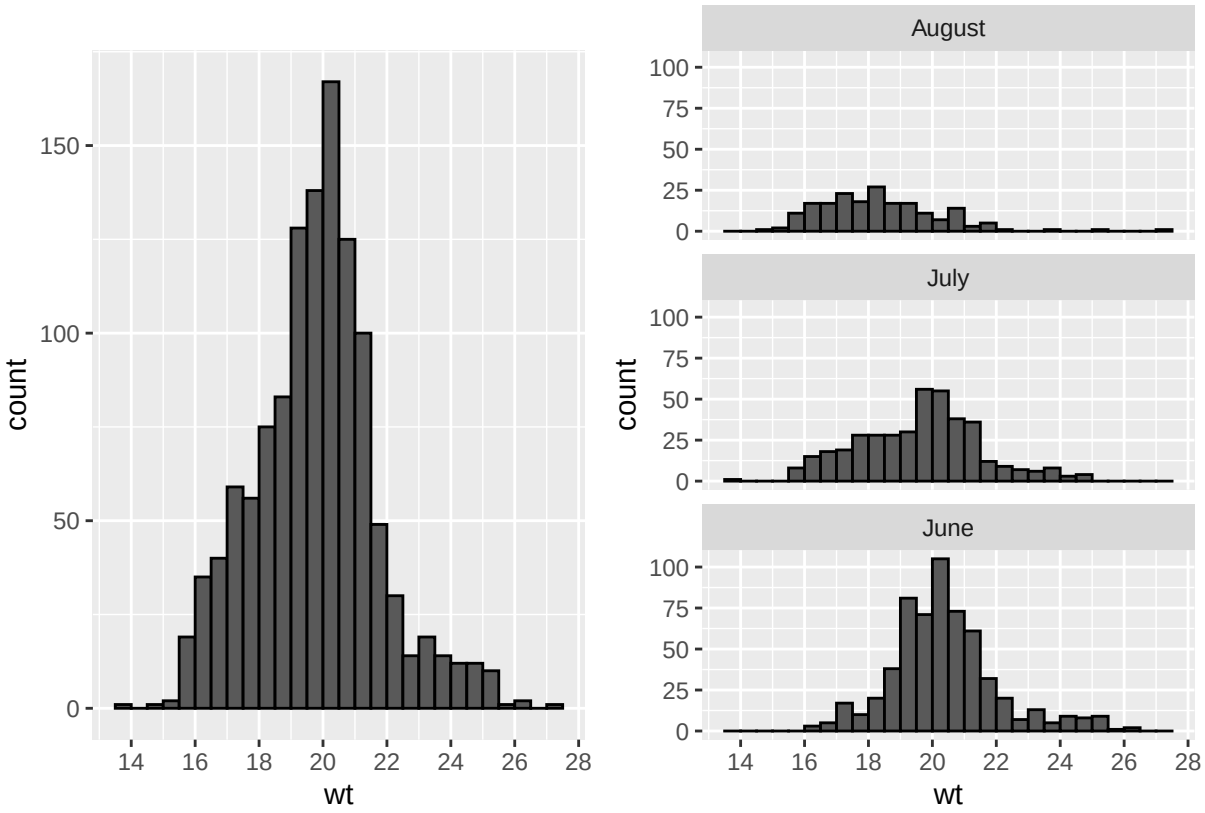
```
  filter(Month %in% 6:8) %>%
  ggplot() +
  geom_histogram(aes(x = wt),
                 binwidth = 0.5,
                 boundary = 0,
                 closed = "right",
                 color = "black") +
  scale_x_continuous(breaks = seq(14, 28, by = 2))
```

Now we have to create the right panel. This code filters out all months except 6, 7, or 8, then mutates the numbers to the month names. `facet_wrap()` arranges the plots alphabetically. We get lucky here because the alphabetical order of June, July, and August correspond to their calendar order. Below, we will have to set the order using factors.

```
# Faceted Histogram
h2 <- sparrows %>%
  filter(Month %in% 6:8) %>%
  mutate(Month = case_when(
    Month == 6 ~ "June",
    Month == 7 ~ "July",
    Month == 8 ~ "August")) %>%
  ggplot() +
  geom_histogram(aes(x = wt),
                 binwidth = 0.5,
                 boundary = 0,
                 color = "black") +
  scale_x_continuous(breaks = seq(14, 28, by = 2)) +
  facet_wrap(~ Month, ncol = 1)
```

Finally, we assemble the two plots into a single figure with `patchwork`.

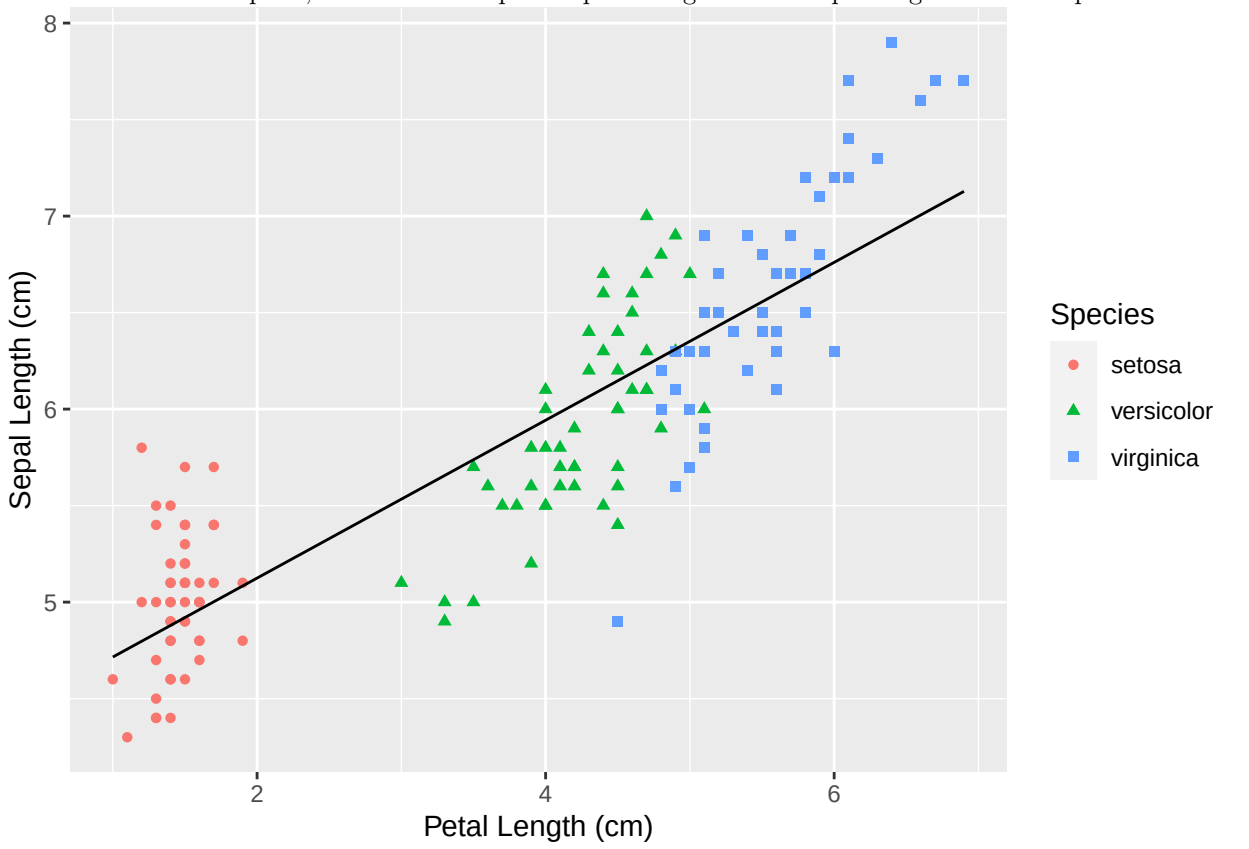
```
h1 + h2
```



Grad and Honors: Recreate this figure using density plots. However, modify the right panel so that all three months are overlaid on the same plot. Go back to the notes and follow the example code for guidance.

Figure 10: Relationships

You have used scatterplots, like this scatterplot of petal length versus sepal length for three species of *Iris.* .



Scatterplots are useful for showing the relationship between two variables. The scatterplot above shows a positive linear relationship between the length of petals and sepals. Individuals (and species) with longer petals tend to have longer sepals. Taken together, those individuals tend to have larger flowers.

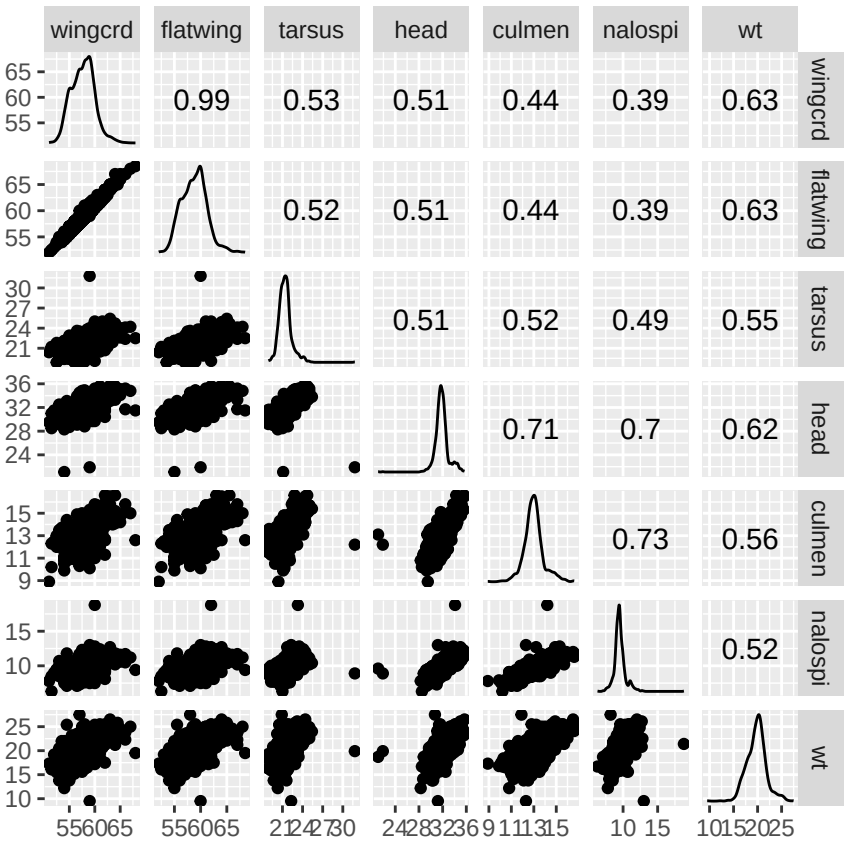
Another essential step in any graphical analysis of data is to explore relationships among the variables, like that shown by Figure 10. The panels above the diagonal are scatterplots that show pairwise relationships between the continuous variables in the sparrow data. The intersection of a row with a column shows the scatterplot for a specific pair of variables. The panels below the diagonal shows the Pearson correlation coefficients, r. The coefficients range from 0 to 1. Higher values indicate stronger relationships between the two variables.

As you might imagine, writing the code for dozens of scatterplots could quickly grow tedious, even with copy and paste. You have to ensure you changed each of the variables exactly right. Fortunately, the GGally package provides the ggscatmat and ggpairs functions. Before you go farther, scroll down to browse through the examples for these functions. `ggscatmat` creates only a scatterplot matrix, which we will use. `ggpairs()` can produce many more types of plots in the matrix, as you saw looking through the examples.

We will use the `columns` argument of `ggscatmat` to specify the first seven columns of data. If you want to plot all columns (helpful but not always wise), you do not have to supply the `columns` argument.

Note: `ggscatmat` and `ggpairs` do a lot of work so this step may take more than the usual time, depending on the speed of your computer. ‘`ggscatmat` is faster than `ggpairs` because it does less.

```
sparrows %>% ggscatmat(columns = 1:7)
```



`ggscatmat` includes a density plot on the diagonal and puts the scatterplots below the diagonal, but we otherwise recreated Figure 10 with a single line of code. Niiiice.

Figure 11: interactions

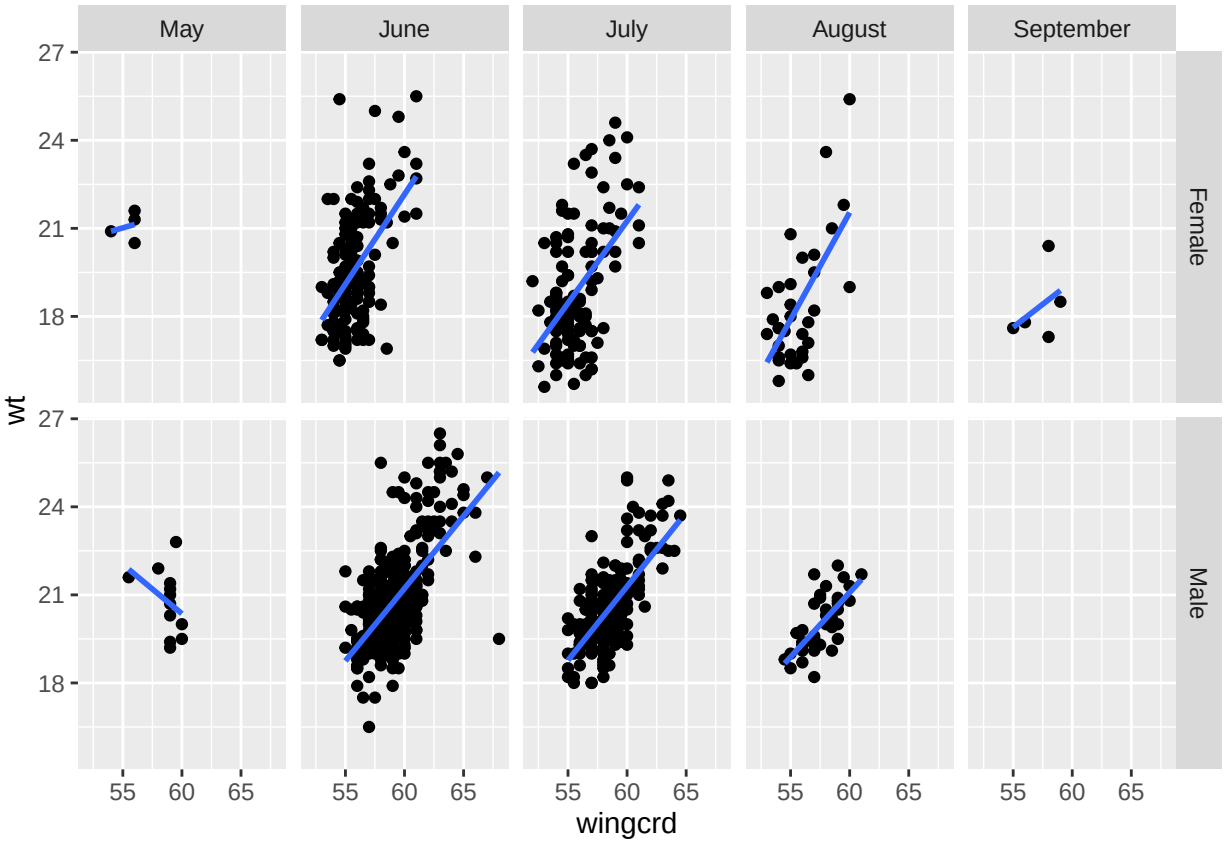
Interaction effects among independent variables that affect the dependent variables can confound a statistical analysis. For example, assume you are studying growth rate and time to sexual maturity in a territorial species. In many species, males grow faster to be able to defend a territory sooner but at the cost of delayed sexual maturity. Male energy is directed toward growth rather than reproduction early in life. In contrast, females might become sexually mature more quickly but at the cost of smaller body size. This represents an interaction between sex and energy investment that influences growth rate and time to sexual maturity.

Run the code below to show the interaction between sex and season (months). A few things about the code. `filter` is keeping only months 5-9 (May through September) and removing immature individuals (coded as 0). `mutate` changes the months from numbers to ordered names (factors) so that the month names appear in proper order as columns. `geom_smooth` uses `method = "lm"` (linear model) to add the trend line, and `se = FALSE` removes the default standard error shading around the trend line.

Note: The code here uses the `factor()` function just like you have used in Base R, but we convert Months to factors as part of `mutate()`.

```
month_order = c("May", "June", "July", "August", "September")
sparrows %>%
  filter(Month %in% 5:9,
         Sex != 0) %>%
  mutate(Month = case_when(
    Month == 5 ~ "May",
    Month == 6 ~ "June",
```

```
    Month == 7 ~ "July",
    Month == 8 ~ "August",
    Month == 9 ~ "September"),
    Month = factor(Month, levels = month_order, ordered = TRUE),
    Sex = ifelse(Sex == 4, "Male", "Female")) %>%
  ggplot(aes(x = wingcrd,
             y = wt)) +
  geom_point() +
  geom_smooth(method = "lm", se = FALSE) +
  facet_grid(Sex ~ Month)
```



If trend lines are parallel, or roughly so, then interaction effects are probably absent or not significant. In the figure above, male and female weights trend in opposite directions during May. This could be because females are producing eggs and males are more aggressively defending terroritories, gathering nest material, or something else. However, as noted in the text, the sample sizes for May are very small, so the best approach would be to gather more data if possible. If collecting additional data is not possile, then that result should be interepreted cautiously.

You are now armed with some of the skills and tools needed for the first stages of robust data analysis. Put your new skills to test in the next assignment.

---

et Vóila