HW 09: Wrangling II

Graphical Analysis of Biological Data

By the end of this assignment, you should be able to achieve the following tasks in R:

- use R notebooks and R markdown;
- insert, write, and evaluate code chucks;
- import data stored in a variety of text file formats;
- make untidy data tidy;
- mutate, summarize, and group data;
- use pipes;
- produce plots with `ggplot2`;
- use a typical workflow to wrangle and plot data;
- be able to source custom functions, and
- confidently stage, commit, and push with Git.

These achievements belong to Learning Outcomes 2, 3, 4, 5, 6.

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.
- If you haven't already, install the `here` package from `Tools > Install Packages...` or run `install.packages("here")` from the console.
- Install the `smatr` package.
- You should already have an `scripts` folder inside the same folder as your project file. If not, make it now and make sure your *working* `my_functions.R` script from HW08 is inside the `scripts` folder.
- Create an `hw09` folder inside the same folder as your project file.
- Create a new notebook file called `<lastname>_09.Rmd`. Edit the YAML file as you have for previous assignments. Save this in the `hw09` folder.

Wrangling II

Your task here is simple. Write code chunks that

- import data,
- wrangle the data as specified, and
- graph the data. Correct plots are shown throughout so you know your goal. Plots for the graduate/honors student section are not shown.

Then, describe the results briefly. Identify patterns and trends, potential outliers, and other results you find interesting. Think like the scientist you are!

You may need to review your Introduction to R coding exercise to meet some requirements, especially factors and how to select one column from a dataframe, and how to change one value to another.

A few things:

- Download feathers.csv and darters.csv data files and put them in your `data` folder. You may have to right-click on each file to save as `.csv` files. The `darters.csv` file is a different format than `darters.txt` from a previous assignment but the data is otherwise the same.
- Load the `tidyverse`, `here`, `knitr`, and `smatr` libraries.
- Remember to format your code properly. Review the notes from assignment 08 if necessary.
- Commit early. Commit often. Push regularly but at least push your completed assignment.

Importing Data

As a reminder, The `here` package automatically sets all search paths relative to the project file. We're keeping our files to import in a folder called `data`. Thus, the general format you should use is

```
my_data <- read_csv(here("data", "my_file.csv"))
```

`here` tells `read_csv` to look for your file in the `data` folder. This is a simple and portable solution across all computers. **Use this format for this and all future assignments.**

Source your functions

You can use R's `source()` function to run R code contained in an external file. Source your `my_functions.R` script that you wrote in the previous assignment by including this code chunk. Use the same format you do with your data folder and `here()` but use your `scripts` folder instead. You'll have to change the name of `my_functions.R` if you did not use that name for your scripts file.

Darters in riffles

Import data

- Read in `darters.csv`. These are the same data as an earlier assignment but in csv format.
- Use `filter()` to remove `tetrazonum` and `zonale`.
- Do *not* delete the `minsub` column like you did in the previous assignment.

Wrangle data The substrate was measured at each trap site. The relative proportion of the two most commmon substrate types (sand, fine gravel, small gravel, large gravel, and cobble) was estimated. The relative abundance of the most abundant substrate type was recorded as `majsub`. The less abundant substrate type was recorded as `minsub`. Together, `majsub + minsub = 1`. If they do not sum to 1, then there was a data recording error.

- Use `mutate()` to perform these wrangling tasks
  - Sum together the `majsub` and `minsub` into a new column called `total_substrate`. Does `total_substrate` sum to 1 for all observations?
  - Change the riffle values of `1` and `2` to `Riffle 1` and `Riffle 2`.
  - Change the length data from centimeters to millimeters. It is more common to use mm for small fishes.
  - Change the sex values of `f` and `m` to `Female` and `Male`.
- Only `total_substrate` needs a new column. The other changes should use their own columns.

Summarize data Use `group_by()` and `summarize()` to summarize the mean length, depth, and velocity for each species for each riffle. *Save this summary to a different object* than the original data. You will need both in the following plot.

Table results

- Use `kable` to make a table with the summary means.

Graph data Plot 1

- Make a plot that uses the `stat_summary()` technique shown in R4ds section 3.7, reason 3 (scroll down, just before 3.7.1 Exercises) but use `x = species` and `y = length`.

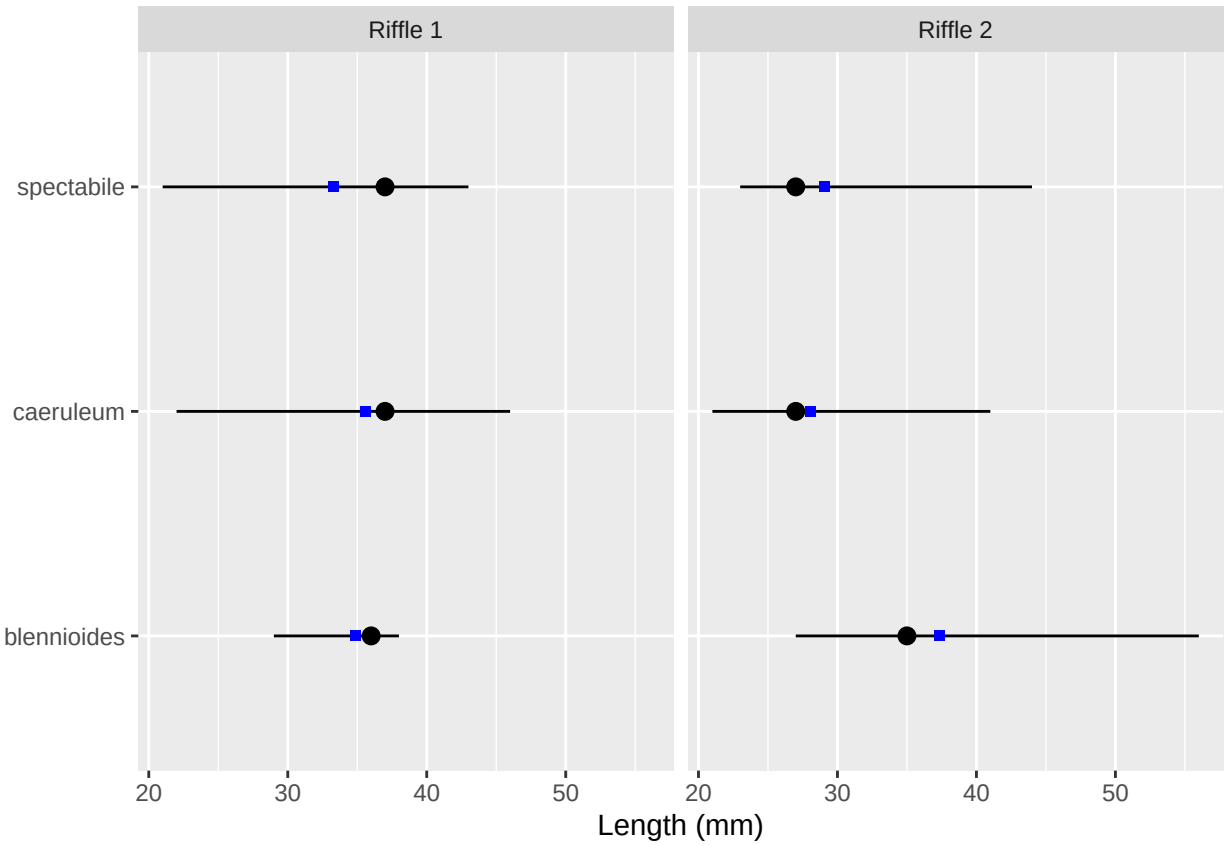
  This will plot a point to show the median length, and draw a line from the shortest (`min`) to the longest (`max`) length to show the range of lengths for each species.

- Remember that `ggplot2` uses a "layered grammar of graphics." Add a `geom_point()` layer to add the mean length of each species to the plot. Use a filled square shape. *You will have to use the summarized data set to plot the mean for this geom.*

Note: If you do not get a point in the next plot showing the mean for *E. blennioides,* then you did something wrong. What was it? Find your error and fix it.

- Turn the plot so that the species are listed on the left axis and length on bottom axis.
- Add `facet_wrap()` to separate the two riffles.
- Add `labs()` so that length includes the appropriate unit of measurement in parentheses and that the species axis is not labeled (`x = NULL`).

How does the relationship of the median, mean, and range change between the two riffles? In other words, how does the size distribution change between the two riffles? A correct plot looks this this.

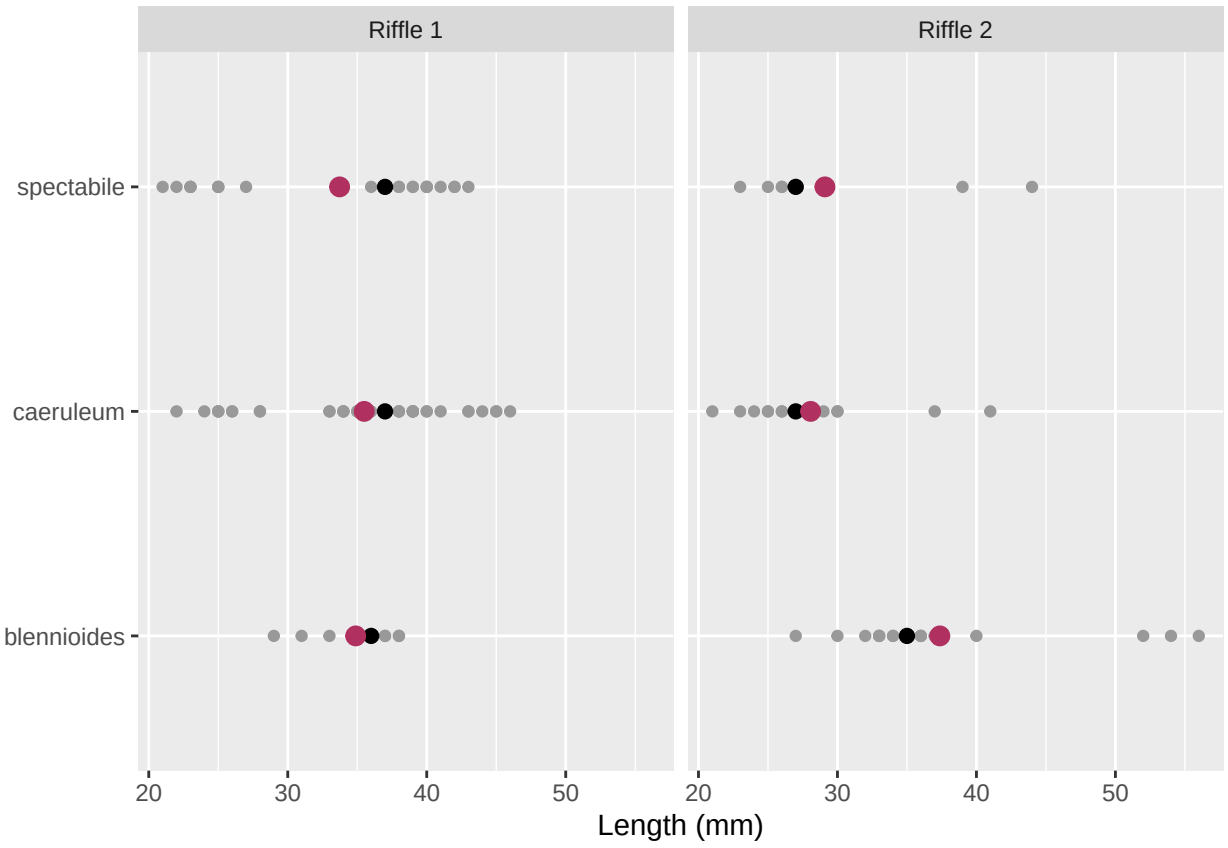


Plot 2

You probably noticed that the location of the median is not centered between the minimum and maximum values, even though the median is the middle value.

You decide that you could display the data more effectively. You want to show the individual measurements and emphasize the mean over the median. You also decide you want to try a slightly different way of graphing the data.

- Make a new plot with a `geom_point()` layer of the length variable for each species. Change the point color to a lighter shade in the range of `gray50` to `gray70`. The points should contrast against the overall gray background but they are not the main visual focus of the graph.
- Add a `stat_summary()` layer with these arguments:
  - `fun = median`
  - `geom = "point"`
  - `size = 2.2`
  - Do not include `fun.min` or `fun.max` arguments.
- Add another `stat_summary()` with the same arguments but change the function to `mean`, size to 3, and add the `color = "maroon"` argument. If you get a really large point, then you didn't quite follow the instructions. Read them again carefully and fix it. Your maroon point should be just a bit larger than the other points.
- Facet, label, and flip the graph as you did above. A correct plot looks like this.



Stress and corticosterones in birds

Beaugeard et al. 2018 (open access) studied the effects of stress from urbaniztion on House Sparrow (*Passer domesticus*).

Import data

- load the `smatr` library when you load `tidyverse`.
- Data file: `feathers.csv`.
- Tidy the data if and as necessary.
- The data were recorded by a team of French scientists, so decimal numbers use a decimal comma instead of decimal point (e.g., 2,31 instead of 2.31). Use the `locale = locale(<argument>)` argument in the `read_csv()` function to specify the decimal mark. Type `?locale` to read the help file. The only argument you need to supply to `locale()` is for the decimal mark.
- Their column names include spaces and units of measurement, generally not a good idea because you have to use backticks to reference the columns. Rename the columns when you import them, or rename them after you import the data using the `rename()` function. Use these names, in order from left to right:
  - capture_date, day, site, urban_score, cortF, cortB, sex, smi, tarsus, mass, basal_cort, stress_cort
- *Hint:* If you rename the columns when you read in the data, you will need to use the `skip` argument to remove the row of column names in the file.

Wrangle data The authors use an index called the "scaled mass index" (SMI), included as a column. You will use your `scaled_mass` R function to recreate that value. Recall that the equation to calculate SMI for each individual (i) is,

$$\text{SMI}_i = M_i \times (L_0/L_i)^b,$$

where L_0 is the mean tarsus length of all birds in the sample, M_i and L_i are the mass and tarsus[1] length of each individual, and b is the slope estimate of the standardized major axis regression on the log-transformed mass and log-transformed tarsus length.

Fortunately, you do not need to know the details of the analysis. You have the tarsus lengths and mass from the individual birds in your tibble. You need to calculate b, the slope of the regression. **Follow these steps carefully.**

- Calculate the mean tarsus length of all individuals in the data set, and store it in a variable called `mean_tarsus`. You should know how to calculate the mean of all values in a column.
- Use `mutate()` to log-transform (`log()`) the body mass (`mass`) and the tarsus length (`tarsus`). Save these in new columns as `log_mass` and `log_tarsus`, respectively.
- Run `major_axis <- sma(log_mass ~ log_tarsus, data = <data>)`. Substitute the name of your imported data for `<data>`. The `sma()` function calculates the standardized major axis regression.
- Apply the `summary()` function to `major_axis` to see the results of the regression.
  - The value you want for b is the slope of the estimate.
- Apply the `coef()` function to `major_axis`. The output is a named numeric vector with two elements. The value you want is the second element. Store that value in the variable `b`. **Do not hard code the value.** Review Homework 1, Part 2 to recall how to get values from a named vector. Make R do the work for you!
- Use mutate and your `scaled_mass` function to calculate SMI and store the values in a new column called `new_smi`. You need to pass the mass column, the tarsus column, and b to your function.
- Compare your new column with the column you imported. Your values should be nearly identical, differing by no more than 0.1. *Hint (optional):* Use the `select()` function of `dplyr` to rearrange your columns so that `smi` and `new_smi` are side by side. Or, print them out together. **Note:** If your numbers are not correct, figure out where you made a mistake and correct it.

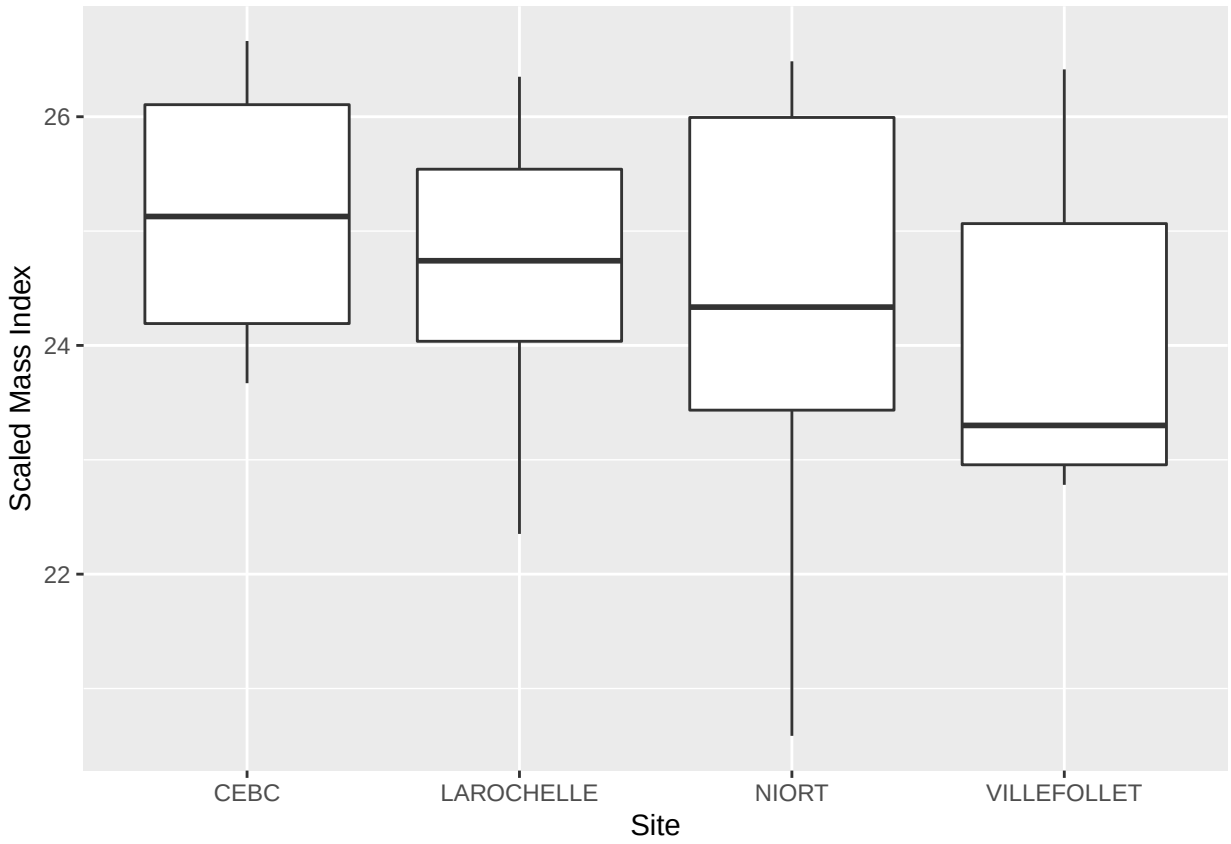
Summarize data Next, summarize the basal and stress corticosterone data.

- Group by site, and then use `summarize()` to calculate the mean and standard error of the mean (SE_Y) for `basal_cort` and `stress_cort` for each site. Use your `std_err` function to calculate standard error of the mean.

Graph data *Plot 1*

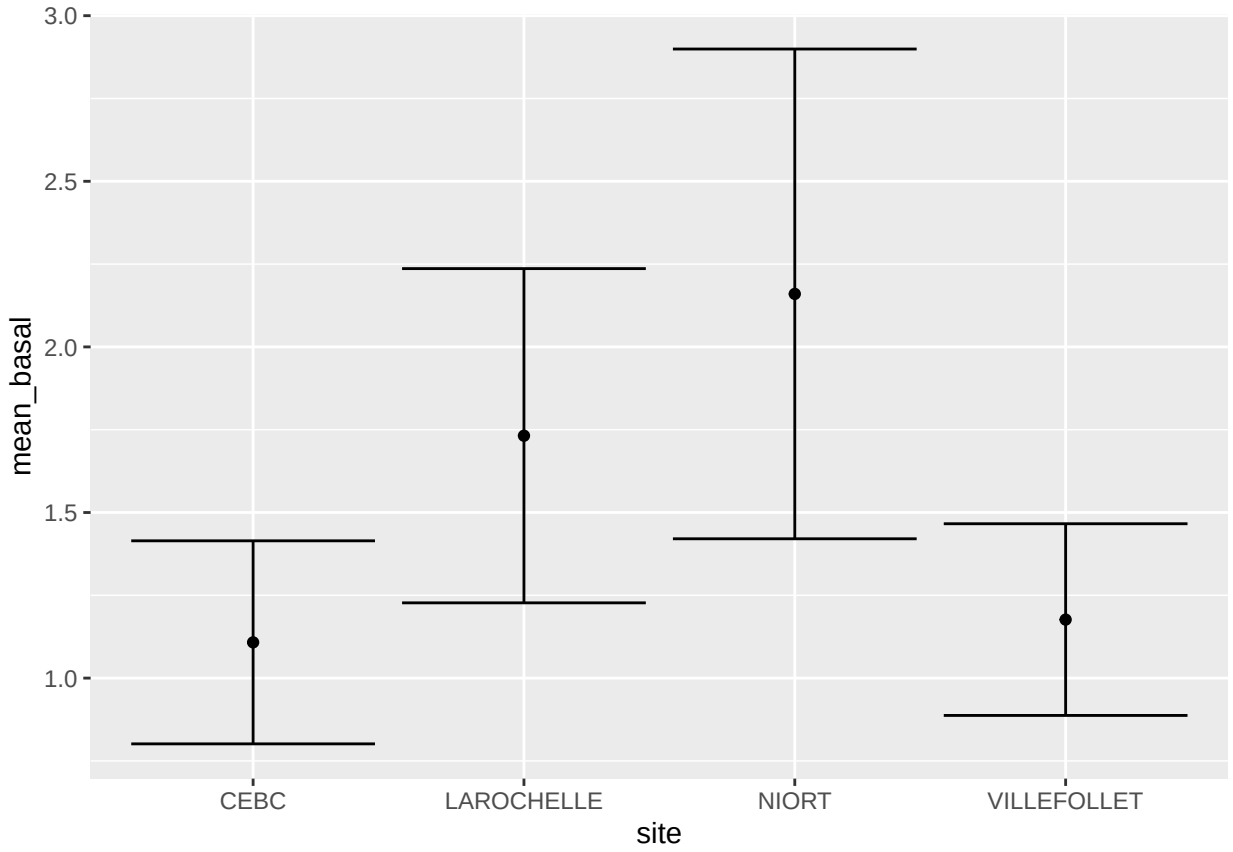
- Make a boxplot of `new_smi` for each site. Do you see any clear differences among the four sites?

[1] The tarsus looks like part of the bird's leg but is homologous to the tarsal bones in the vertebrate foot.



Plot 2 A common graph is to plot the means with error bars based on the standard errors of the means.

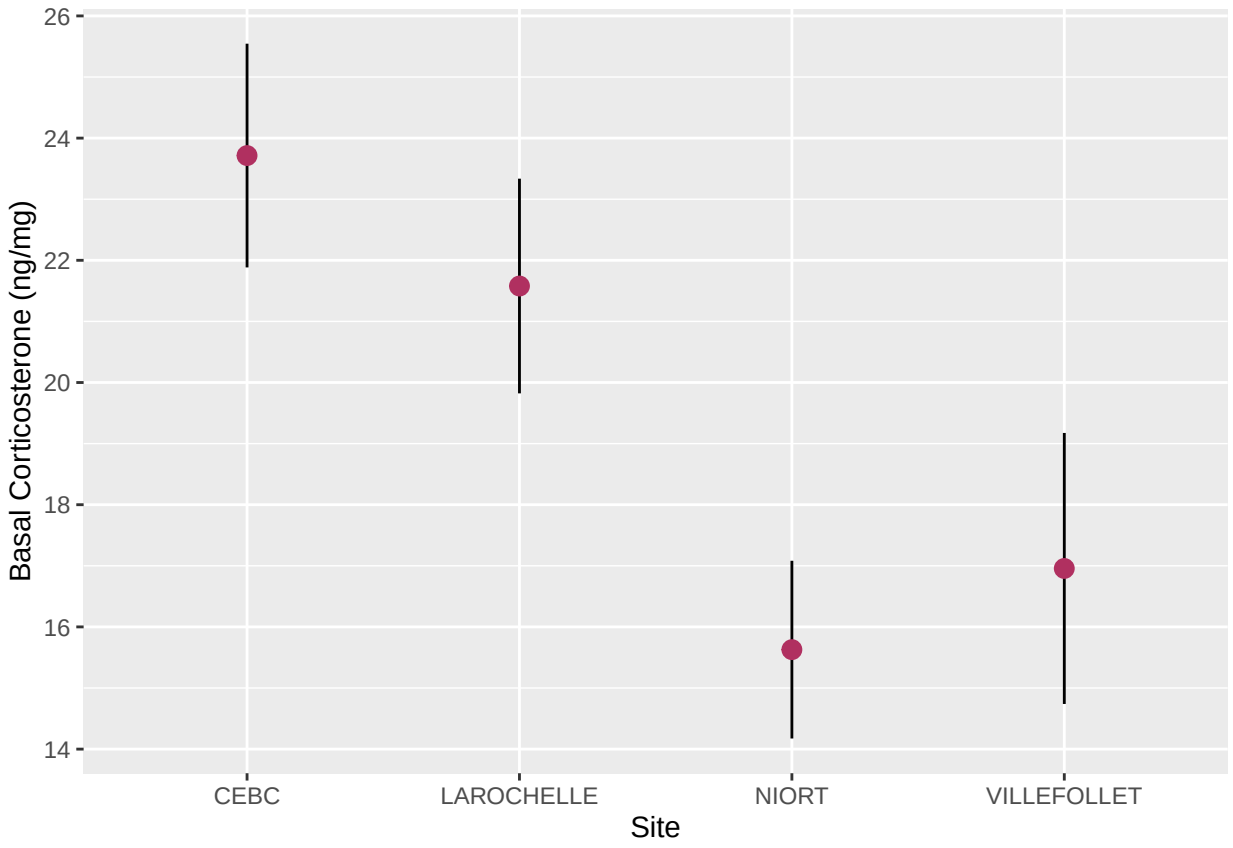
- Make a plot of the basal corticosterone concentration (ng/mg) for each site. Use `geom_point()`. Add a `geom_errorbar()` layer. Use the `?` function and the interwebs if you need, but here are a few hints:
  - x should be the sites,
  - `geom_errorbar()` needs `ymin` and `ymax` to draw the lower and upper ends of the error bars. Here, `ymin` should be one standard error below the mean and `ymax` should be one standard error above the mean. How could you do that? Good. Now do it.



Plot 3

I am not a fan of the Star Wars Tie Fighter look of the default error bars. The horizontal lines do not add any information about the data. In other words, they decrease the data-ink ratio.

Repeat the above plot but with two differences. Use the `mean_stress` and its standard error, and use `geom_linerange()` instead of `geom_errorbar`. Linerange and errorbar use the same arguments.



Graduate and Honors students You must modify one of the above graphs with the following requirements:

- use size of 3 for the point located at the mean,
- color the point maroon,
- place the error bars behind the points so that a black line does not cut across the face of each point.
- Assume you like the Tie Fighter look but you do not want them so exaggerated. Size them to something much smaller but still present. The exact size depends on your tastes. **Note:** I encourage undergrads to attempt this but this does not count for or against your grade.

---

et Vóila