Section 2: Mapping Occurrence Data with ggplot2

Maxine Cruz

04 April 2024

Here we will demonstrate how to plot species occurrence data on a ggplot2 map.

Learning Objectives

1. Be able to plot species occurrence data from GBIF.
2. Be able to plot occurrence data using `ggplot2`.
3. Understand the value of knowing where a species has been observed.
4. Understand and practice good file management.

Why Should We Care Where a Species is?

The occurrence data that we acquired from the Global Biodiversity Information Facility (GBIF) in the previous section (insert link to Section 1: Acquiring Occurrence Data) represents actual observations gathered by everyday people in society — which might even be your neighbor, doctor, or professor!

Using citizen science data for research gives us the opportunity to explore areas outside our current residence or domain, without all the travel expenses and paperwork. Citizen science data also improves citizen engagement in research and the openness of information. A popular site for submitting observations is iNaturalist, and contributing to the data collection is as easy as snapping a photo and uploading it through the app. GBIF is a collection of many data sources — including iNaturalist — but the data here has typically been passed as "research-grade", and this is one of the reasons why we are using GBIF as opposed to iNaturalist.

You may have gathered from the data acquisition that there are a number of variables that get recorded with an observation, with two of the most important information being the date and location of the observation. Having the date and location for each observation helps us to analyze trends in the data, such as where a species occurs most frequently or how their population numbers have fared over the years. It is difficult to assess where these locations are just by looking at a data table with rows and rows of values, so we visualize the data on a map. A map makes both our and others' lives easier when it comes to interpreting where a species has been observed.

This tutorial will demonstrate two options for mapping. One version will be using `ggplot2`, and the other will use `leaflet`. The former is a static mapping option and creates visualizations by "adding" together different components, whereas the latter is a dynamic and relatively easy way to create more realistic-looking maps. In this section, we will practice plotting using `ggplot2`.

Necessary packages

This section of the tutorial requires the following R packages to assist in our mapping:

- **`ggplot2`:** Creates static graphics, such as maps.

We can install and use them by running the following lines:

```r
# Install packages (Note: This only needs to be done once!)
install.packages("ggplot2")
```

Afterwards, let us read these libraries into R so we can use them in the plotting process:

```r
# Load dependencies
library(ggplot2)
```

Great! Now we have the tools to move forward in this section.

Preparing the Data

Load Our Data

Just like before, let us say you exited the program after saving the raw data and lost the objects in your R Environment. And now you are continuing this tutorial but need the data in there again. Thankfully you do not need to go through all that data acquisition again (assuming you did save the file) and all we need to do is read that .csv back in and call it `data`:

```r
data <- read.csv("data/cleaned_data.csv")

# Data currently has: 389 observations, 10 variables
```

We can remind ourselves what this data set contains by using the `head()` function to view some of it:

```
##   year month day  latitude longitude              species speciesKey  kingdom
## 1 1900    11  NA -29.85417   31.0500 Egybolis vaillantina    1782645 Animalia
## 2 1929    10  NA  -3.21490   40.1218 Egybolis vaillantina    1782645 Animalia
## 3 1930    11   1  -3.21490   40.1218 Egybolis vaillantina    1782645 Animalia
## 4 1930    12   7  -3.21490   40.1218 Egybolis vaillantina    1782645 Animalia
## 5 1936     9  NA  -3.21490   40.1218 Egybolis vaillantina    1782645 Animalia
```

Note that each observation (or at least a majority of them) has a latitude and longitude. Those coordinates will be used when plotting our occurrence maps. However, if an observation is missing the latitude and/or longitude then those observations will not be plotted — neither us or R will know what the missing values truly are. We omitted observations with an NA latitude or longitude in the previous section, so that should have been taken care of.

Mapping with ggplot2

The ggplot2 vignette (link in Additional Resources) will tell you there are seven customizable parts in plotting a chart, and at least three are needed at minimum: the data, mapping, and layer. In our case:

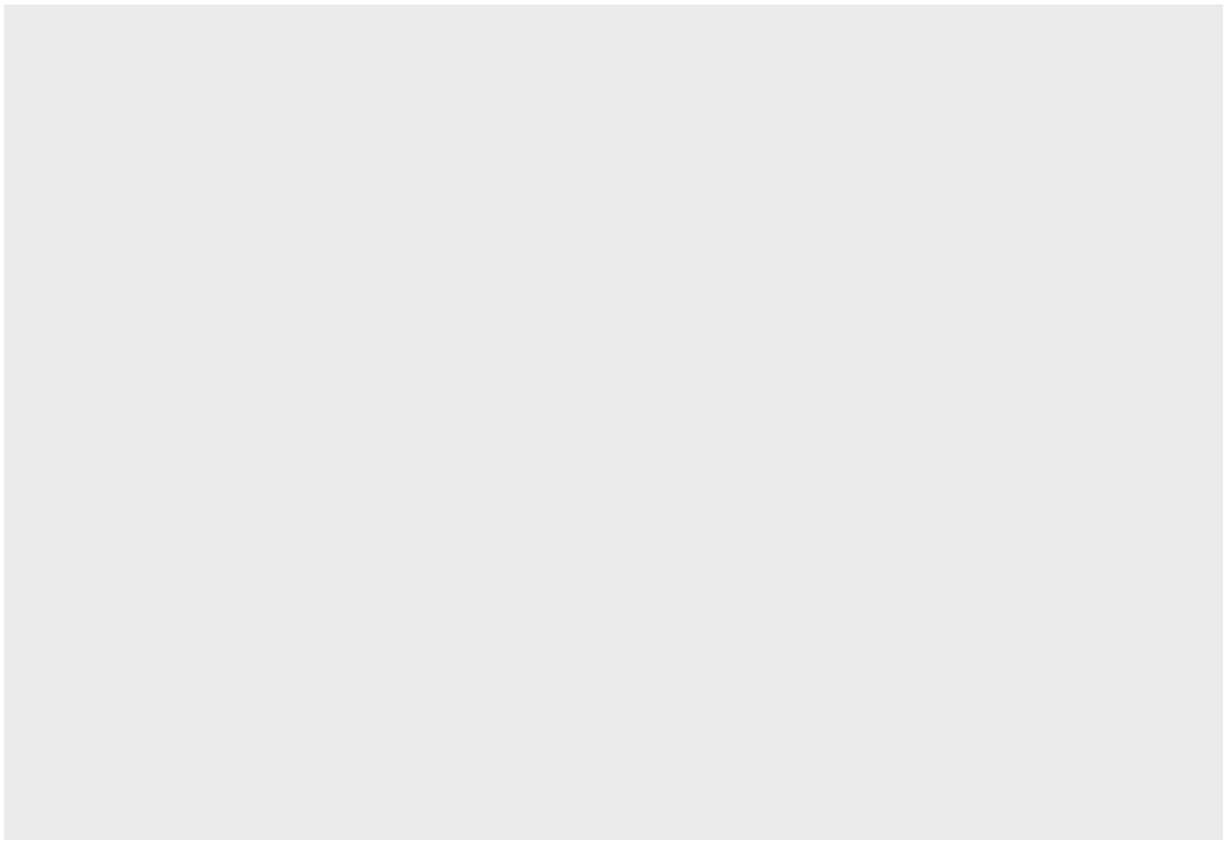
- ***Data***: We will be plotting our species occurrence data collected from the previous section — that of the African Peach Moth (*Egybolis vaillantina*).
- ***Mapping***: These are the instructions for how this data is going to be plotted on the map, which will be through aes(). The "aes" is short for "aesthetic". Our map will have longitude on the x-axis and latitude on the y-axis, so we will need to specify that.
- ***Layer***: These arguments determine how the data will be displayed. Since we are interested in plotting each individual observation at the location they were observed at, we will be using a function to add points to our map. Fittingly, this will be done using geom_point().

Let us try forming our ggplot2 map by seeing what the addition of each component does.

i) The ggplot() Function by Itself

Plots from ggplot2 typically start with ggplot(). What happens when you run that by itself is this:

```
# Base function
ggplot(data = data)
```

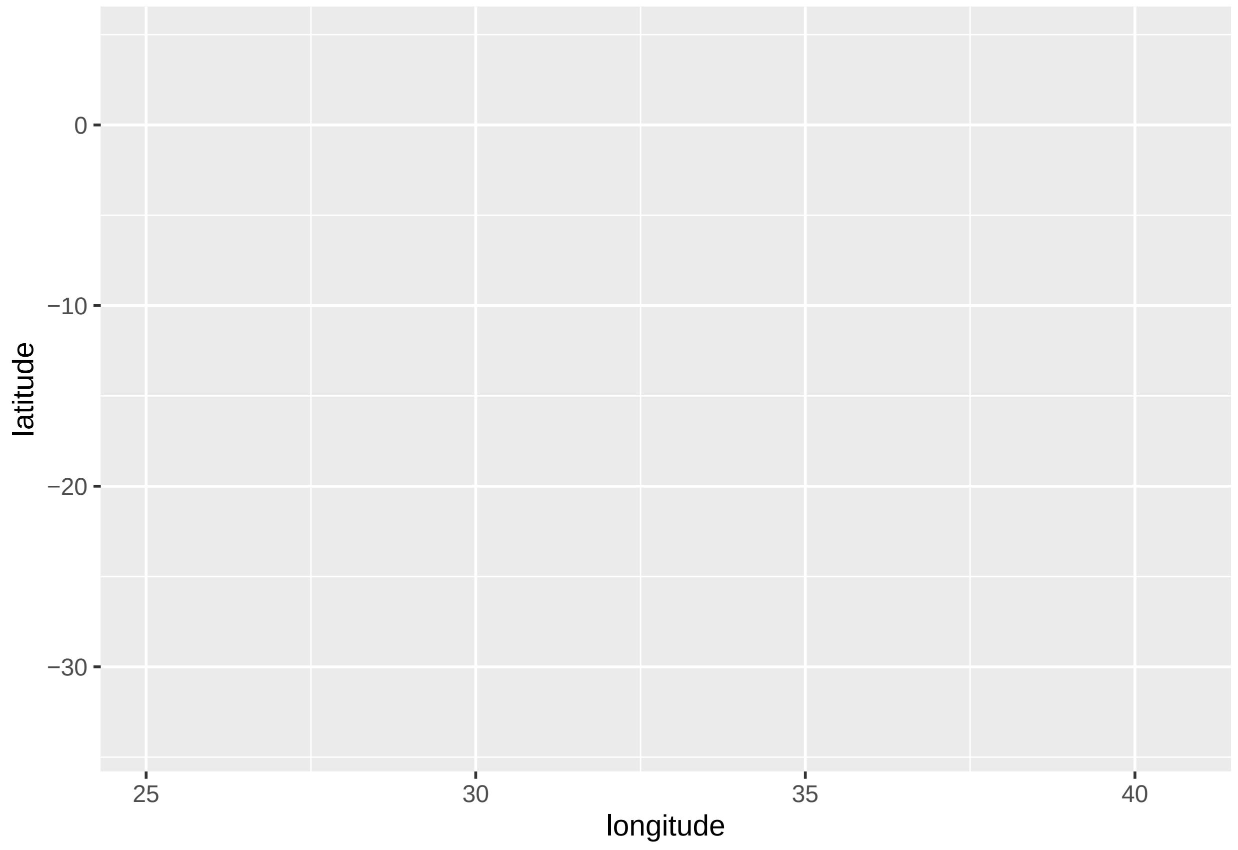


Not much going there yet, but we can change that! (P.S. If you see a grey box above, that is exactly what is supposed to happen.)

ii) Adding the Mapping Component

To know where our species occurrences need to be plotted, we need an axis on the graph. So within `ggplot()`, we will need to add the `aes()` function. This is a little different compared to the other components in that it is within the base function and not added on (as you will see later). Remind yourselves that for geographic coordinates, the lines running horizontal (left to right, —) on the Earth are called longitude and those running vertical (up and down, |) are latitudes. This would translate to the longitudes being on the x-axis of a Cartesian plane, and the latitudes on the y-axis:

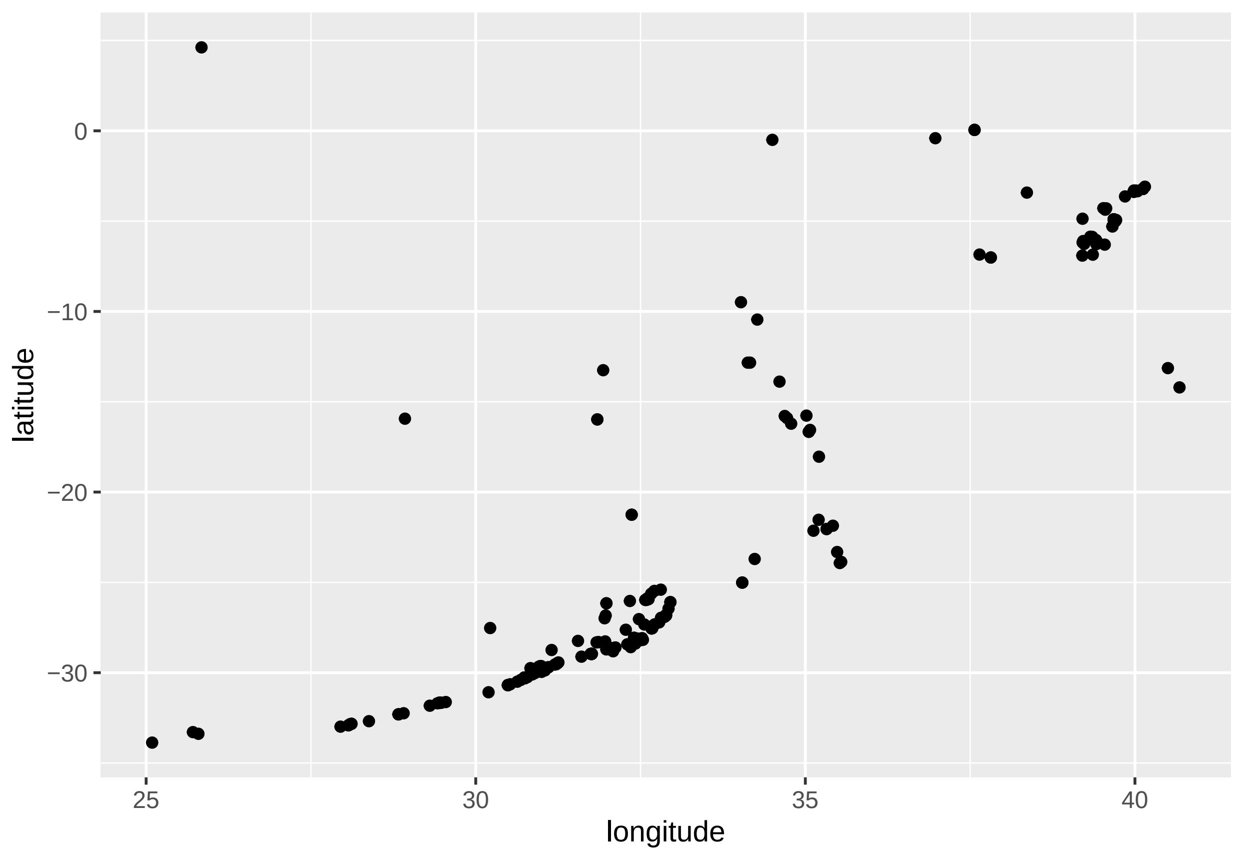
```r
# Base function + define mapping
ggplot(data = data, aes(x = longitude, y = latitude))
```



iii) Adding the Occurrence Points

Great! Now we have an area to plot the latitude and longitude associated with each of the African Peach Moth observations. To add points to this graph, we will add on `geom_point` to the existing code. Note that adding this layer requires a "+":

```r
# Base function + define mapping
ggplot(data = data, aes(x = longitude, y = latitude)) +
  # Add occurrence points
  geom_point()
```



Note: There is an alternate route to take when plotting the occurrences. Rather than having the data be passed through `ggplot()`, we can have it inside `geom_point()`. The code for that case would look like this:

```r
# Base function + define mapping
ggplot() +
  # Add occurrence points
  geom_point(data = data, aes(x = longitude, y = latitude))
```

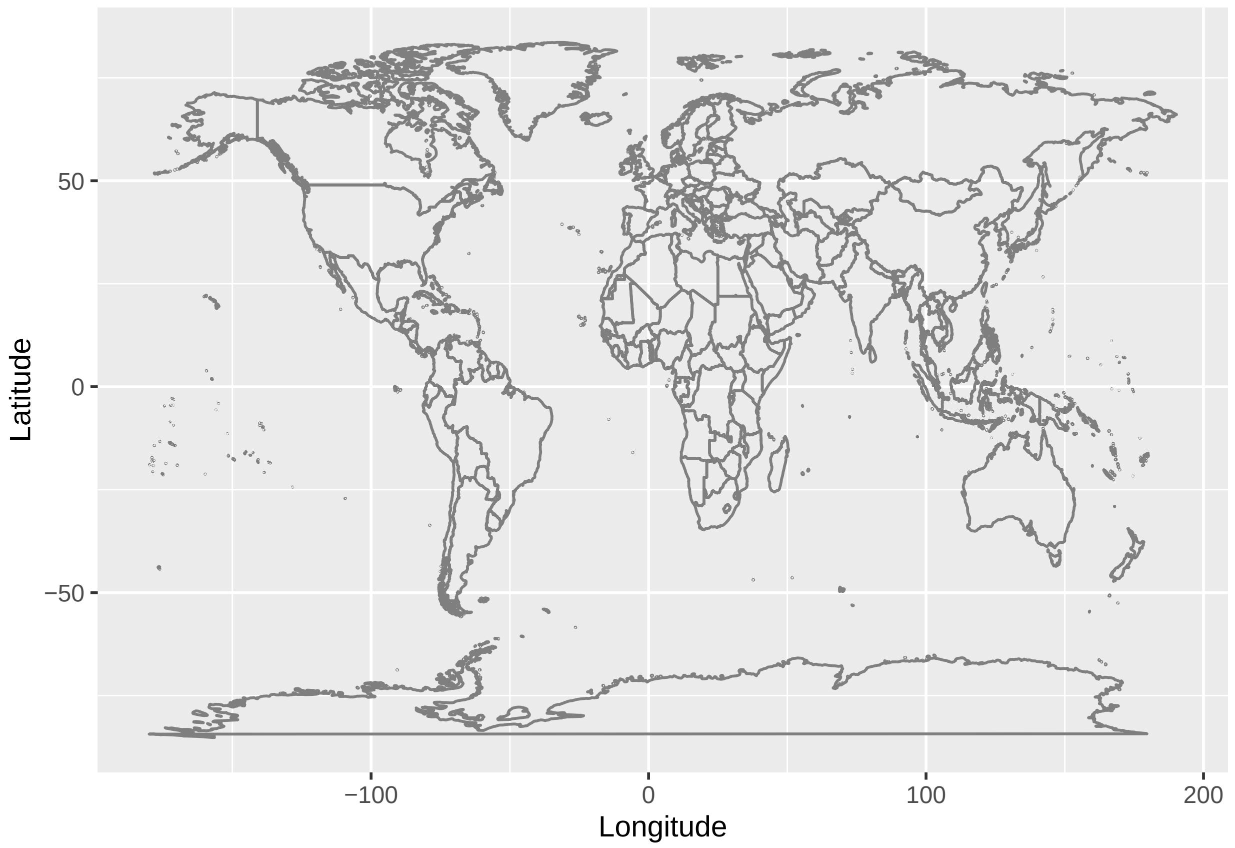
The benefit to this is it groups our data to the function that plots the occurrences, as opposed to the base plotting function. Either way works, as you can see, but if we were to plot multiple and different data sets on the map then clustering each set to their respective `geom_` would make the code a little more organized.

iv) Adding Country and Province Boundaries

So we have our occurrences on the map… but it is hard to tell where in the world they are, isn't it? To solve this, we will pull out some world map data that is provided by `ggplot2` through a function called `map_data`:

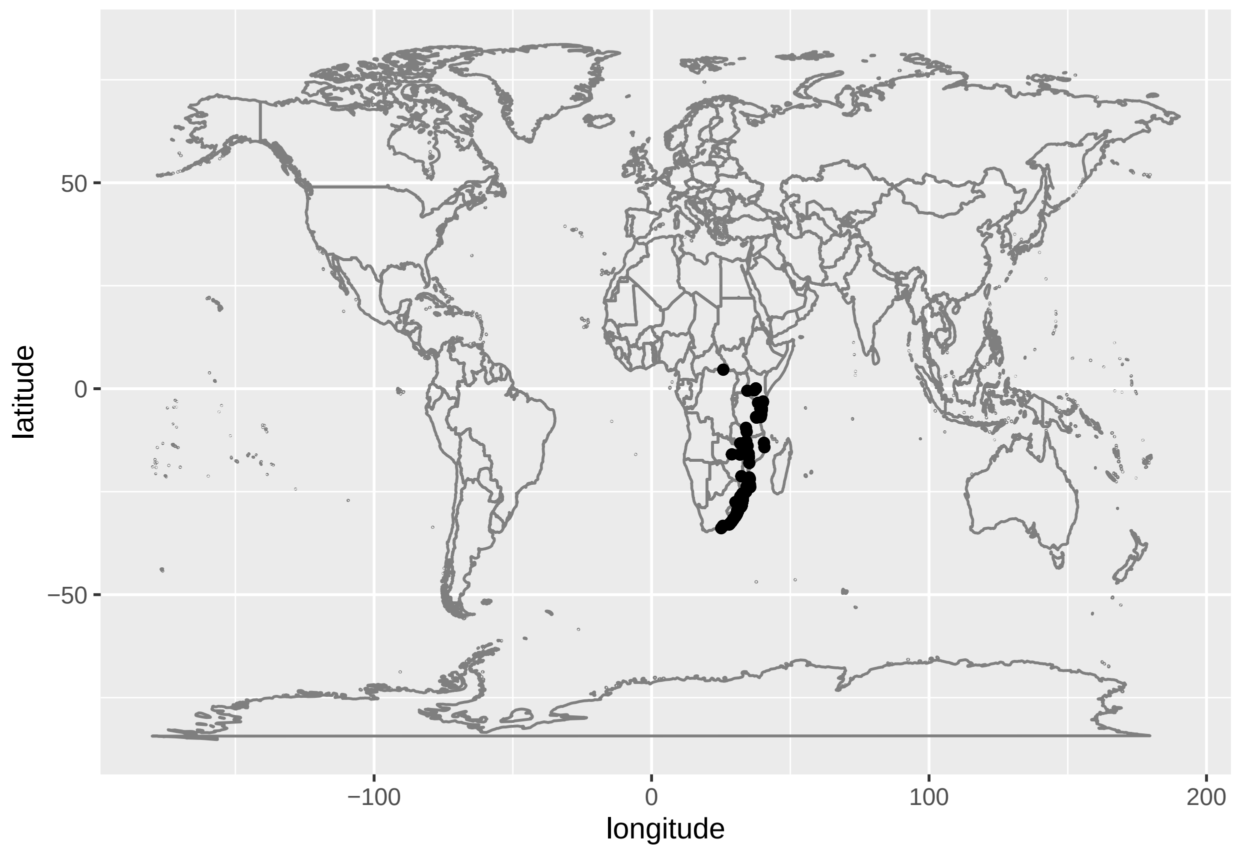
```r
# Get world map boundaries
world_map <- ggplot2::map_data("world")
```

Of course, you might wonder what this data even looks like. And to that I present this visualization of that data we just gathered:



Now that that is settled, we can add boundaries to our moth map. This is done by adding `borders("world")` to the code. But note that we will be adding it before `geom_point()`. This is so that the borders layer is *under* the occurrence points layer:

```r
# Base function + define mapping
ggplot(data = data, aes(x = longitude, y = latitude)) +
  # Add world boundaries
  borders("world") +
  # Add occurrence points
  geom_point()
```



v) Adjusting Axis Limits

It would be beneficial to us and our peers if the observation points were easier to see, since those are what we are interested in after all. We can make the limits of the latitude and longitude axes hone in to where the observations are plotted by determining what the maximum and minimum values for each coordinate are. To do this, we will use `max()` and `min()` to find the outermost coordinates. Those values will then be used to construct a box around all the observations, to which we will add a buffer so that the plot does not cut off abruptly at those maximums and minimums. For the buffer, let us add 5 degrees to each maximum side of the box and subtract 5 degrees to each minimum:

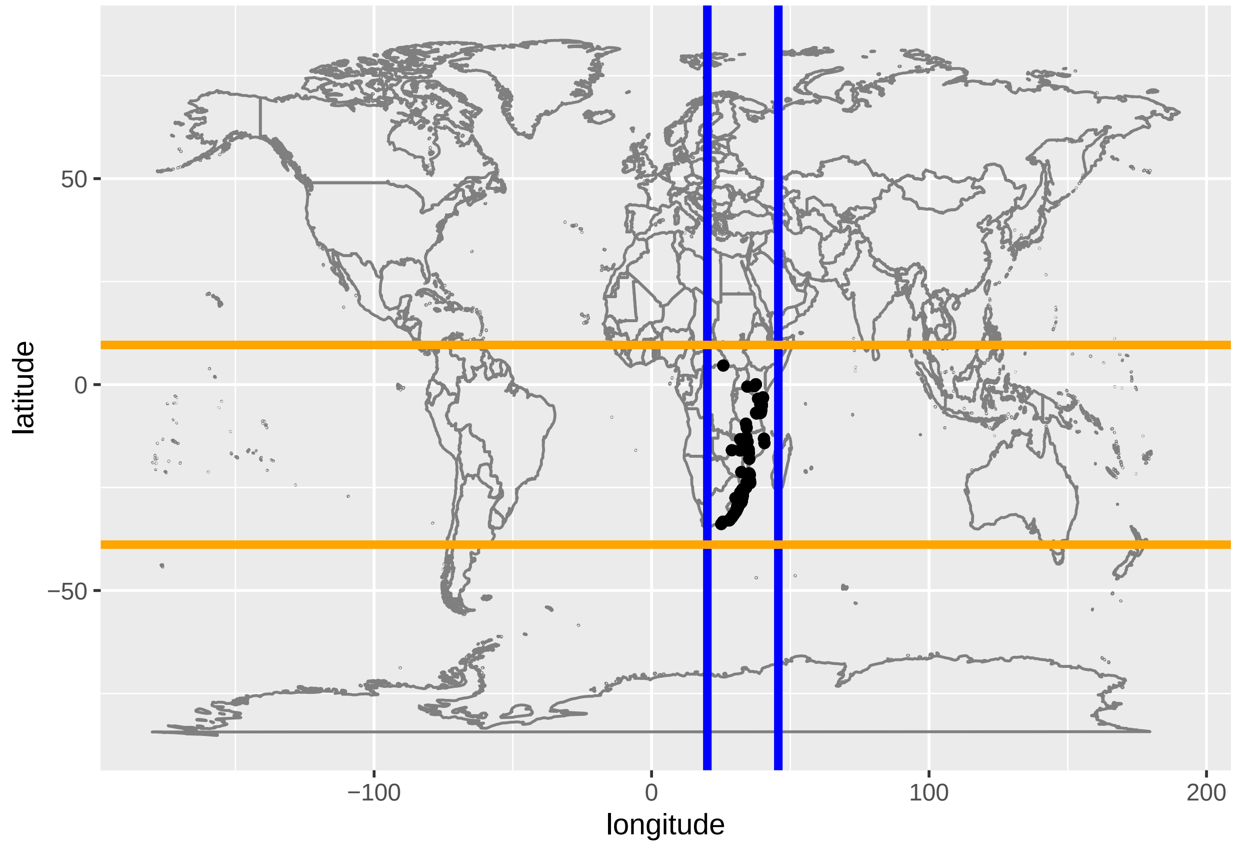
```r
# Find northern-most latitude (this will be the top of our box)
top <- max(data$latitude) + 5

# Find southern-most latitude (bottom of box)
bottom <- min(data$latitude) - 5

# Find eastern-most longitude (right side of box)
right <- max(data$longitude) + 5

# Find western-most longitude (left side of box)
left <- min(data$longitude) - 5
```

Note that southern and western coordinates here are represented by negative values. This differs from what might be more commonly associated with an image of a globe, where the hemispheres are differentiated by

"N", "S", "E", or "W" while the degrees from the equator / prime meridian remain positive. Viewing the values we attained above, we would see the following:

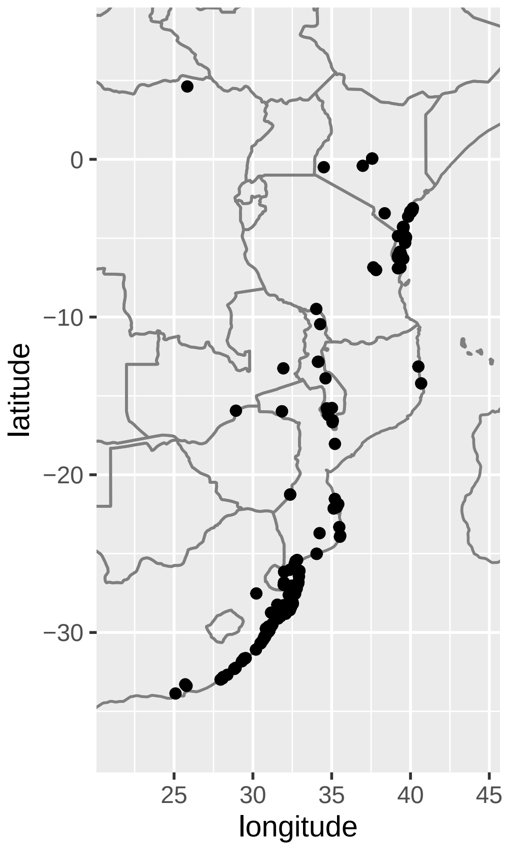
```
##      north     south     east    west
## 1 9.619151 -38.86877 45.67643 20.0913
```

And plotting this bounding box on the map prior to cropping will show:



Now that that is taken care of, we can adjust the axes by adding our limits to arguments in `coord_fixed()`.
:

```
# Base function + define mapping
ggplot(data = data, aes(x = longitude, y = latitude)) +
  # Add world boundaries
  borders("world") +
  # Add occurrence points
  geom_point() +
  # Specify axes limits
  coord_fixed(xlim = c(left, right),
              ylim = c(bottom, top),
              expand = F)
```



vi) Making it Look Pretty

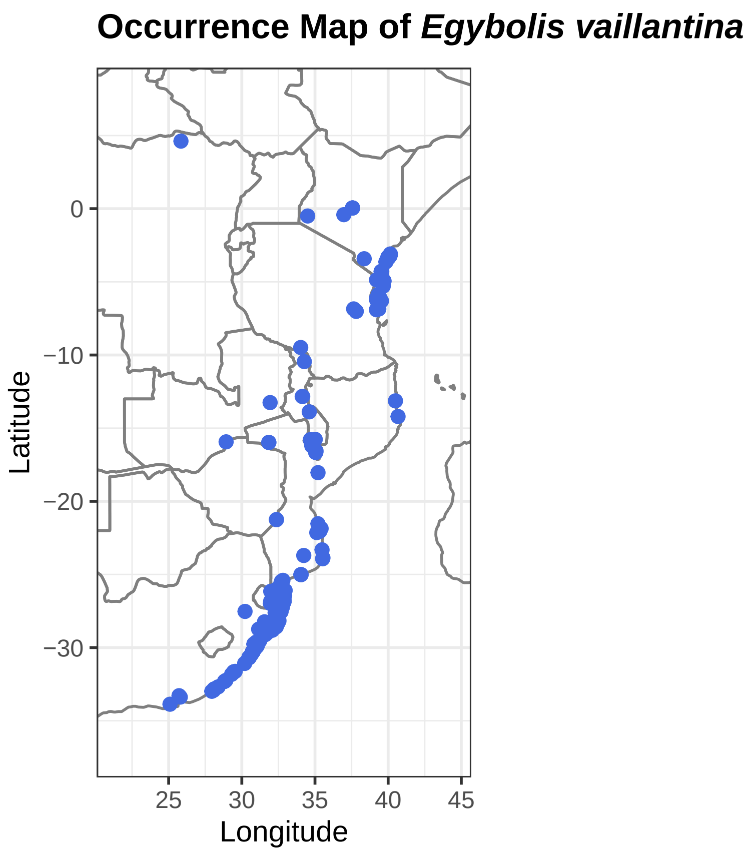
Wonderful! Now that we have a good foundation of a map, we can make it look a little jazzier. Of course, you could go down an entire rabbit hole of customizations and find yourself checking the time two hours later only to ask "At what cost?". I am sure you would then look at your map and think about how glorious it is and think all that time was not wasted (or not… depending on your styling choices). Feel free to do some exploration, but here I will present some modifications to the following:

- ***Axis labels*****:** If we were presenting this or adding it to a scientific paper, it would be a good idea to make sure grammar and capitalization of the content is reviewed to have that professional and publication-ready look. Changing the axis labels can be done with `xlab()` and `ylab()`, but here we will compound it under one function (`labs`) that also takes care of our title.
- ***Theme*****:** Adding a `theme_` component will change the overall look of the plot. Mostly the background and grid lines in this case, but in other types of plots there may be changes to the design and colors as well.
- ***Occurrence points*****:** We can change the color and size of our points on the map. It is also possible to change the points into other shapes, but I think I like to keep them as dots for this map.
- ***Title*****:** If you handed this map by itself to someone and left them alone, they might want to know what they are looking at. Of course, this is solved by adding a title. But I am sure you knew that. Below you will notice that the title has some additional code to it compared to the axes labels — `bquote()`, `paste()`, and `bold()` / `bolditalic()`. If our title was just "Occurrence Map of the African Peach Moth", then the extra code is not needed and we can follow the format used for the axes (`title = " "`). However, since species names are typically italicized we will want to represent it that way in our title.

- **`bquote()`:** This function will evaluate font specifications, which in our case is making certain words bold and/or italic.
- **`paste()`:** Takes different "strips" (also called "strings") of characters and puts them together. Here we have a string with "Occurrence Map of" and another with " Egybolis vaillantina". The function will take those two and turn it into "Occurrence Map of Egybolis vaillantina".
- **`bold()` / `bolditalic()`:** These are functions that... well make characters bold and/or italicized.

What do these look like in action? We can take a look:

```r
# Base function + define mapping
ggplot(data = data, aes(x = longitude, y = latitude)) +
  # Add world boundaries
  borders("world") +
  # Add occurrence points
  geom_point(color = "royalblue",
             size = 2) +
  # Specify axes limits
  coord_fixed(xlim = c(left, right),
              ylim = c(bottom, top),
              expand = F) +
  # Capitalize axis labels and add a title
  labs(title = bquote(paste(bold("Occurrence Map of"),
                            bolditalic(" Egybolis vaillantina"))),
       x = "Longitude",
       y = "Latitude") +
  # Change theme
  theme_bw()
```



Yay! We have a reasonably nice-looking plot of our occurrence points.

vii) Saving the Map

We did a lot of work to build that map! Now if we wanted to export it to presentations and documents, we should save it to our output folder. The `ggplot2` package has a function to do just that, called `ggsave()`. We will need to store our map in an object first, which we will call `gg_map`:

```r
# Store our map in an object
gg_map <- ggplot(data = data, aes(x = longitude, y = latitude)) +
  borders("world") +
  geom_point(color = "royalblue",
             size = 2) +
  coord_fixed(xlim = c(left, right),
              ylim = c(bottom, top),
              expand = F) +
  labs(title = bquote(paste(bold("Occurrence Map of"),
                            bolditalic(" Egybolis vaillantina"))),
       x = "Longitude",
       y = "Latitude") +
  theme_bw()

# Save our map
ggsave(filename = "output/egybolis_occurrence_map.png",
       plot = gg_map,
       width = 15,
       height = 25,
       units = "cm")
```

It is possible to save the map under certain dimensions to fit your needs, but make sure the ratios roughly follow that of the map (e.g. saving this map under a longer width than height will stretch it horizontally). I like to use centimeters as the image units because it is easier to visualize how large the outcome would be, but feel free to use pixels or inches.

Practicing good file management and naming conventions is important for many reasons, one of which being that it makes it easier to find your files in the first place. Here I chose to name the map as "egybolis_occurrence_map.png" which I think is something most people (or me in the distant future!) will be able to decipher if they were to go through my work. The prefix to that, "output/…", places that file in the output folder since this is a product of our manipulations with the data. There are many different ways to organize your files, so do what works best for you!

Additional Resources

- Best practices for file management: MIT Communication Lab | File Structure
- The `ggplot2` package has a lot of customization settings and visualizations other than a map. Learn more about it here: Introduction to ggplot2
- R has a sea of color options. You can even make your own! Check out this site: Colors in R
- If you are unsure about the accuracy of your map, you can always check the map provided on GBIF for your species. Searching up the species of your choice should give you a visual map of all the occurrences (before any data processing) alongside other information on the organism.

Next Section: [link to next lesson]

For questions, comments, or concerns (or even tattoo design requests) please email tmcruz@arizona.edu.