Biodiversity vs Pollution: How Our Lives Impact the Ocean

Ainsley Smith

4/19/2025

Contents

Introduction to the Topic

Within our world holds thousands upon thousands of creatures. They have watched as our world and technology has changed, as they were here long before us, yet are taking the brunt of the effects of our evolution. They cannot keep up with the constant evolution, which is leading to their physical decline. Our biggest impact on the animals of the world is our damage to their habitats. Whether the damage is to the forests or to the oceans, the impact is all the same. It is very well known that environmental effects, such as pollution, cause harm to everything in its path. Specific chemicals can cause adverse shifts to the ocean water and air, which in turn, hurts the animals. But just how bad has this damage gotten? Of course pollution is a visible affect, but what about the species and their habitat? As a landlocked state, we are not in direct eyesight of trash-filled coasts and pieces of plastic floating up on shore, so we might not be as aware of just how bad it is.

This analysis delves into the specifics of the marine animals and the changes in their environment over the years, and how these changes are becoming even more detrimental to everything around them. The main questions of this analysis are as follows. Where do animals lie in lieu of trash from pollution, is it in their main habitat? What types of trash is found within the water? Within the species, what is the breakdown of their Red List categories, is there an ocean with the biggest amount of endangered species? What type of affect has general pollution and human disturbances caused to the ocean, specifically within the pH? Lastly, what type of pattern can be found when modelling a species population trend and its primary threats together? This type of path allows a thorough breakdown of the habitat and animals, the outputs of trash, and the harm done to the water, which in turn harms the animals even more. All of this coming together to answer the question of how pollution and our presence has affected the marine creatures.

Part 1: Trash

The Search for Datasets: Trash Part 1

I started off by finding my trash datasets. At first I found an ocean trash occurrence map that was generated by code, however when I went to plot the longitudes and latitudes, it was obvious that it was completely random and did not properly outline any countries like I expected. After going back to the drawing board, or rather scouring the web for other ocean trash occurrence datasets, I found a marine microplastic concentration map from the National Centers for Environmental Information. The NCEI is a branch under te National Oceanic and Atmospheric Administration, well known as NOAA. As the help page of the map states, each record indicates where at least one microplastic concentration was recorded. The concentration levels were a visual feature that was sadly not picked up within the export of the data, no matter what I did to retrieve it. However, I felt as though simply having the occurrences and mapping the latitudes and longitudes would still help achieve the goal. 90% of marine life is found within about 230 kilometers, or about 143 miles from land. This is where most life is physically able to thrive due to the abundance of sunlight, which provides for rich biodiversity in the region. This is why finding out where trash lies in lieu of where species live can bring light to the potential harms species may face when it comes to trash.

The sample collected from my query spanned from the beginning of 2013 to the end of 2023. This range was selected as I was able to find similar ranges within other datasets, which will be discussed later on. It resulted in a csv of 12,514 records and 22 variables. While looking at the csv from the query, I also noticed the Oceans were their own column. This was a good categorical feature that could be included within the visualizations and next graphs.

Data Wrangling For Figure 1

I first started by filtering out junk variables that would provide no purpose to my analysis, and that would only slow the code down. These variables included URLS and long strings, such as the references of the record. Any record that did not possess a value for the ocean variable was also filtered out. This created the final filtered trash table of 12,243 records and 14 variables.

```
#reading in the trash file
trash <- read.csv("C:/Users/ats20/OneDrive/Documents/mycode/final/datasets/realtrash2013.csv")
```

```
#filtering out the junk columns
#filtered to take out any point that does not have an ocean listed, these were
#missing a lot of data in general
trashfilt <- trash %>% select(OBJECTID : Density.Class, Latitude : Date, x : y) %>%
      filter(Oceans != "")
```

Once this was done, creating the visual aspect for the oceans was next. Using aesthetics and changing the colors of the points would result in a very hard to read graph, so in the end, adding a blocking of the oceans using geom_rect was the best choice. To do this, I had to find the maximum and minimum latitude and longitude for each ocean. This was done by filtering each ocean, and using summarize to retrieve the maximums and minimums. The Pacific Ocean required two different queries for each side of the ocean, as doing them together would result in an improper finding. Initially, in my trials, I graphed these as is, which resulted in some unwanted overlap. These queries below were the final result of finding out the proper values to piece together to make the most cohesive map, so any oceans with specific values missing rely on other oceans values for its border. Adding these oceans into the trash occurrences enriches the visual to help provide information on species oceans that suffer the most from pollution.

```
#To create all of the oceans for the graph, we find the maximum and minimum
#latitudes and longitudes for each ocean to piece together into rectangles
#For the Left side of the Pacific Ocean
trashfilt %>% filter(Oceans == "Pacific Ocean") %>%
      summarize(Xmax = max(Longitude[Longitude < 0]), Ymax = max(Latitude), Ymin = min(Latitude))
```

```
##       Xmax  Ymax      Ymin
## 1 -67.2688 66.22 -59.61187
```

```
#for the right side of the Pacific Ocean
trashfilt %>% filter(Oceans == "Pacific Ocean") %>%
      summarize(Xmin = min(Longitude[Longitude > 0]), Ymax = max(Latitude), Ymin = min(Latitude))
```

```
##       Xmin  Ymax      Ymin
## 1 99.13861 66.22 -59.61187
```

```
#for the Southern Ocean, the Pacific ocean ymin is used as the ymax, -90 for ymin,
#and -180 to 180 for xmax and xmin

#for the arctic ocean
trashfilt %>% filter(Oceans == "Arctic Ocean") %>%
      summarize(ymin = min(Latitude[Latitude > 0]))
```

```
##      ymin
## 1 61.0145
```

```
#indian ocean - ymin is the pacific ymin so it matches up
trashfilt %>% filter(Oceans == "Indian Ocean") %>% summarize(xmin = min(Longitude),
      xmax = max(Longitude), ymax = max(Latitude))
```

```
##     xmin     xmax    ymax
## 1 21.006 146.8219 27.7306
```

```
#atlantic ocean, split into two parts for the sake of shape, found the minimum lat
#of the gulf by scanning through the points by hand
trashfilt %>% filter(Oceans == "Atlantic Ocean") %>%

      summarize(xmin = min(Longitude), xmax = max(Longitude), ymax = max(Latitude))
```

```
##       xmin     xmax    ymax
## 1 -98.0994 35.10167 66.1741
```

```
#atlantic part two: using the border of the Indian ocean for xmax, xmin found by
#the min latitude, and ymax is atlantic ocean ymin
trashfilt %>% filter(Oceans == "Atlantic Ocean") %>% summarize(ymin = min(Latitude))
```

```
##        ymin
## 1 -59.50325
```

With the information above, a dataframe was created for each ocean. When accounting for the multiple sections of the ocean, the variable simply holds the numbers in proper order according to each section. So all the numbers in the second position align together. These rectangles will create the smallest bit of overlap at the top with the Arctic Ocean. It was too hard to decipher where to cut it to match up nicely. I ultimately decided to keep it because some species could be counted as the Pacific, for example, and another species close beside it is counted as the Arctic.

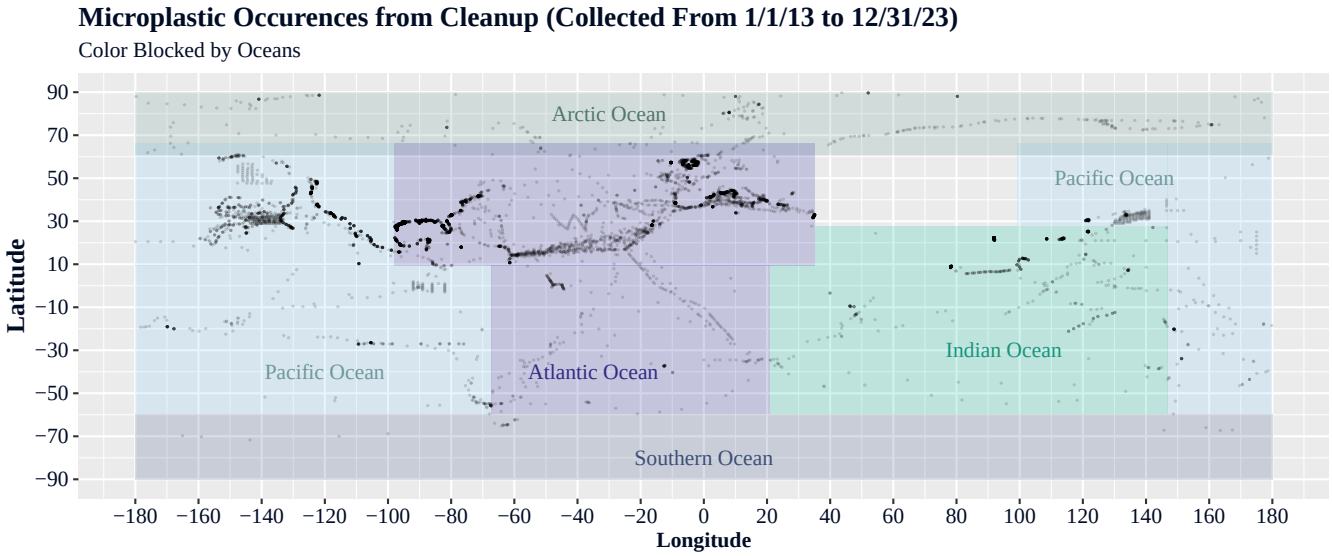
```
#Each variable represents an ocean that will be shown on the graph
#Some oceans, like the Pacific, have multiple rectangles, this is to stop overlap
#creating the dataframes beforehand is also for ease for the two graphs that use these
Pacific <- data.frame(
      xmin = c(-180, -180, 99.13861, 146.8219),
      xmax = c(-98.0994, -67.2688, 146.8219, 180),
      ymin = c(9.4197, -59.61187, 27.7306, -59.61187),
      ymax = c(66.22, 9.4197, 66.22, 66.22)
)
Southern <- data.frame(xmin = -180, xmax = 180, ymin = -90, ymax = -59.61187)
Arctic <- data.frame(xmin = -180, xmax = 180, ymin = 61.0145, ymax = 90)
Indian <- data.frame(
      xmin = c(21.006, 35.10167),
      xmax = c(35.10167, 146.8219),
      ymin = c(-59.61187, -59.61187),
      ymax = c(9.4197 ,27.7306)
  )
Atlantic <- data.frame(
      xmin = c(-98.0994, -67.2688),
      xmax = c(35.10167, 21.006),
      ymin = c(9.4197, -59.50325),
      ymax = c(66.1741, 9.419)
)
```

The Product: Trash Occurences

This is where everything is combined to create the final graph. Each ocean is assigned a different color, with the opacity knocked down low to not make it super difficult to read. The longitudes and latitudes are plotted with geom_point, then a theme (seen in each graph following this one) is applied for cohesiveness. Lastly, to make it clear as to what each rectangle represents, annotate is used to label each ocean by the most barren x and y coordinate as to not overlap any points.

```
ggplot() + #creating the rectangles for the ocean
  geom_rect(data = Pacific, aes( xmin = xmin, xmax = xmax, ymin = ymin, ymax = ymax),
      alpha = 0.2, fill = "lightskyblue" ) +
  geom_rect(data = Southern, aes( xmin = xmin, xmax = xmax, ymin = ymin, ymax = ymax),
      alpha = 0.2, fill = "#405990") +
  geom_rect(data = Arctic, aes( xmin = xmin, xmax = xmax, ymin = ymin, ymax = ymax),
      alpha = 0.2, fill = "#6B9B95") +
```

```
geom_rect(data = Indian, aes( xmin = xmin, xmax = xmax, ymin = ymin, ymax = ymax),
    alpha = 0.2, fill = "#0AC5A0") +
geom_rect(data = Atlantic, aes( xmin = xmin, xmax = xmax, ymin = ymin, ymax = ymax),
    alpha = 0.2, fill = "#3128A5") +
geom_point(data = trashfilt, aes(x = Longitude, y = Latitude), size = 0.1, alpha = 0.15) +
#the theme for making each graph cohesive, provides a font to each element, fixes everything
#to a readable size, and provides color to these titles and labels.
theme(legend.position = "none", axis.text = element_text(color = "#010C25", family = "serif",
    size = 11),
    axis.title.x = element_text(family = "serif", face = "bold", color = "#010C25"),
    axis.title.y = element_text(family = "serif", size = 13, face = "bold", color = "#010C25"),
    plot.title = element_text(size = 14, family = "serif", hjust = 0, face = "bold",
    color = "#010C25"),
    plot.subtitle = element_text(family = "serif", hjust = 0, color = "#010C25")) +
#these breaks provide nice tick marks for the lats and longs, and labels them accordingly
scale_x_continuous(breaks = seq(-180, 180, by = 20), labels = seq(-180, 180, by = 20)) +
scale_y_continuous(breaks = seq(-90, 90, by = 20), labels = seq(-90, 90, by = 20)) +
labs( #labeling everything to help with clarity
    title = "Microplastic Occurences from Cleanup (Collected From 1/1/13 to 12/31/23)",
    subtitle = "Color Blocked by Oceans"
) + #labeling each ocean so it is further clarified what it represents.
annotate("text", x = -120, y = -40, label = "Pacific Ocean", color = "#6E989C",
    family = "serif") +
annotate("text", x = 0, y = -80, label = "Southern Ocean", color = "#3C4F77",
    family = "serif") +
annotate("text", x = -30, y = 80, label = "Arctic Ocean", color = "#507570",
    family = "serif") +
annotate("text", x = 95, y = -30, label = "Indian Ocean", color = "#18947B",
    family = "serif") +
annotate("text", x = -35, y = -40, label = "Atlantic Ocean", color = "#312B83",
    family = "serif") +
annotate("text", x = 130, y = 50, label = "Pacific Ocean", color = "#6E989C",
    family = "serif")
```



The graph above shows that the majority of microplastic concentrations are found very close to the coast. This could be simply because of sampling bias, and that it is much easier to collect the samples next to the coast, nonetheless, its most prominent around land mass. This could be brought down to beaches, boating, and also runoff from rivers. Another notable feature is the large mass of recordings to the left of the US. This is due to the Great

Pacific Garbage Patch, which is a mass of floating debris spanning from the coordinates 135°W to 155°W and 35°N to 42°N. This is roughly 1.6 million square kilometers. Its' makeup majorly consisted of microplastics, despite its size, with 92% of the mass being made up of larger objects. The same can be said about the long strand of recordings to the right of the US. This is known as the North Atlantic Garbage Patch. Though not as heard of or as notable as the Great Pacific Garbage Patch, it is thought to be just as large, if counting the microplastics that have started to go deeper within the ocean over the years. This visual provides a basis of the conditions species may potentially live in, with microplastics found all over the ocean.

The Search for Datasets: Trash Part 2

As the last graph accounted for microplastics, the same stress needs to be applied to bulkier materials. Finding out the general composition of the trash is equally as important as finding out where the majority of it lies. Learning exactly what type of chemicals could be leaching out from materials could be a potential approach as to helping regulation and protecting species. To do so, I found a coastal cleanup site that is volunteer based, called Trash Information and Data for Education and Solutions, or TIDES for short. It is a public data system containing the world's largest ocean trash dataset. This data system has years of data, in the similar span as the previous data, from 2015 to the present day. Some reports that were available included a top ten item summary and a total item summary. There was an option for a detailed summary report, however, there were errors in downloading it.

Data Wrangling For Figure 2

The one of most use was the total item summary. It provided the total quantities of cleanup from the land, underwater, and watercraft, and contained the quantities of each item found within the all the combined years. Even though this data was from volunteer input, it was the most polished and gone through set out of all of them, with no oddly named variables causing issues. There was a total of 62 types of items listed, but 18 of them were items that were no longer tracked. These were removed from the dataset as to give a more accurate listing as to what is currently tracked today. I removed the land column from the table as well, and fixed the counts of the total items and percent of total columns as needed to represent the new dataset.

```
trashSummary <-
      read.csv("C:/Users/ats20/OneDrive/Documents/mycode/final/datasets/TrashSummary.csv")

#filtering and fixing the old columns to account for the new table
#This fixed the percentage and the totals
trashSummary <- trashSummary %>% mutate(Total.Items = Total.Items - Land) %>%
      mutate(rowId = row_number()) %>% filter(rowId < 45) %>%
      select(Items, Underwater:Total.Items) %>%
      mutate(PercentOfTot = round((Total.Items / 587828) * 100, digits = 2)) %>%
      filter(PercentOfTot != 100)
```

The Product: Trash Types by Quantity

For this graph, I went for a lollipop graph to switch up something that could have been done as a bar graph. This is simply graphing the totals of each item to show the spread of the most found types of trash and the lesser found items. Geom_segment reads in the length from 0 to the end, or the quantity of that type of trash. In this case, the quantity is logged to shorten the scale. It would have been done regularly, except the spread of values was too large, which is why it required the log10 for readability. The coordinates are also flipped for ease of reading each type of trash. I didn't make any labels to compensate for the log10 scale to show the true values, since that wasn't the main goal of this visual, which was more to show the breakdown of the types of trash.

```
#using line segments and points from 0 to the quantity number to make the stick and lollipop
ggplot(trashSummary, aes(x = reorder(Items, log10(Total.Items)), y = log10(Total.Items))) +
  geom_segment(aes(xend = Items, y = 0, yend = log10(Total.Items)), color = "#97C3C9",
      linewidth = 1.5) +
  geom_point(size = 3.5, color = "#233588") +
```

```
coord_flip() + #the theme
theme( axis.text = element_text(color = "#000000", family = "serif", size = 9.5),
    axis.title.x = element_text(family = "serif", face = "bold"),
    axis.title.y = element_text(family = "serif", size = 13, face = "bold"),
    plot.title = element_text(size = 14, family = "serif", hjust = 0, face = "bold"),
    plot.subtitle = element_text(family = "serif", hjust = 0)) +
labs( #the labels
  y = "Logged Quantity of Trash",
  x = "Types of Trash",
  title = "Types of Trash Found",
  subtitle = "By Quantity, From 1/1/15 to 12/31/23, Log10 Taken for Visualization"
  )
```

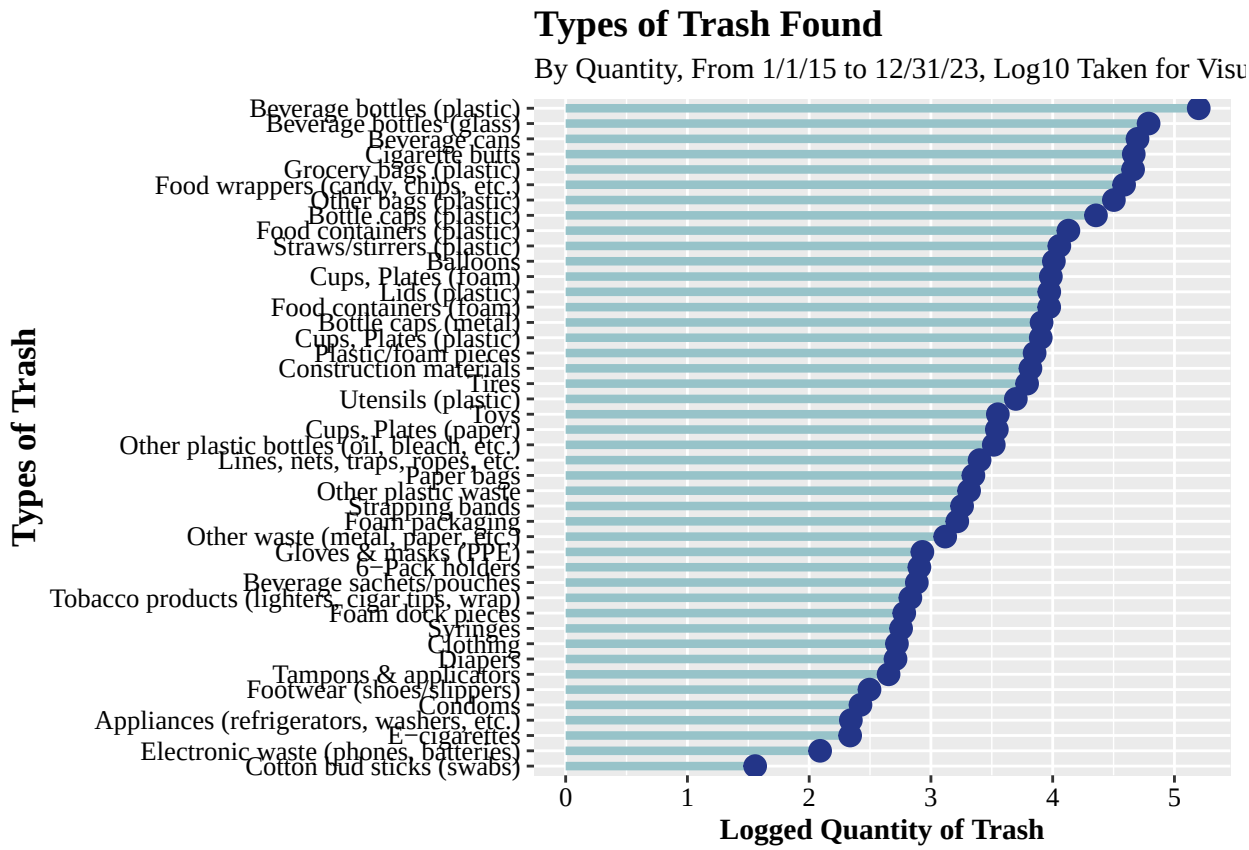


Figure 1: Code Result

As shown by the graph, the majority of trash found in the ocean is in fact plastic, with the leading item being plastic bottles. The unlogged number of the plastic bottles is 158,359. Another feature to mention is that cigarette butts are fourth place with a total of 46,340 found, which makes up for 7.8% of the total items in the water found. Types of chemicals in cigarettes include hydrogen cyanide, lead, and arsenic. These chemicals are notably terrible for humans, as they cause issues in the nervous and cognitive systems, and disorders and diseases. If a cigarette butt is mistaken for a small fish and eaten by a small creature, the effects that are already not good for human consumption are amplified and if not worse for the animal. Not only from consumption, but from chemicals leaching into the water can issues occur. A study done on this topic states that smoked butts with residual tobacco can be toxic to fish at levels of 1 butt per liter of water. Electronic items such as e-cigarettes, and appliances such as refrigerators have also been found, though lower on the list. In composition, they may be more metallic, but also chemicals such as Polypropylene and High impact Polystyrene are present. These are typically stable in its proper form, but when it begins to degrade, its affects are unknown (specifically for High impact Polystyrene). These polymers also make up a big portion of microplastics, especially polypropylene. Aside from all the toxicity leaching out from materials at different stages of decomposition, the sheer amount, as seen here, is the biggest issue. It must be remembered that this is only the spread from cleanup efforts over a rough ten year span, this does not predict the amount of trash still in the ocean in any way, therefore cannot fully show the dangerous materials being left around the animals.

Part 2: Animals

The Search for Datasets: Animals Part 1

When trying to find animal datasets, I initially searched through kaggle with absolutely no luck at finding exactly what I was looking for. Other official dataset websites were few and far between in terms of finding datasets for animals. That's when I remembered the official threatened species webpage, IUCN, that I heard of years ago. The International Union for Conversation of Nature's Red List of Threatened Species is the world's most comprehensive information source on the statuses of all forms of life. As stated on their about page, biodiversity is on the decline, with more than 47,000 species of plants, fungi, and animals threatened with extinction. Their goal is to inform and help bring action to this issue, along with providing open source data and querying pages for anyone to use. Through this querying page, I was able to filter through a wide variety of options to create my own dataset. I did not know what the results of my query would lead to, and within the first couple exports, the information was useless. It wasn't until my fifth or sixth try that I got a different csv for some unknown reason. This was a zipped file including three separate csvs called data points, assessments, and taxonomy. The data points were records of occurrences of the species within my query, taxonomy was the taxonomic breakdown of each species, and the assessments were the IUCNs' own conclusions, such as their Red List breakdown.

The exact query I performed and the reasons as to why is as follows. Taxonomy: Animals. This was to rule out plants and fungi. Within redlist categories, data deficient was excluded, as these lacked lots of information to begin with, and would result in lots of NA values within the final graphs. These would ultimately take away from what I wanted to find out from the data, though the information is interesting in itself. Every marine region and each habitat with marine in the name were also selected for the best chance to get all marine species. Lastly, each type of potential threat was selected to help specialize the results. Every other filter had nothing selected, which gave 7,356 species in the dataset. This definitely does not account for all marine life, as there are 243,613 accepted marine species as of 3/8/2025, however, this data is definitely the most inclusive, trust-worthy, and best-suited data for this analysis.

Data Wrangling For Figure 3

First, I decided to utilize the data points to map where species sightings occur in comparison to trash. As we found in the first graph, microplastics are highly centralized around the coasts, where animals are typically found. However, to confirm this pattern, it is key to assess where species have been specifically tracked. As this dataset has latitude and longitude variables, this can be easily visualized in the same fashion as the trash graph. However, this dataset does not contain any ocean variable, which will be needed for a future graph to answer where most at risk species lie. This was done by using the coordinates of the rectangles from earlier and assigning the different ocean name to when the case applies. Because of the sheer size of this csv, being 118,172 rows and 20 variables, I also filtered out long string columns that would be of no use for the analysis. This cut it down to ten columns.

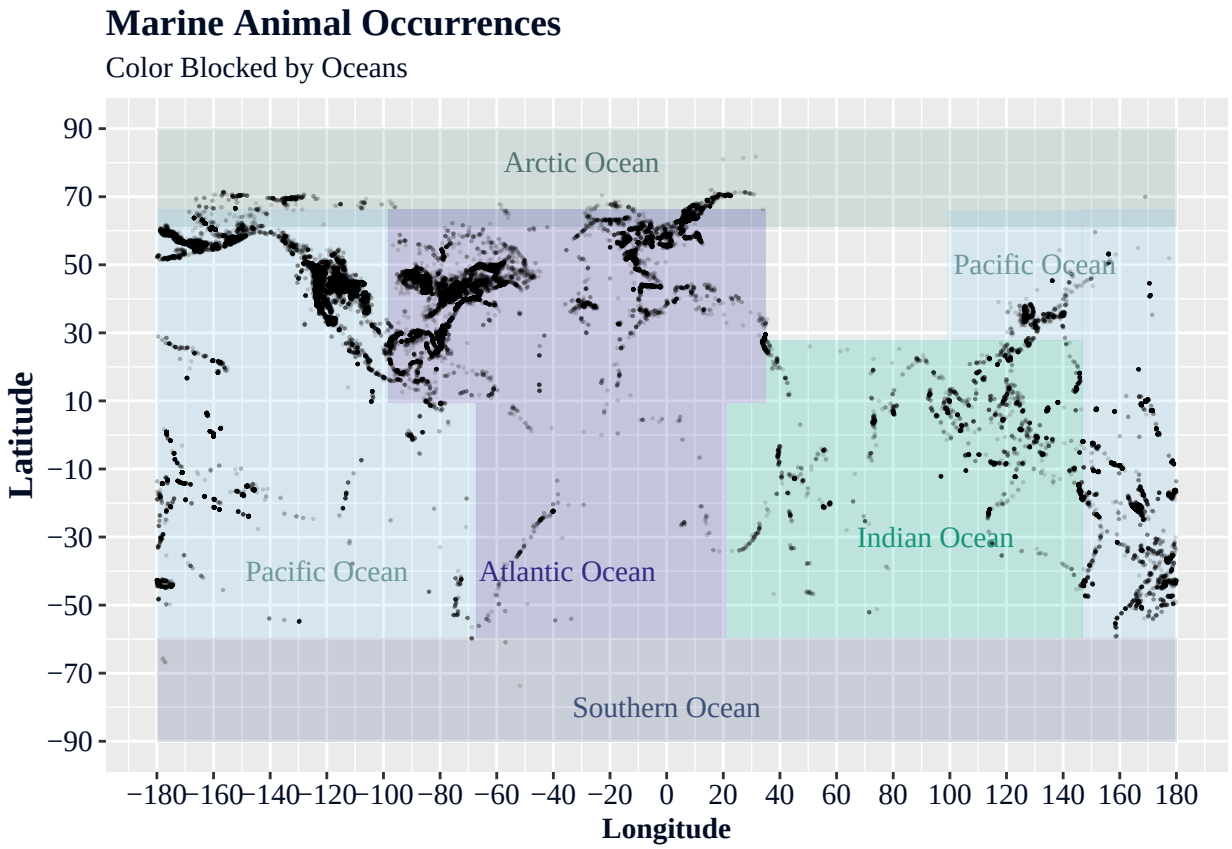
```
#animal occurrences
#we filter out all of the unimportant columns, and create a new column based off of our sections
#of oceans made previously. We will use these later
animalpts <- read.csv("C:/Users/ats20/OneDrive/Documents/mycode/final/datasets/points_data.csv")
#assigning a value per case if it matches the correct lats and longs with mutate and case when
animals <- animalpts %>% rename(assessmentId = assessment_id) %>% select(assessmentId : seasonal,
      legend, dec_long, dec_lat) %>%
  mutate(Oceans = "") %>%
  mutate(Oceans = case_when(
    dec_long >= -180 & dec_long <= 180 & dec_lat <= -59.61187 & dec_lat >= -90 ~ "Southern Ocean",
    dec_long >= -180 & dec_long <= 180 & dec_lat >= 61.0145 & dec_lat <= 90 ~ "Arctic Ocean",
    dec_long >= -180 & dec_long <= -98.0994 & dec_lat >= 9.4197 & dec_lat <= 66.22 |
      dec_long >= -180 & dec_long <= -67.2688 & dec_lat >= -59.61187 & dec_lat <= 9.4197 |
      dec_long >= 99.13861 & dec_long <= 146.8219 & dec_lat >= 27.7306 & dec_lat <= 66.22 |
      dec_long >= 146.8219 & dec_long <= 180 & dec_lat >= -59.61187 & dec_lat <= 66.22 ~
      "Pacific Ocean",
    dec_long >= 21.006 & dec_long <= 35.10167 & dec_lat >= -59.61187 & dec_lat <= 9.4197 |
```

```
      dec_long >= 35.10167 & dec_long <= 146.8219 & dec_lat >= -59.61187 & dec_lat <= 27.7306 ~
      "Indian Ocean",
    dec_long >= -98.0994 & dec_long <= 35.10167 & dec_lat >= 9.4197 & dec_lat <= 66.1741 |
      dec_long >= -67.2688 & dec_long <= 21.006 & dec_lat >= -59.61187 & dec_lat <= 9.419 ~
      "Atlantic Ocean"))
```

The Product: Marine Animal Occurrences

The code below is the exact same as the trash occurrence visual, but with the animal occurrence dataset instead.

```
#Using the same ocean rectangles as before
ggplot() +
  geom_rect(data = Pacific, aes( xmin = xmin, xmax = xmax, ymin = ymin, ymax = ymax),
      alpha = 0.2, fill = "lightskyblue" ) +
  geom_rect(data = Southern, aes( xmin = xmin, xmax = xmax, ymin = ymin, ymax = ymax),
      alpha = 0.2, fill = "#405990") +
  geom_rect(data = Arctic, aes( xmin = xmin, xmax = xmax, ymin = ymin, ymax = ymax),
      alpha = 0.2, fill = "#6B9B95") +
  geom_rect(data = Indian, aes( xmin = xmin, xmax = xmax, ymin = ymin, ymax = ymax),
      alpha = 0.2, fill = "#0AC5A0") +
  geom_rect(data = Atlantic, aes( xmin = xmin, xmax = xmax, ymin = ymin, ymax = ymax),
      alpha = 0.2, fill = "#3128A5") +
  #this is the same setup, just animal points dataset and the proper names of the
  #columns to follow
  geom_point(data = animals, aes(x = dec_long, y = dec_lat), size = 0.1, alpha = 0.15) +
  theme(legend.position = "none", axis.text = element_text(color = "#010C25", family = "serif",
      size = 11), #Theme, fixing fonts, colors, and sizes
      axis.title.x = element_text(family = "serif", face = "bold", color = "#010C25"),
      axis.title.y = element_text(family = "serif", size = 13, face = "bold", color = "#010C25"),
      plot.title = element_text(size = 14, family = "serif", hjust = 0, face = "bold",
      color = "#010C25"),
      plot.subtitle = element_text(family = "serif", hjust = 0, color = "#010C25")) +
  #Using scale to properly break and label the axes for readability
  scale_x_continuous(breaks = seq(-180, 180, by = 20), labels = seq(-180, 180, by = 20)) +
  scale_y_continuous(breaks = seq(-90, 90, by = 20), labels = seq(-90, 90, by = 20)) +
  labs( #labels
    title = "Marine Animal Occurrences",
    subtitle = "Color Blocked by Oceans",
    x = "Longitude",
    y = "Latitude"
  ) + #these label the oceans, the x and y situate it nicely on the graph
  annotate("text", x = -120, y = -40, label = "Pacific Ocean", color = "#6E989C",
      family = "serif") +
  annotate("text", x = 0, y = -80, label = "Southern Ocean", color = "#3C4F77",
      family = "serif") +
  annotate("text", x = -30, y = 80, label = "Arctic Ocean", color = "#507570",
      family = "serif") +
  annotate("text", x = 95, y = -30, label = "Indian Ocean", color = "#18947B",
      family = "serif") +
  annotate("text", x = -35, y = -40, label = "Atlantic Ocean", color = "#312B83",
      family = "serif") +
  annotate("text", x = 130, y = 50, label = "Pacific Ocean", color = "#6E989C",
      family = "serif")
```



As seen, the majority of the occurrences are around land, which confirms the information above. And with the pieces being incredibly small, as these were microplastic concentrations, this makes for hazards for the sea creatures. As with any small piece of plastic from a store, there are warnings all over the packaging, saying how its hazardous to small children for choking. The same logic can be applied to the ocean and the sea creatures who are even less aware of the dangers these materials can cause them if consumed or entrapped by them. Microplastics may be able to pass through properly without causing choking, however these materials can build up inside and ultimately cause blockages and starvation, eventually leading to death if not properly dislodged. Another thing to note in this is the large inland clusters, seemingly in the Great Lakes. As these were the findings of the query I did that only had "marine" selected, I kept them in, especially since those face similar conditions as to the ocean, especially with the population of people that go to the lakes for tourism. The only difference being freshwater instead of saltwater.

Data Wrangling For Figure 4 and 5

Since figures 4 and 5 were quite similar in nature, the wrangling was done together. The goal was to determine the specific breakdown of Red List categories, especially within each ocean. First, the assessments csv was read in, as this contained the required information, and then I found the primary key, which was the assessment ID. I then joined the points data to the assessments data using a left join, which duplicated all the lines in the assessments data. This was the desired outcome as each assessment applied to each point in the points data.

For the specified wrangling, I started by creating two different datasets to hold seen species and unseen species. This separation occurred from the fact that only 268 species actually came up in the points data, with the other 7,090 not found or spotted. This is why is.na() is used, to collect these differences. For the seen animals, I filtered out all of the duplicates of a species, as I only wanted to graph the species per ocean, not all animals per ocean. So once I got everything proper, I grouped by Ocean and then the Red List category and summarized each line to get one line per ocean per category, then used summarize to display the sum of the species per ocean. For the unseen species, I simply grouped by the red list category and summarized each line with count representing the sum of points. Also, I created labels for the x axis, as I knew I wanted the true number shown, regardless if I took the log of the counts for readability.

```r
assessments <- read.csv("C:/Users/ats20/OneDrive/Documents/mycode/final/datasets/assessments.csv")
#there are no duplicates, so this is our primary key that we can join on
assessments %>% count(assessmentId) %>% filter(n >1)
```

```
## [1] assessmentId n
## <0 rows> (or 0-length row.names)
```

```r
#now this is going to create multiples, which is fine and wanted.
animalassess <- assessments %>% left_join(animals, by = "assessmentId")

#These were the seen species, unimportant columns were filtered out and the oceans
#with a physical value were kept
animalassessSeen <- animalassess %>% select(assessmentId: assessmentDate, id_no:sci_name,
  legend:Oceans, populationTrend, systems:realm) %>% filter(Oceans != "NA")


#filtering out duplicates of animalassess seen
animalsSeenCount <- animalassessSeen %>%  distinct(Oceans, assessmentId, .keep_all = TRUE) %>%
  group_by(Oceans, redlistCategory) %>% summarize(SpeciesPer = n())
```

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

```r
#the tibble size is 268 x 2
animalassessSeen %>% group_by(assessmentId) %>% summarize(n = n())
```

```
## # A tibble: 268 x 2
##    assessmentId     n
##           <int> <int>
##  1       723579    25
##  2       725623   356
##  3       728197     6
##  4      1131667     2
##  5      1139783    26
##  6      1157078   257
##  7      1157356   413
##  8      1526216     3
##  9      1735255    23
## 10      1737230     3
## # i 258 more rows
```

```r
#Not spotted species

#this holds all of the occurrences where they were not spotted or noted,
#is.na grabs the values that do not have an ocean listed
animalassessNA <- animalassess %>% select(assessmentId : assessmentDate,
  threats : populationTrend, systems : possiblyExtinctInTheWild, Oceans) %>%
  filter(is.na(Oceans))


#The tibble size is 7090 x 2
animalassessNA %>% group_by(assessmentId) %>% summarize(n = n())
```

```
## # A tibble: 7,090 x 2
```

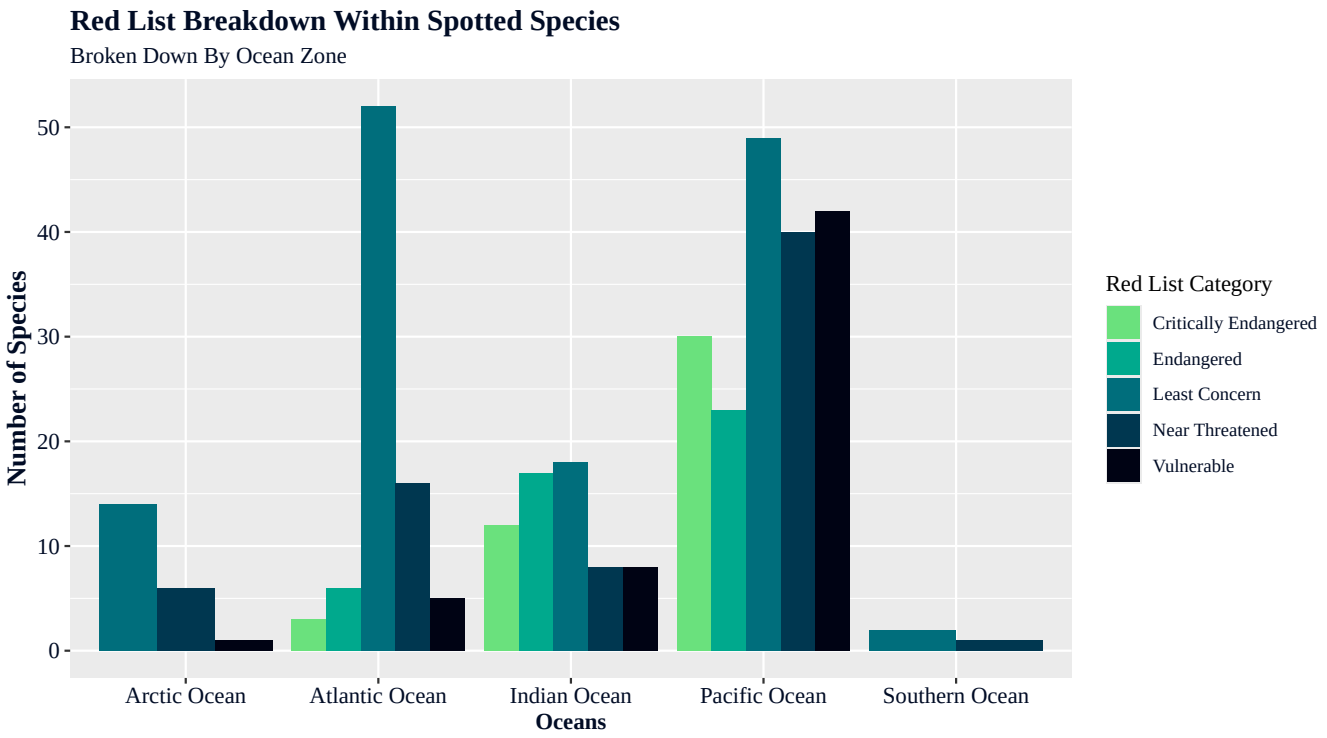
```
##    assessmentId     n
##           <int> <int>
##  1       495630     1
##  2       497802     1
##  3       500969     1
##  4       505458     1
##  5       506633     1
##  6       512306     1
##  7       514996     1
##  8       718871     1
##  9       720431     1
## 10       722547     1
## # i 7,080 more rows
```

```
#NAcounts groups by redlist and shows the sum for each category
NAcounts <- animalassessNA %>% group_by(redlistCategory) %>% summarize(count = n())

#label making by using pull to pull out the values of the colum
countsList <- NAcounts %>% pull(count)
RLC <- NAcounts %>% pull(redlistCategory)

#to stop overlap
RLC[6] <- paste("Lower Risk", "\n" ,"Near Threatened")

labels <- paste(RLC, "\n", countsList)
```

The Product: Spotted Species

This is a bar graph using aes fill to separate each ocean bar into smaller bars that represent the number of species that fall within a Red List category. To make it easily legible, dodge is used to stop the bars from stacking. Colors were manually assigned, and the theme was applied, along with labeling everything as needed.

```
#filling in the data, using identity to calculate the sum of y and dodge to separate the bars
ggplot(animalsSeenCount, aes(x = Oceans, y = SpeciesPer)) +
  geom_bar(stat = "identity", position = "dodge", aes(fill = redlistCategory)) +
  scale_fill_manual(values= c("#6ae17d", "#00a98d", "#006e7c", "#003750", "#000314"))+
  #theme code for fonts, sizes, and colors
  theme(axis.text = element_text(color = "#010C25", family = "serif", size = 11),
      axis.title.x = element_text(family = "serif", face = "bold", color = "#010C25"),
      axis.title.y = element_text(family = "serif", size = 13, face = "bold", color = "#010C25"),
      plot.title = element_text(size = 14, family = "serif", hjust = 0, face = "bold",
      color = "#010C25"),
      plot.subtitle = element_text(family = "serif", hjust = 0, color = "#010C25"),
      legend.text = element_text(family = "serif", size = 9, color = "#010C25"),
      legend.title = element_text(family= "serif")) +
  labs( #label code
    title = "Red List Breakdown Within Spotted Species",
    subtitle = "Broken Down By Ocean Zone",
    y = "Number of Species",
    fill = "Red List Category"
  )
```



As stated, the main purpose of this graph is to show the breakdown of the Red List categories within each ocean. It was expected that the Atlantic and the Pacific ocean would have the most endangered species, which was proven partially true. This fact, specifically the Pacific Ocean, should also be remembered for later analysis. The Indian Ocean also possessed a large portion of endangered species, even more than the Atlantic. In the Indian Ocean lies many coral reefs, which are known as the hot spot of biodiversity, with one reef in this particular region comprising of around 600 species. However, because of ocean acidification, pollution, and more interference, these reefs are suffering, which could be interpreted in this increase in vulnerable species (in general, not the category) in the graph.

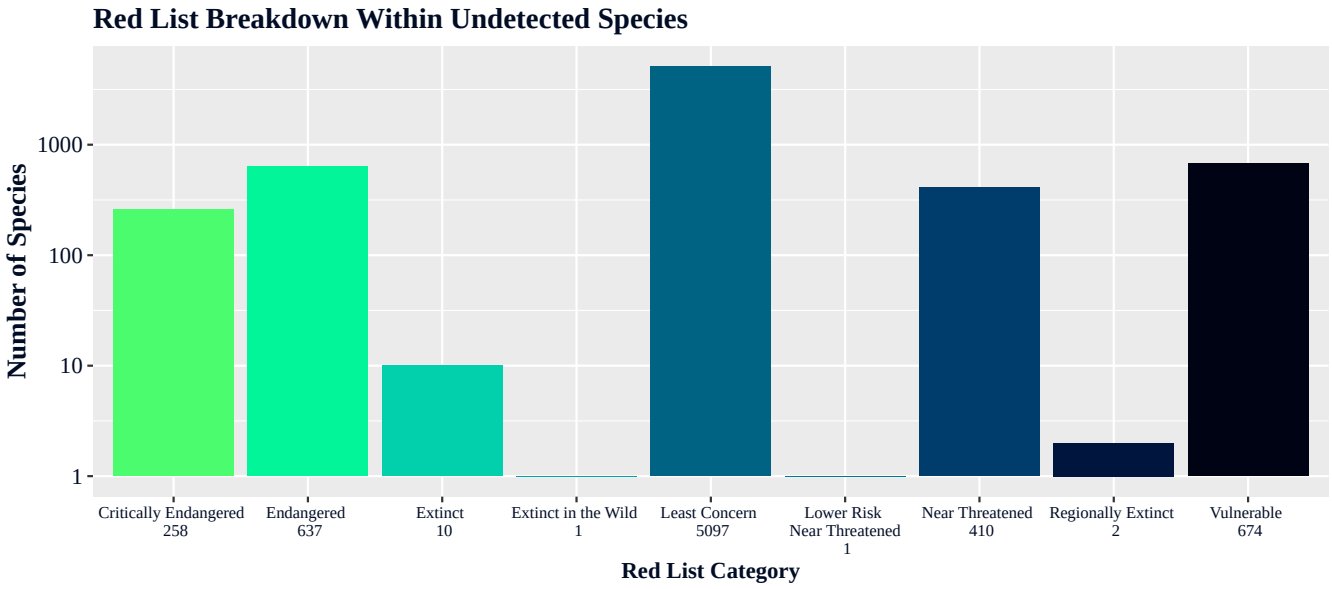
It was also expected to show a lot more endangered species because of the fact that most of the data points were located around the coast, which is where most of the trash is found. This could be because of some potential unintentional bias in the data because these endangered species were possibly simply not spotted because of their lack of numbers. This, or there may have been certain parameters that did not allow for all species to be tracked like the others. It should also be taken into consideration that these seen animals were only a fraction of the total species within the dataset, and this dataset is only a fraction of the actual amount of species in the ocean.

The Product: Undetected Species

The purpose here is to finish our question by seeing the distribution of Red List categories of the other species not in the previous graph. This bar graph uses aes fill for Red list categories similarly to the last graph, however there is no further grouping outside the x axis like in the previous. This is because there are no latitudes or longitudes (occurrences) for these, so nothing extra was needed. There are a little over 7000 points, with a large portion being within least concern. Because of this, it made the table very hard to read because of the shrinkage occurring. To fix this, a log scale was applied. However, since the quantities of each were still a crucial part of the graph, the labels created earlier were used to properly show the true quantities.

```
#count sums the x axis, and fill separates it by redlist breakdown
#using count to sum up x axis values
ggplot(animalassessNA, aes(x = redlistCategory)) +
  geom_bar(stat = "count", aes(fill = redlistCategory)) +
  scale_y_log10() + #applying a log10 scale to the y axis
  scale_x_discrete(labels = labels) +  #the labels from the preprocessing
  scale_fill_manual(values= c("#4afc6e", "#03f599", "#02cfac", "#01a3a9", "#016383", "#00769a",
      "#003c6c", "#00153d", "#000314"))+
```

```
theme(legend.position = "none", #the theme
    axis.text = element_text(color = "#010C25", family = "serif", size = 11),
    axis.title.x = element_text(family = "serif", face = "bold", color = "#010C25"),
    axis.text.x = element_text(family = "serif", size = 8),
    legend.text = element_text(family = "serif"),
    legend.title = element_text(hjust = 0.5, family = "serif", face = "bold"),
    axis.title.y = element_text(family = "serif", size = 13, face = "bold", color = "#010C25"),
    plot.title = element_text(size = 14, family = "serif", hjust = 0, face = "bold",
    color = "#010C25"),
    plot.subtitle = element_text(family = "serif", hjust = 0, color = "#010C25")) +
labs( #labelling
  title = "Red List Breakdown Within Undetected Species",
  y = "Number of Species",
  x = "Red List Category"
)
```



Even though an overwhelming amount of the species in this visual are in the least concern, the other categories are proportionately higher as well. With the endangered and vulnerable categories both greater than 600, this shows that more species available for analysis will result in more at-risk species appearing in the data. This table even shows two regionally extinct and one completely extinct species, categories that would not have been gathered in the last two visualizations. Once again, it should be stated that this is only a fraction of all endangered species, with more species on the decline from our disruption of their habitats.

Intermission: Modeling

The Idea Behind

Previously, I wanted to do some sort of regression with my CO2 emissions data and my water statistics. After thoroughly looking within the CO2 data, it was no where near what I was imagining, so the dataset was pitched. I tried searching for another CO2 dataset, but they were all worse than the last, so my regression plan and CO2 emission stats idea was completely pitched. When analyzing the columns of the datasets I already had, I was brought back to the animals, specifically the taxonomy dataset. This dataset had each level of classification for each species as the other two datasets, which were all strings of text. This initially steered me away from using it, but I thought it could be used in a clustering model as a way to label. Initially I wanted to do the Red List category versus the threat from the assessments table, but I decided against this since the Red List is already a product of certain factors,

like the population and threats it faces. I then thought that the threats and population trend could prove to bring insightful findings. The idea was that different orders of species could potentially group together based on similar threats faced and if their populations are stable, increasing, decreasing, or unknown. I wanted to do it with species, however, with how this was done, a lot of overlap would occur, and limiting my sample even more than I already did felt like it wouldn't represent as well. The order is the fourth level from the top in the breakdown, so its broad but not too broad to where no insightful findings would occur.

An issue within this model lied in the variable "threats" in the assessments data. This variable was a paragraph of text with the threat buried within it. To be able to properly use it, I went back to the IUCN website and found the threat filter breakdowns. Here it listed very specific threats such as Biological resource use, which included unintentional fishing, as in bycatch, and intentional fishing, or Natural system modification, that considered dams and habitat loss. So I went in Excel, created a column, and hand scraped the specific threat data based off what the text said and named it by the IUCN threat name. This was done for 300 of the undetected species and all of the seen species to give a variety, since different species of cuttlefish, for example, all fell one after the other. (I do not want to mention how long this took). The end result produced 12 different threats for 567 species.

To be able to create the cluster model, specific packages were needed. These included Tidyclust and Tidymodels. Tidyclust contains functions primarily for unsupervised clustering models, as could be inferred by the name. The most important function used within this package is k_means, which specifies the k-means clustering model. Next, tidymodels is the main package, holding all of the specifics for recipes and workflows, which are required for the process. The recipe functions are needed for preprocessing and normalization, and workflow is what combines every component together. These can be customized and refined for whatever the use may be, such as for Principal Component Analysis. Another package required for my specific model is ggrepel, which is an add-on to ggplot that simply helps with overlap in labels in a quick and easy fashion, which will be seen in use later on.

Data Wrangling For Figures 6 Through 9

To make sure everything reproduces nicely within each run of the file, a seed is set that keeps everything together. Taxonomy is brought in and filtered to grab all of the distinct Taxon Ids. The two datasets containing the threats and population trends are brought in and filtered to remove excess columns, rows with empty threat cells, and fixed to numerical values. These two datasets were combined together with rbind, and left-joined with taxonomy to match everything to the 567 species. Variables that did not apply to the process were removed, which left the final dataframe with 4 variables. Plus, an additional dataframe was created for future use.

```
library(tidyclust)
library(tidymodels)
library(ggrepel)

set.seed(333) #seed for reproducibility

#importing in the taxonomy csv and keeping all different values
taxonomy <- read.csv("C:/Users/ats20/OneDrive/Documents/mycode/final/datasets/taxonomy.csv")
taxonomy <- taxonomy %>% distinct(internalTaxonId, .keep_all = TRUE)

#reading in the Seen (pullout) and the undetected (pull) species after hand scraping out the
#threats from the paragraphs in the table
clustertable <- read.csv("C:/Users/ats20/OneDrive/Documents/mycode/final/pullout.csv")
clustertable1 <- read.csv("C:/Users/ats20/OneDrive/Documents/mycode/final/pull.csv")

#selecting a couple columns from the first table, filtering out the rest that I did not
#scrape and fixing internalTaxonId as it somehow wasn't an integer anymore
clustertable <- clustertable %>% select(assessmentId, scientificName, internalTaxonId,
    Threat, threats, redlistCategory, populationTrend) %>% filter(Threat != "",
      populationTrend != "") %>%
    mutate(internalTaxonId = as.integer(internalTaxonId))

#Doing the same here just for reassurance everything is proper.
```

```r
#Removing an index that appeared in the csv when it was pulled out
clustertable1 <- clustertable1 %>% filter(Threat != "", populationTrend != "") %>%
    mutate(internalTaxonId = as.integer(internalTaxonId)) %>% select(-X)

#combining these together by rows with rbind
clustertable <- rbind(clustertable,clustertable1)

#left joining it to the taxonomy table
clustertable <- clustertable %>% left_join(taxonomy, by = "internalTaxonId")

#Before filtering out the unimportant columns, a new table is created for later use
#after the cluster model is made
clustertablenew <- clustertable %>% select(redlistCategory, Threat, populationTrend, orderName,
    familyName)

#final table with columns removed before extra processing
clustertable <- clustertable %>% select(redlistCategory, Threat, populationTrend, orderName)
```

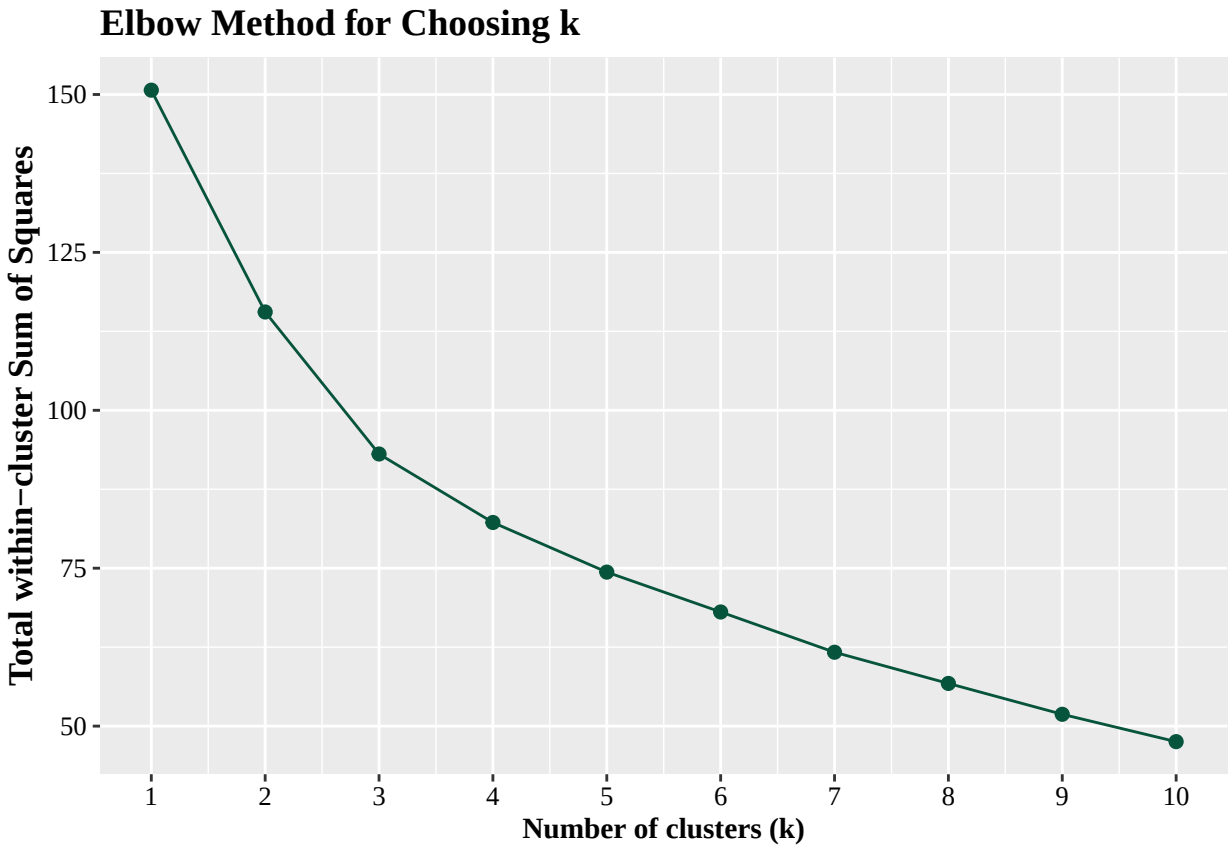
Both of my variables for this cluster model are categorical, which cannot work for this unless they are in a numerical form. So to do this, I used one-hot encoding, which is a technique that is used to turn variables into the proper numeric format. This is done by representing each by a 0 or 1, where 1 represents if it is that threat or trend, and 0 if it isn't. With this, pivot_wider is used, which spreads out the column for one feature per column. Once the two parts are combined, it turns into a high-dimension dataframe from the large amount of attributes. The orders are also lost later one, so to retain those for labeling, they are put into a variable here.

```r
#one-hot
#select the order name and threat column, separate out the , and then bring these
#out using pivot wider to split the threats into a threat per column. values function is
#used to apply the length number to the cell that actually has content, and 0 elsewhere
#. > 0 says if the value of the cell is greater than 0, then it is replaced with 1
threatsOneHot <- clustertable %>%
    select(orderName, Threat) %>%
    separate_rows(Threat, sep = ",") %>%
    pivot_wider(names_from = Threat, values_from = Threat,
                values_fn = length, values_fill = 0) %>%
  mutate(across(-orderName, ~ ifelse(. > 0, 1, 0))) #across applies across columns

#same done here but for population, no need to separate anything out prior to
#pivot wide as it was only one value per cell to begin with
trendOneHot <- clustertable %>% select(orderName, populationTrend) %>%
    pivot_wider(names_from = populationTrend, values_from = populationTrend, values_fn =
    length, values_fill = 0) %>% mutate(across(-orderName, ~ ifelse(. > 0, 1, 0)))

#joining the tables together to create one big table of 60 observations and 18 variables
#high dimensionality
prepped <- threatsOneHot %>% left_join(trendOneHot, by = "orderName")

#One read as not assigned, this was excluded
prepped <- prepped %>% filter(orderName != "NOT ASSIGNED")

#holding the Order name of the species for later
OrderSpec <- prepped %>%
  select(orderName)
```

Figuring out k Amount of Clusters

Before PCA begins (explained later on), we need to know the amount of clusters to use for the model, which can be found through the elbow graph. This graph visualizes the total within-cluster sum of squares decreasing in each increase of k, or number of clusters. The way to figure out the optimal number of clusters is where this decrease begins to slow down, which is seen by a smaller slope as opposed to the other clusters. Using map from purrr is something I picked up in earshot of others, which after reading its help page, it seemed much easier to follow than the loop in the class code. It iterates within the sequence of numbers and applies a function to each number of the sequence. The result of this was put into a tibble as the y-axis, with a sequence of 1-10 for the x-axis. This table graphed as a line plot, with each point representing the number of clusters, with its y representing the within-cluster sum of squares.

```
#creating a table holding the features only
features <- prepped %>% select(-orderName)


#checking how many clusters I should use as where it starts to slow down
#map_dbl cycles 1 through 10 and runs function(k) on each number, retrieving the within-cluster
#sum of squares for each number
  wss <- map_dbl(1:10, function(k) {
    model <- kmeans(features, centers = k, nstart = 10)
    model$tot.withinss
  })

#this is put into a tibble where x is 1 through 10 and y is the sum of squares
elbow <- tibble(k = 1:10, wss = wss)

#plotting these and cleaning up the final result with some color, labels, and the theme-
ggplot(elbow, aes(x = k, y = wss)) +
  geom_point(size = 2, color = "#07543C") +
  geom_line(color = "#07543C") + #theme for colors, fonts, and sizes
  theme( axis.text = element_text(color = "#000000", family = "serif", size = 9.5),
         axis.title.x = element_text(family = "serif", face = "bold"),
         axis.title.y = element_text(family = "serif", size = 13, face = "bold"),
         plot.title = element_text(size = 14, family = "serif", hjust = 0, face = "bold"),
         plot.subtitle = element_text(family = "serif", hjust = 0)) +
  scale_x_continuous(breaks = seq(1, 10, by = 1), labels = seq(1, 10, by = 1)) +
  labs( #labels
    title = "Elbow Method for Choosing k",
    x = "Number of clusters (k)",
    y = "Total within-cluster Sum of Squares"
  )
```



From this, we can see that three is the optimal amount of clusters for the graph, as it is the last point that follows a steep decrease. The rest have a substantially smaller slope, that shows that the sum of squares is not minimized as greatly as it was before k = 3. This number will be used in the final model.

Before the elbow method in the one hot encoding, it was mentioned that our prepped table now has high dimensionality. This is not good, however it can easily be fixed by using Principal Component Analysis. PCA has the ability to fix the high dimensionality all while keeping the most significant information. It is an unsupervised learning method that creates the variables called principal components. Once this process is done, these new principal components can be clustered together to ultimately make the final model.

The process below is what makes the final table for graphing the model. First, the recipe preprocesses, scales and reduces the dimensionality. K means is used to define the amount of clusters, using the stats engine. Both of these components are then combined in workflow, which begins the process of clustering with the three k-means clusters. Fit is used to fit the workflow to the dataset. Lastly, predict returns the cluster assignments, which is piped into a bind columns functions that applies the trained recipe to the prepped dataset. This is then piped into a column bind with the order names that were pulled out after one-hot encoding.

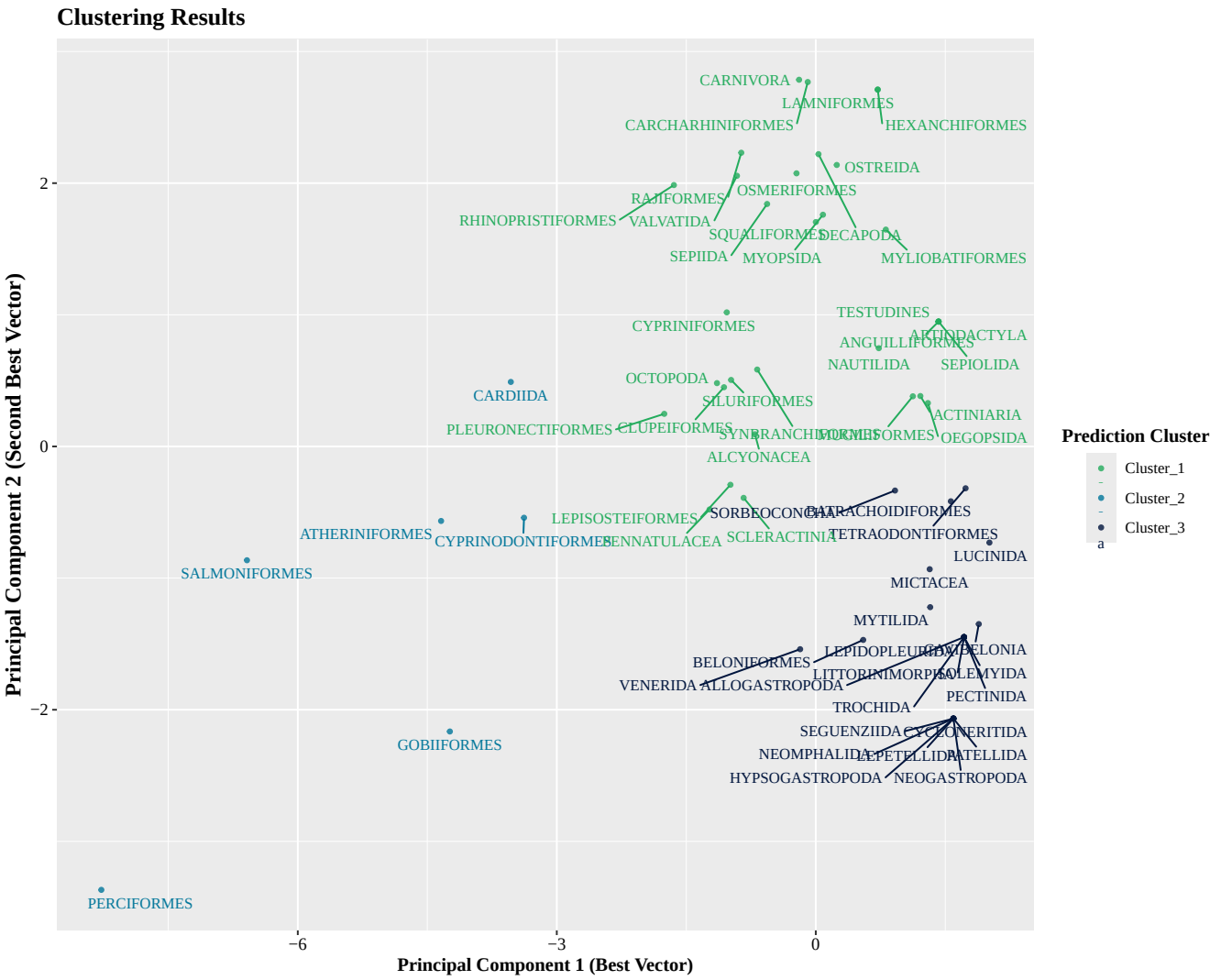
```
#Applying Principal component analysis within normalizing
pca_recipe <- recipe(~ ., data = prepped) %>%
  #assigning an initial role to the orderName
  update_role(orderName, new_role = "id") %>%
  #centers and scales
  step_normalize(all_numeric_predictors()) %>%
  #running pca and reduces it down to two components
  step_pca(all_numeric_predictors(), num_comp = 2)

# KMeans after PCA -> declaring three clusters, the number found in the elbow plot
pca_kmeans_spec <- k_means(num_clusters = 3) %>%
  set_engine("stats") #the type of engine used, based in base r
```

```r
#combining the recipe and model together, begins the cluster process
pca_workflow <- workflow() %>%
  add_recipe(pca_recipe) %>%
  add_model(pca_kmeans_spec)

#fitting the workflow to the prepped dataset
pca_fit <- fit(pca_workflow, data = prepped)

# Get cluster assignments and PCs
pcaDone <- predict(pca_fit, prepped) %>%
  bind_cols( #applying the trained recipe to the prepped data
    bake(prep(pca_recipe), new_data = prepped) %>% select(starts_with("PC"))
  ) %>% cbind(OrderSpec) #adding back the order names from earlier
```

The Product: Cluster Model

Now the finalized cluster model can be graphed. Below, the data is put into ggplot. In aes, PC1 is assigned to x, PC2 is assigned to y, the color comes from the prediction clusters, and the labels are from the order names. Scale color manual applies custom colors, and the theme and labels are applied. This is where geom_text_repel is used, to stop the overlap of labels. Since there was only so much variation in the points with 0s and 1s, many landed in the same locations, which made it hard to read the labels. Max overlap is a way to stop large amounts of lines occurring, this was set to 20 so everything can run smoothly without error.

```r
ggplot(pcaDone, aes(x = PC1, y = PC2, color = .pred_cluster, label = orderName)) +
  geom_point(size = 1, alpha = 0.8) + #changing the size of the points
  scale_color_manual(values= c("#23AC5D", "#00769a", "#00153d")) + #adding colors
  #applying size and fonts for the text in repel
  geom_text_repel(vjust = -0.5, size = 3, max.overlaps = 20, family = "serif") +
  #theme for fonts, sizes, and colors
  theme( axis.text = element_text(color = "#000000", family = "serif", size = 9.5),
         axis.title.x = element_text(family = "serif", face = "bold"),
         axis.title.y = element_text(family = "serif", size = 13, face = "bold"),
         plot.title = element_text(size = 14, family = "serif", hjust = 0, face = "bold"),
         plot.subtitle = element_text(family = "serif", hjust = 0),
         legend.title = element_text(hjust = 0.5, family = "serif", face = "bold"),
         legend.text = element_text(family = "serif")) +
  labs( #labels
    title = "Clustering Results",
    x = "Principal Component 1 (Best Vector)",
    y = "Principal Component 2 (Second Best Vector)",
    color = "Prediction Cluster"
    )
```



First, let's take a look at the axes. As it's known, the axes capture the variation, with the x-axis capturing the most, and the y-axis capturing the second-most. The x-axis, Principal component 1, is labelled from -6 to 0, showing that most of the points fall left from zero, with few going to the right. This captures a lot of the spread occurring. The points further away from 0 are likely have stronger variation in the features as opposed to the others. This is coming from the primary threats. Principal Component 2 goes from -2 to 2, which shows how the values are spread evenly above and below 0. This is coming from the population trends. A downside of this setup of cluster model is the discrete variables (0,1) make interpretation a little bit harder to do.

In terms of the clusters themselves, there are a couple different things happening. Cluster 1 in the top right corner has a bit of spread occurring, which shows there is some variation within the cluster itself. PC2 varies a bit, and Cluster 2 is more spread out than the last, with fewer points and a less defined shape, but showing more variation in primary threats, as seen with PC1. Cluster 3 is the most compact out of all of them and is tight in the bottom right corner. It has minimal variation in terms of PC1 and PC2. They are all fairly separated from each other, with no specific points of overlap. It's hard to classify the findings as is, especially with the order names. So to further analyze these clusters and to prove if similar species were in fact grouped together by the threats and population trends, we are going to analyze the families.

Analyzing the Findings Visually with Word Clouds

The family of a species is the step under order, so it breaks it down even further. These are also more recognizable by their name unlike order, which can be a bit obscure at times. This is to see if there is in fact, similar species within the clusters. Seeing this could help show the patterns in the families that face similar threats and population

trends. To do this analysis, word clouds will be utilized. The purpose of the word cloud is to show frequency of words by proportionally changing its size, so small words appear less and big words appear more. Word clouds are not available in regular ggplot, but in another package that can be applied with ggplot syntax, the same way ggrepel is used. This package is called ggwordcloud. However, this did not simply install, like packages normally should. Another package called xfun was not completely updated, and ggwordcloud required at least version 0.52 to run. Why ggwordcloud needed xfun is because it contains utility functions to run, as they are created by the same person. Yihui Xie, the creator, stated that he had wrote around 20 R packages, and after accumulating many basic utility functions that were used among multiple packages, he decided to put all of them into one package. This is why it was a necessary install / update.

To make word clouds for each of the three clusters, the table that was pulled out in the data wrangling step of the clustering model was finally used. This table contained the family names. First, the order names for each cluster were filtered out, and using pull, were turned into a sequence variable. Next, the family name table is used, and it is filtered by the order names in the list. This is then piped into a select function, that grabs the family name column. Group_by is used to group by the family name, and summarize is used to sum up the counts of each. Mutate creates a color column that cycles through the sequence of 6 colors, one per row. This process is done for each cluster, making them ready for graphing.

```
library(xfun)
library(tools)
library(ggwordcloud)

colors = c("#6CB355", "#0F8A59", "#00a98d", "#006e7c", "#003750", "#000314")

#cluster 1
#filtering cluster 1 and selecting the order names
list1 <- pcaDone %>% filter(.pred_cluster == "Cluster_1") %>% select(orderName)

#making them into a list
list1 <- list1 %>% pull(orderName)

#in the family table, filter the order names in the list, and select the family name column
families1 <- clustertablenew %>% filter(orderName %in% list1) %>% select(familyName)

#group by family name and count up the occurrences of each
families1 <- families1 %>% group_by(familyName) %>% summarize(count = n())

#adding the colors
families1 <- families1 %>% mutate(colors = rep(colors, length = nrow(families1)))


#cluster 2
#filtering cluster 2 and selecting the order names
list2 <- pcaDone %>% filter(.pred_cluster == "Cluster_2") %>% select(orderName)

#making them into a list
list2 <- list2 %>% pull(orderName)

#in the family table, filter the order names in the list, and select the family name column
families2 <- clustertablenew %>% filter(orderName %in% list2) %>% select(familyName)

#group by family name and count up the occurrences of each
families2 <- families2 %>% group_by(familyName) %>% summarize(count = n())

#adding the colors
families2 <- families2 %>% mutate(colors = rep(colors, length = nrow(families2)))
```

```r
#cluster 3
#filtering cluster 3 and selecting the order names
list3 <- pcaDone %>% filter(.pred_cluster == "Cluster_3") %>% select(orderName)

#making them into a list
list3 <- list3 %>% pull(orderName)

#in the family table, filter the order names in the list, and select the family name column
families3 <- clustertablenew %>% filter(orderName %in% list3) %>% select(familyName)

#group by family name and count up the occurrences of each
families3 <- families3 %>% group_by(familyName) %>% summarize(count = n())

#adding the colors
families3 <- families3 %>% mutate(colors = rep(colors, length = nrow(families3)))
```

The Product: Cluster 1

GGplot is used to add the data, and aes declares the labels (the family), size (from the count of each family), and color. Geom_text_wordcloud_area creates the wordcloud and rm_outside stops overlap. Scale_size_area makes the image bigger or smaller, and scale_color_identity allows for custom colors to be applied.

```r
ggplot(families1, aes(label = familyName, size = count, color = colors)) +
  geom_text_wordcloud_area(rm_outside = TRUE, family = "serif") +
  scale_size_area(max_size = 38) +
  scale_color_identity() +
  theme(plot.title = element_text(size = 20, family = "serif", hjust = 0, face = "bold")
        ) +
  labs(
    title = "Families Within Cluster 1"
  )
```

Families Within Cluster 1

The most frequent occurrences in cluster 1 were Octopodidae, Ommastrephidae, Sepiidae, and Megaleledonidae. These are Octopi, small to large squids, cuttlefish, and another family of octopi. Other species in here include families of sharks, rays, more squids, skates, and flatfish. This shows some unity in the findings of the first cluster as all live in the relatively same habitat, with similar features to each other.

The Product: Cluster 2

Same process as above, scale_size_area at 70 because of less family names in the cluster.

```
ggplot(families2, aes(label = familyName, size = count, color = colors)) +
  geom_text_wordcloud_area(rm_outside = TRUE, family = "serif") +
  scale_size_area(max_size = 70) +
  scale_color_identity() +
  theme(plot.title = element_text(size = 20, family = "serif", hjust = 0, face = "bold")
        ) +
  labs(
    title = "Families Within Cluster 2"
  )
```

Families Within Cluster 2

Most common families in cluster 2 were sparidae, gobiidae, acanthuridae, cardiidae, and salmonidae. These are ray-finned fish, bony fish, and cockles. Cockles (cardiidae) are a type of shelled species like mollusks. considering the other species are bony/ray-finned fish, this seems to be an outlier that arose from the clustering, as the other families seem to be ones that may be affected by fishing threats.

The Product: Cluster 3

Same process as cluster 1, scale_size_area at 47 because of more family names in the cluster.

```
ggplot(families3, aes(label = familyName, size = count, color = colors)) +
  geom_text_wordcloud_area(rm_outside = TRUE, family = "serif") +
  scale_size_area(max_size = 47) +
  scale_color_identity() +
  theme(plot.title = element_text(size = 20, family = "serif", hjust = 0, face = "bold")
        ) +
  labs(
    title = "Families Within Cluster 3"
  )
```

Families Within Cluster 3

The main families of cluster 3 were provannidae, mytilidae, lepetodrilidae, peltospiridae, and neomphalidae. These are mollusks, sea snails, and mussels. This seems like it would be the better cluster for the mollusk family cardiidae, which is was right on the line for. Tetraodontidae, which has only one occurrence, is a puffer fish. It also appears on the border of its cluster. This shows that there is some pattern and similarities within the clusters created from the population trends and threats. Though not perfectly matching everything together, it does a great job when many occurrences of a specific family is present.

Part 3: Water

The Search for Datasets: Water

After showing trash (pollution) statistics and animal statistics, information about the habitat could provide insight to the clustering model and declining species. The ocean is a finicky environment, where the smallest change can cause major stress. Initially, the ocean pH used to be 8.2, which is slightly alkaline. This created the perfect environment for life, however, with pollution and air emissions, this has decreased to 8.1. Though seemingly not a big change, it is a 30% increase in acidity as the pH scale is logarithmic. The goal was to find a dataset containing measures of ocean water and see what types of information has been collected. I found the Ocean Carbon and Acidification Data System which contained information mainly about the Pacific Ocean on NOAA, where microplastic data was found previously. This dataset contained 28,207 record and 72 variables. Many values contained -999 which I assumed meant the measure was not taken, as the code book didn't explicitly describe what this meant. It also contained many In-Situ variables, which refers to being measured directly within the environment without being removed or transported.

Data Wrangling For Figure 10 and 11

I first decided to take a small sample of the data, to see where each individual point lied, as I wanted to represent all points as a density plot. To do this I began by pulling out all of the columns I would be using for the next couple

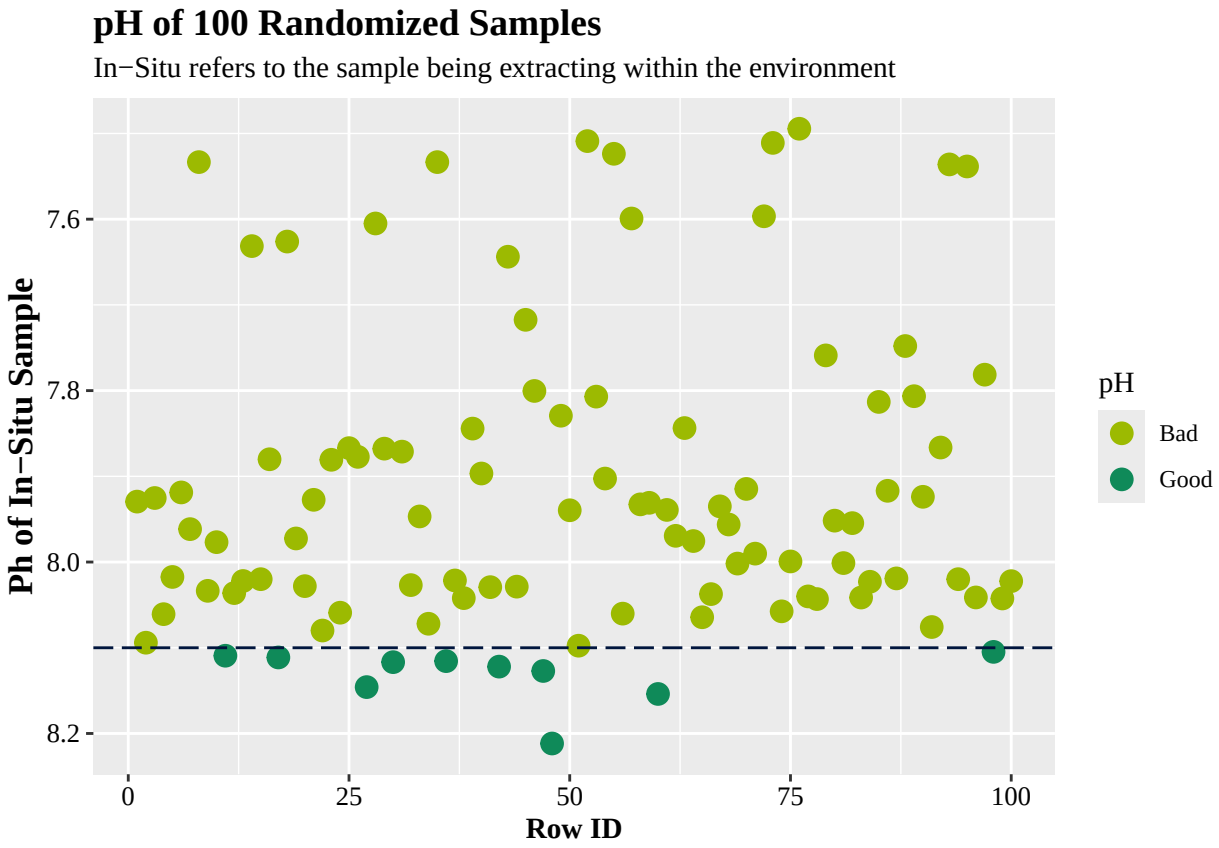
graphs. I then made a new table to hold the pH insitu measurement, and filtered out anything that was not an actual measurement. Then, 100 samples were randomly selected. A row ID column was added to give something to graph the x-axis by. This has no meaning within the graph except for showing all 100 samples separately without overlap. Another column was added to sort the pH as good or bad depending on its value. This would be for a color aspect in the graph.

```
#reading in the file
water <- read.csv("C:/Users/ats20/OneDrive/Documents/mycode/final/datasets/usawaterstuff.csv")

#selecting the columns I felt could be insightful for later
water <- water %>% select(Depth, Salinity_PSS78,
    recommended_Salinity_PSS78, pH_TS_measured, TEMP_pH, pH_TS_insitu_measured,
    pH_TS_insitu_calculated, Carbonate_measured,
    Carbonate_insitu_measured, Carbonate_insitu_calculated)

#removing the -999 variable and the row that read a n.a., this left the
#table with 10,684 observations.
waterpH <- water %>% filter(pH_TS_insitu_measured != -999 & pH_TS_insitu_measured
    != "n.a.") %>% mutate(pH_TS_insitu_measured = as.numeric(pH_TS_insitu_measured))

#I grabbed 100 random points for my random sample
waterpHSample <- sample_n(waterpH, 100)
#adding a rowId and adding another column classifying the pH as good or bad
#depending if it goes over 8.1 or not.
waterpHSample <- waterpHSample %>% mutate(rowId = row_number()) %>%
  mutate(pHGB = ifelse(pH_TS_insitu_measured >= 8.1, "Good", "Bad"))
```

The Product: pH of 100 Randomized Samples

This visual is plotting each point, with a horizontal line marking the average pH of the ocean currently. The y-axis was flipped to mimic how a pH scale is read. The colors also mimic the typical pH scale colors, despite being discrete.

```
ggplot(waterpHSample, aes(x = rowId, y = pH_TS_insitu_measured)) +
  geom_point(aes(color = pHGB), size = 3.5 ) + scale_color_manual(
    values = c( "#9CBA01", "#0F8A59")) +
  scale_y_reverse() + #scale reverse to flip y axis
  #creates horizontal line at 8.1
  geom_hline(yintercept = 8.1, linetype = "longdash", color = "#00153d") +
  #theme
  theme( axis.text = element_text(color = "#000000", family = "serif", size = 9.5),
         axis.title.x = element_text(family = "serif", face = "bold"),
         axis.title.y = element_text(family = "serif", size = 13, face = "bold"),
         plot.title = element_text(size = 14, family = "serif", hjust = 0, face = "bold"),
         plot.subtitle = element_text(family = "serif", hjust = 0),
          legend.title = element_text(family = "serif"),
         legend.text = element_text(family = "serif")) +
  labs( #labels
    y = "Ph of In-Situ Sample",
    x = "Row ID",
    title = "pH of 100 Randomized Samples",
    subtitle = "In-Situ refers to the sample being extracting within the environment",
    color = "pH"
  )
```

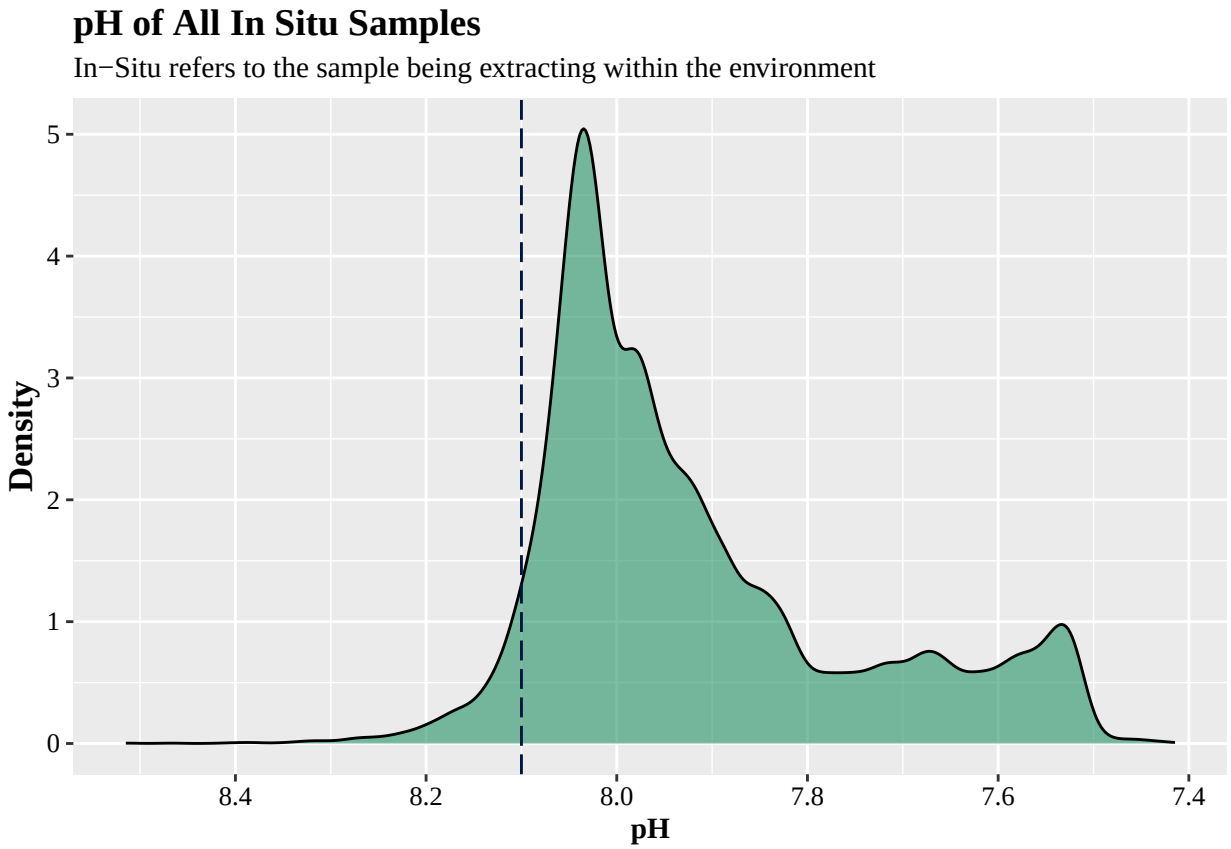


The graph shows the most of the points get very acidic, with some even going past 7.6. There isn't much to note outside of the overwhelming proportion of values above what water should be. This can cause severe stress to the environment and species, which can cause their physical decline.

The Product: Density of all pH Samples.

This maps the density of all of the points to compare to the randomized sample. I wanted to see where the majority of the pHs were located for all 10,000 points and if there were any patterns among this. Again, a line is drawn for what the pH currently is for the visual of the healthy and unhealthy pH.

```
#density
ggplot(waterpH, aes(x = pH_TS_insitu_measured)) +
  geom_density(fill = "#0F8A59", adjust = 0.65, alpha = 0.55)+
  #vertical line at 8.1
  geom_vline(xintercept = 8.1, linetype = "longdash", color = "#00153d") +
  scale_x_reverse()+ #flipping axis again
  #theme
  theme( axis.text = element_text(color = "#000000", family = "serif", size = 9.5),
         axis.title.x = element_text(family = "serif", face = "bold"),
         axis.title.y = element_text(family = "serif", size = 13, face = "bold"),
         plot.title = element_text(size = 14, family = "serif", hjust = 0, face = "bold"),
         plot.subtitle = element_text(family = "serif", hjust = 0)) +
  labs(#labels
    y = "Density",
    x = "pH",
    title = "pH of All In Situ Samples",
    subtitle = "In-Situ refers to the sample being extracting within the environment"
  )
```



Luckily, it seems as though the majority of the points are a bit above 8.0, except this is still just as bad. This small of a change is detrimental to everything in the ocean, and species cannot adapt to this type of change at the speed at which it is occurring. There is also a peak under 7.6, which is incredibly bad. The fact that there are barely any measures in a healthy pH shows just how the water quality is beginning to get, causing harm to species and biodiversity.

Data Wrangling for Figure 12

After looking at the randomized samples and seeing how many samples continuously come up with a low pH, I thought to look at the salinity levels and carbonate levels. All materials that make up the ocean are connected in very complex ways, so when one part, like the pH, begins to suffer, the rest follow suit. Carbonate is used by species like corals and oysters to create and maintain their shells and skeletons. With more ocean acidification occurring, this carbonate in the ocean begins to bond with hydrogen, leaving fewer ions for these species to use, and creating byproducts. The salinity also plays an equally important role on carbonate. Salinity influences the density of water and the solubility of carbonate, this in turn affects the formation of carbonate as well.

The idea in this graph was to see if there was a correlation seen in mapping the pH, salinity, and carbonate together by general depths. Hypothetically, levels with a lower pH (more acidic) and a lower salinity would in turn have a lower carbonate level. To model this I decided to get with a heat map, with three different parts for pH, salinity, and carbonate, with each tile representing a range of depth. The package patchwork was required to simply add the graphs together side by side. This could have been done with a facet, but each scale was very different, which would have poorly affected the heatmap. I did this by creating the table and checking that all of my numbers are reading as numeric values with mutate. I needed to make the depth ranges, so I used a case when statement and mutate to do this. Each range is represented by the lower of the two, so 100-200 is represented by 100, etc. Once this was done, I grouped all the measurements by depth, and averaged each separately, to get the most fair measurement. Lastly, I pulled out each of the components of water for their respective heatmap.

```
library(patchwork)
```

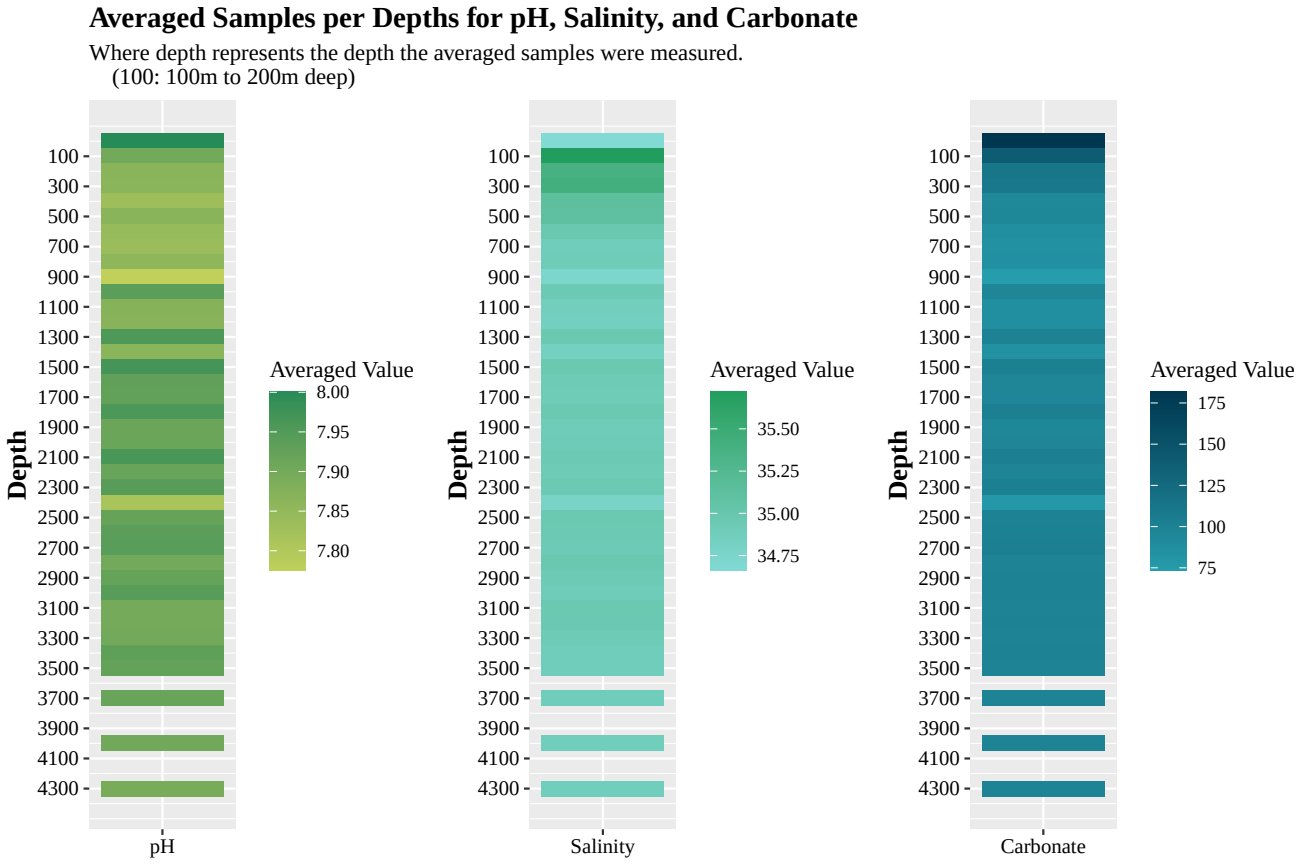
```r
#creating the water table with ph, salinity, and carbonate
waterCarbonateSalinitypH <- water %>% filter(Carbonate_insitu_measured != -999 &
    Carbonate_insitu_measured != "n.a." & Salinity_PSS78 != -999 & Salinity_PSS78
     !="n.a." & pH_TS_insitu_measured != -999 & pH_TS_insitu_measured != "n.a.") %>%
    select(Carbonate_insitu_measured, Salinity_PSS78, Depth,
      pH_TS_insitu_measured) %>%
    mutate(Depth = as.numeric(Depth), Carbonate_insitu_measured =
      as.numeric(Carbonate_insitu_measured),
    Salinity_PSS78 = as.numeric(Salinity_PSS78), pH_TS_insitu_measured =
      as.numeric(pH_TS_insitu_measured))

#each case represents a different level of the ocean to map the samples together
#to be able to bar them in the graph
waterCarbonateSalinitypH <- waterCarbonateSalinitypH %>% mutate(depthcat =
  case_when( Depth > 1 & Depth <= 100 ~ 0,  Depth > 100 & Depth <= 200 ~ 100,
    Depth > 200 & Depth <= 300 ~ 200,  Depth > 300 & Depth <= 400 ~ 300,
    Depth > 400 & Depth <= 500 ~ 400,  Depth > 500 & Depth <= 600 ~ 500,
    Depth > 601 & Depth <= 700 ~ 600,  Depth > 701 & Depth <= 800 ~ 700,
    Depth > 800 & Depth <= 900 ~ 800,  Depth > 901 & Depth <= 1000 ~ 900,
    Depth > 1001 & Depth <= 1100 ~ 1000,  Depth > 1101 & Depth <= 1200 ~ 1100,
    Depth > 1201 & Depth <= 1300 ~ 1200,  Depth > 1301 & Depth <= 1400 ~ 1300,
    Depth > 1401 & Depth <= 1500 ~ 1400,  Depth > 1501 & Depth <= 1600 ~ 1500,
    Depth > 1601 & Depth <= 1700 ~ 1600,  Depth > 1700 & Depth <= 1800 ~ 1700,
    Depth > 1801 & Depth <= 1900 ~ 1800,  Depth > 1901 & Depth <= 2000 ~ 1900,
    Depth > 2001 & Depth <= 2100 ~ 2000,  Depth > 2101 & Depth <= 2200 ~ 2100,
    Depth > 2201 & Depth <= 2300 ~ 2200,  Depth > 2301 & Depth <= 2400 ~ 2300,
    Depth > 2401 & Depth <= 2500 ~ 2400,  Depth > 2501 & Depth <= 2600 ~ 2500,
    Depth > 2601 & Depth <= 2700 ~ 2600,  Depth > 2701 & Depth <= 2800 ~ 2700,
    Depth > 2801 & Depth <= 2900 ~ 2800,  Depth > 2901 & Depth <= 3000 ~ 2900,
    Depth > 3001 & Depth <= 3100 ~ 3000,  Depth > 3101 & Depth <= 3200 ~ 3100,
    Depth > 3201 & Depth <= 3300 ~ 3200,  Depth > 3301 & Depth <= 3400 ~ 3300,
    Depth > 3401 & Depth <= 3500 ~ 3400,  Depth > 3501 & Depth <= 3600 ~ 3500,
    Depth > 3601 & Depth <= 3700 ~ 3600,  Depth > 3701 & Depth <= 3800 ~ 3700,
    Depth > 3801 & Depth <= 3900 ~ 3800,  Depth > 3901 & Depth <= 4000 ~ 3900,
    Depth > 4001 & Depth <= 4100 ~ 4000,  Depth > 4101 & Depth <= 4200 ~ 4100,
    Depth > 4201 & Depth <= 4300 ~ 4200,  Depth > 4301 & Depth <= 4400 ~ 4300))

#here we group by the depth of the ocean where the measure was taken, then each of
#the variables are summarized with a average within each depth
waterCarbonateSalinitypH <- waterCarbonateSalinitypH %>% group_by(depthcat) %>%
  summarize(Salinity = mean(Salinity_PSS78), Carbonate = mean(Carbonate_insitu_measured),
  pH = mean(pH_TS_insitu_measured))

#these are then stacked with pivot_longer
waterCarbonateSalinitypH <- waterCarbonateSalinitypH %>%
  pivot_longer(cols = c(Carbonate, pH, Salinity), names_to = "groupings",
  values_to = "avg")

#filtering out each different variable
ph <- waterCarbonateSalinitypH %>% filter(groupings == "pH")
carbonate <- waterCarbonateSalinitypH %>% filter(groupings == "Carbonate")
salinity <- waterCarbonateSalinitypH %>% filter(groupings == "Salinity")
```

The Product: Comparing pH, Salinity, and Carbonate

Each graph is done the same, but with names altered to match the proper graph. The process was applying the variables in ggplot, and adding geom_tile. Scale_y_reverse makes the depth go deeper the further down you go. Labels and breaks were made for the y axis to make, a theme and coloring was applied, and labels were added when necessary. The colors for pH and salinity somewhat mimic their color scale, and carbonate did not have a clear color scale to follow. Xlab null helps from a duplicate x title from appearing.

```
#each graph represents a different thing, this is pH
ggplot(ph, aes(x = groupings, y = depthcat, fill = avg)) +
  geom_tile() + #reverse and labels
  scale_y_reverse(breaks = seq(100, 4400, by = 200), labels = seq(100, 4400, by = 200)) +
  #theme
  theme( axis.text = element_text(color = "#000000", family = "serif", size = 9.5),
         axis.title.x = element_text(family = "serif", face = "bold"),
         axis.title.y = element_text(family = "serif", size = 13, face = "bold"),
         plot.title = element_text(size = 14, family = "serif", hjust = 0, face = "bold"),
         plot.subtitle = element_text(family = "serif", hjust = 0),
         legend.title = element_text(family = "serif"),
         legend.text = element_text(family = "serif")) + xlab(NULL) +
  scale_fill_continuous(low = "#C0D05B", high = "#268B57") +
  labs( #labels
    y = "Depth", #title applied here and stretches over other two
    title = "Averaged Samples per Depths for pH, Salinity, and Carbonate",
    subtitle = "Where depth represents the depth the averaged samples were measured.
    (100: 100m to 200m deep)",
    fill = "Averaged Value"
  ) +
  #this is salinity
ggplot(salinity, aes(x = groupings, y = depthcat, fill = avg)) +
  geom_tile() +
  scale_y_reverse(breaks = seq(100, 4400, by = 200), labels = seq(100, 4400, by = 200)) +
  #theme
  theme( axis.text = element_text(color = "#000000", family = "serif", size = 9.5),
         axis.title.x = element_text(family = "serif", face = "bold"),
         axis.title.y = element_text(family = "serif", size = 13, face = "bold"),
         legend.title = element_text(family = "serif"),
         legend.text = element_text(family = "serif")
         ) + xlab(NULL) +
    scale_fill_continuous(low = "#82DAD5", high = "#219E5D") +
  #labels
  labs(
    y = "Depth",
    fill = "Averaged Value"
  ) +
  #this is carbonate
ggplot(carbonate, aes(x = groupings, y = depthcat, fill = avg)) +
  geom_tile() +
  scale_y_reverse(breaks = seq(100, 4400, by = 200), labels = seq(100, 4400, by = 200)) +
  theme( axis.text = element_text(color = "#000000", family = "serif", size = 9.5),
         axis.title.x = element_text(family = "serif", face = "bold"),
         axis.title.y = element_text(family = "serif", size = 13, face = "bold"),
         legend.title = element_text(family = "serif"),
         legend.text = element_text(family = "serif")) +
  xlab(NULL) + scale_fill_continuous(low = "#269DAC", high = "#003750") +
  labs(
  y = "Depth",
```

```
  fill = "Averaged Value"
)
```



As seen, the hypothesis was overall correct, where lower pH and salinities had lower carbonate levels as well. This proves how ocean acidification does simply affect the pH of the water, but every other part of the water as well. There are some pieces where the trend is not as strong, however the lowest pH sections have the most visible correlation. The highest pHs show the most Carbonate. The skipping at the bottom is where samples for those levels were not gathered. I also suspected that there would be more carbonate as the bottom of the ocean since many creatures use it to build their shells, however this was not the case.

Conclusion

In conclusion, our disturbances to this planet have not been kind to the species that were here before us. The threats they face that put them at high-risk for potential extinction only seems to increase. Pollution is a driving factor, as these materials are causing irreversible damage to them. From viewing where trash lies in line of where species live, and seeing what Red List category they fall under, we have seen from a statistical viewpoint the conditions they live in. Outside of that, the water has also taken a hit. These foreign materials leach chemicals into the water, which affect the delicate balance that makes up the ocean habitat. Another big factor of this is CO2 which is heavily related to ocean acidification, which would be a potential branch for further analysis. Another great point for more analysis would be looking into my graphs more thoroughly and seeing if more can be drawn from them with additional data added, such as plant and fungi data that is located in the ocean.

Sources

Datasets

For Trash

https://experience.arcgis.com/experience/b296879cc1984fda833a8acc93e31476/page/Page?views=Data-Table

https://www.coastalcleanupdata.org/reports

For Animals

https://www.iucnredlist.org/

For Water

https://www.ncei.noaa.gov/products/ocean-carbon-acidification-data-system

https://www.ncei.noaa.gov/data/oceans/ncei/ocads/data/0219960/

For General Information

https://en.wikipedia.org/wiki/Great_Pacific_Garbage_Patch

https://en.wikipedia.org/wiki/North_Atlantic_garbage_patch

https://tobaccofreelife.org/resources/marine-life-cigarettes-pollution/

https://acmecology.com.au/hanges-in-salinity-affect-marine-life/#How_Salinity_Levels_Can_Change

https://wwf.panda.org/wwf_news/?334171/Importance-of-the-marine-biodiversity-of-the-Western-Indian-Ocean

https://www.noaa.gov/education/resource-collections/ocean-coasts/ocean-acidification

https://pubs.usgs.gov/pp/0186n/report.pdf

https://www.sciencelearn.org.nz/resources/141-environmental-conditions-affecting-the-sea

https://salinity.oceansciences.org/aq-salinity.htm

https://madisongroup.com/plastics-in-appliances-a-comprehensive-overview/

https://www.webmd.com/a-to-z-guides/what-to-know-about-the-toxicity-of-polypropylene

https://www.sciencedirect.com/science/article/abs/pii/S030438941630629X

https://www.marinebio.org/creatures/facts/#:~:text=According%20to%20World%20Register%20of%20Marine%20Species%20%

https://biologyinsights.com/microplastics-and-their-impact-on-marine-ecosystems/

https://cran.r-project.org/web/packages/xfun/vignettes/xfun.html

chatGPT Searches:

another word for things in the context of “certain things”

dplyr function similar to sql case when

how to fix the size of the margins of code in rmarkdown files for pdf output

how to take values from a table and pipe them into a sequence variable in r

how to middle justify headers in an r markdown file

can you assign your own shape for geom_point or can you only use what is preset can i use arrow unicode chracters and how

ggrepel: 27 unlabeled data points (too many overlaps). Consider increasing max.overlaps how do i do this