Scraper Reef base

Edgar Chicurel

2023-03-20

Introduction

In this notebook, we used XPath (XML) and the Selenium package to build a web scraper that extracts data from a reef webpage. Specifically, we created scraper that extracts reef and marine data from different countries. In the following sections, we demonstrate the step-by-step process of building this web scraper using R and finish with some ways to visualize and analyse the extracted data.

Reef Webpage

Set-up

In this first part, we explain the process and code for setting up the required packages and getting to run Selenium (not easy).

Load needed libraries

To get started, we first installed the following R packages and we saved the reef_website link.

```
library(RSelenium)
library(xml2)
library(magrittr)
library(dplyr)
library(tidyr)
library(rvest)
library(httr)
library(cluster)
library(mclust)
library(countrycode)
library(rworldmap)
library(plotly)
library(ggplot2)
library(stringr)
library(tidyverse)


reef_link <- "http://www.reefbase.org/global_database/default.aspx?section=r1"
```

Open the explorer and name the driver

Selenium is a web testing framework that allows us to automate interactions with a website. In order to use Selenium with R, we needed to install the RSelenium package and after fixing some issues we got it running and named the driver **"driver"**.

```
remDr <- rsDriver( port = 45893L, browser = "chrome",
                   chromever = "110.0.5481.77")

driver <- remDr$client
```

Extract data for 1 country

In this part, we first built a web scraper to extract reef and marine data from a single country. We will use Cameroon as an example, and demonstrate how to navigate the website, select the desired country from a dropdown menu, and extract the relevant data.

Access the Reef webpage directly in "Overview" section

First part is to access the webpage through Selenium directly in the "Overview" section with the saved link specified before.

```
remDr$client$navigate(reef_link)
```

Find and click Statistics button

In this section, we show how we found and clicked "Statistics" button on the web page using xml2. It can be seen that the link does not changes when that button is clicked. We used the xml_find_all() and findElement() functions to locate the "Statistics" button on a web page and click it using the clickElement() method.

```
reef_link %>%
  read_html() %>%
  xml_find_all("//tr/td/a[contains(text(), 'Statistics')]")

driver$findElement(value = "//tr/td/a[contains(text(), 'Statistics')]")$clickElement()
```

Selecting a Region: Africa

Next, we are in the section of the webpage that we wanted and now we need to select a region, in this case: Africa. We used the same technique as before but now specifying the option for Africa which is 1 in the Selenium function.

```
reef_link %>%
  read_html() %>%
  xml_find_all('//td/select[@name = "ctl00$Content$cboRegion"]')

#option value = 1 for Africa

driver$findElement(value = '//td/select[@name = "ctl00$Content$cboRegion"]//option[@value = "1"]')$clic
```

Selecting a country: Cameroon

Next, bye using same technique as before, now specify in the next dropdown menu the option CMR for the country of interest.

```
reef_link %>%
  read_html() %>%
  xml_find_all('//td//select[@name = "ctl00$Content$cboCountry"]')

driver$findElement(value = '//td//select[@name = "ctl00$Content$cboCountry"]//option[@value = "CMR"]')$
```

Clicking the "search" button

Finally, we find and click on the search button. Once the button is clicked, the output page will be loaded and ready for data extraction.

```
reef_link %>%
  read_html() %>%
  xml_find_all('//div/input[@type = "submit" and @name="ctl00$Content$btnAdvancedSearch"]')

driver$findElement(value = '//div/input[@type = "submit" and @name="ctl00$Content$btnAdvancedSearch"]')$
```

Extract html code and store it

We will use the getPageSource() to extract the page source code, which we will then store in a variable named page_source. We will then use read_html() function in to read the page source into R and store it in a variable named html_code.

```
page_source <- driver$getPageSource()[[1]]

html_code <-
  page_source %>%
  read_html()
```

Get Table and open it

Because the data we want is on a table, we used the function html_table to extract all the tables in the webpage. After looking at each one of them, we found that the one we are looking for is in position 143.

```
all_tables <-
  html_code %>%
  html_table()

data <- all_tables[[143]]
data
```

Clean the data from the table

In this section, we cleaned the data table extracted from the reef webpage. Specifically, we renamed and selected columns, converted data types, and added a new column for country name.

```
colnames(data) = tolower(data[1, ] )
data[-1,]

data %>%
  rename(area_statistics=1, unit=3) %>%
  select(!c(graph, table)) %>%
  mutate(value = as.numeric(value),
         country = "Cameroon")
```

All countries

In this section, we created a loop to extract reef and marine data for all available countries on the website. We first went back to the main Overview page. We noticed that there was a region option for **all** so we clicked it, obtained the page source code, and saved all country nodes. We then created an empty dataframe to store the output of the loop. In the loop, we extracted the data for each country by clicking on the country option and then clicking on the search button. We extracted the data from the resulting table, cleaned it, and added the country code to distinguish it from other countries. Finally, we combined all the extracted data for all countries into one dataframe.

```
# Go back to the main Overview page
driver$findElement(value = '//td/select[@name = "ctl00$Content$cboRegion"]//option[@value = "0"]')$clic

driver$findElement(value = '//td//select[@name = "ctl00$Content$cboCountry"]//option[@value = "0"]')$cl


driver$findElement(value = '//div/input[@type = "submit" and @name="ctl00$Content$btnAdvancedSearch"]')$

# Get the page source code
page_source <- driver$getPageSource()[[1]]

# Save all country nodes
n_countries = page_source %>%
  read_html() %>%
  xml_find_all('//td//select[@name = "ctl00$Content$cboCountry"]//option') %>%
  length()

# Empty data drame to store loop
full_data = data.frame()

# Loop for all countries
for (j in seq_len(n_countries)){
  print(j)
  if(j == 1 | j == 39 | j == 46 | j == 63  | j == 94 | j == 112 | j == 123) {next}
  xpath = paste0("//td//select[@name = 'ctl00$Content$cboCountry']//option[", j, "]")
  Sys.sleep(2)
  driver$findElement(value = xpath)$clickElement()
  Sys.sleep(2)
  driver$findElement(value = '//div/input[@type = "submit" and
                     @name="ctl00$Content$btnAdvancedSearch"]')$clickElement()
  page_source <- driver$getPageSource()[[1]]
  all_tables = page_source %>%
     read_html() %>%
     html_table()

data <- all_tables[[143]]

colnames(data) = tolower(data[1, ] )
data[-1,]

data = data %>%
  rename(area_statistics=1, unit=3) %>%
  select(!c(graph, table)) %>%
  mutate(value = as.numeric(value),
         ctr_code = j)

full_data = full_data %>%
  rbind(data)

}
```

Data cleaning

We then proceeded to clean the full data frame in order to be used for data analysis and/or visualization.

Get country names

The code below clicks on the drop-down menu for selecting countries and extracts the country names from the webpage. The extracted country names are then converted to a data frame and given a new column name "country". A new column "index" is added to the data frame to store the row number.

```
driver$findElement(value = '//td/select[@name = "ctl00$Content$cboRegion"]//option[@value = "0"]')$clic

driver$findElement(value = '//td//select[@name = "ctl00$Content$cboCountry"]//option[@value = "0"]')$cl

driver$findElement(value = '//div/input[@type = "submit" and @name="ctl00$Content$btnAdvancedSearch"]')

page_source <- driver$getPageSource()[[1]]

country_names = page_source %>%
  read_html() %>%
  xml_find_all('//td//select[@name = "ctl00$Content$cboCountry"]//option') %>%
  html_text() %>%
  as.data.frame() %>%
  mutate(index = row_number()) %>%
  rename(country = '.')

head(country_names)
```

Join country code with respective country

We joined the full data frame with the newly created country name dataframe in order to dsiplay each country information correctyl in the dataset.

```
full_data = full_data %>%
  left_join(country_names, by = c("ctr_code" = "index"))

head(full_data)
```

Pivot Data

In order to see the data more clearly, we replaced some characters, renamed some variables and pivoted some columns.

```
reefbase = full_data %>%
  filter(area_statistics != c("Area Statistics", "Socioeconomic Statistics")) %>%
  rename(statistic = "area_statistics") %>%
  mutate(statistic = str_replace_all(statistic, " ","_")) %>%
  select(!unit) %>%
  group_by(ctr_code, country) %>%
  pivot_wider(names_from = "statistic", values_from = "value")

head(reefbase)
```

Save the dataset

We then proceeded to save the data as a csv for future analysis.

```
setwd("C:/Users/usuario/Desktop/Master/Data Harvesting/Project")
#write.csv(reefbase, "reefbase_by_country.csv")
reefbase = read_csv("reefbase_by_country.csv")
```

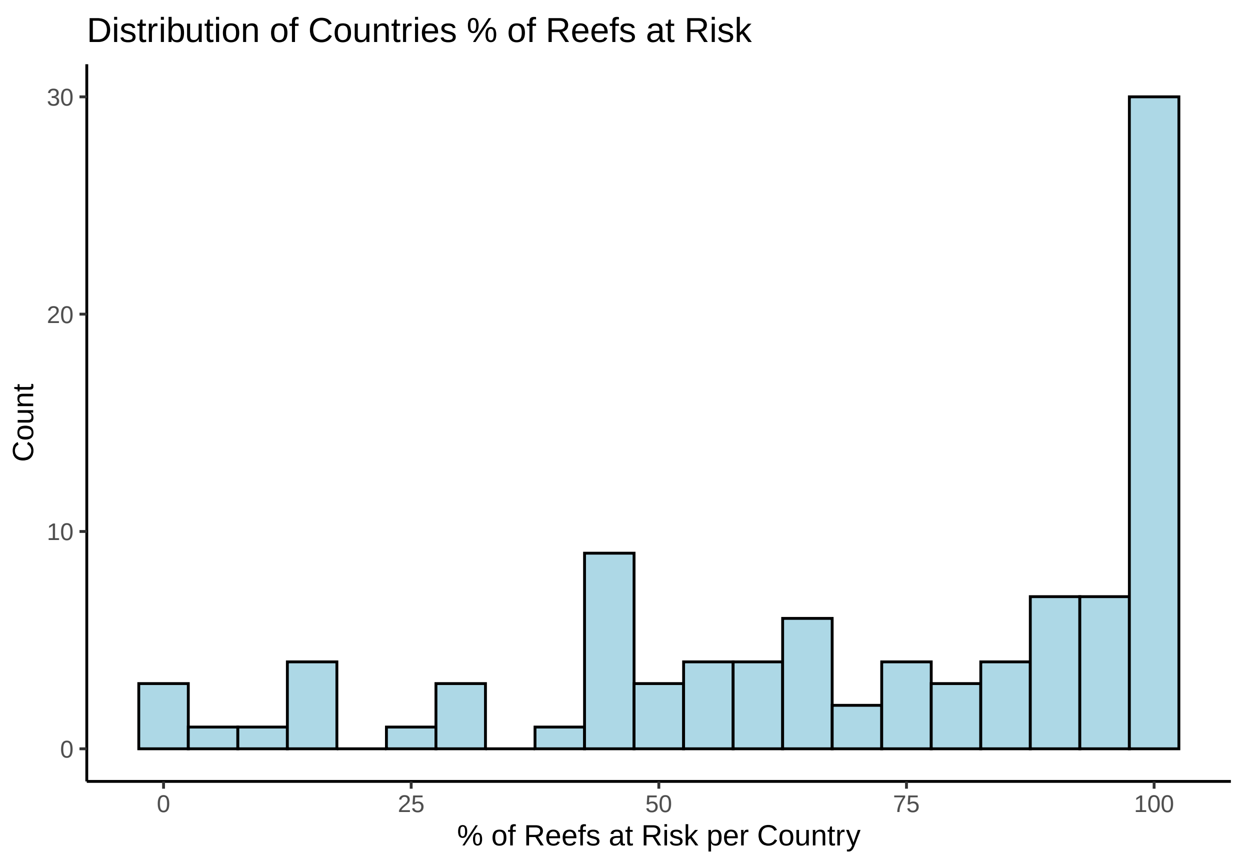
Data visualization

With the data that has been extracted from the ReefBase website, we can perform various data analysis and visualization techniques to gain insights about the reefs and their associated factors.For example, we can plot histograms of the Reefs At Risk percentages to observe the distribution across all countries.

Additionally, we can perform cluster analysis on the extracted indicators such as reef area per marine area, mangrove area, and inshore fishing area, to identify the similarities and dissimilarities between countries. These visualizations can provide valuable insights for policymakers and stakeholders to take informed decisions regarding conservation efforts and sustainable development of marine ecosystems.

Reefs at Risk Histogram

```
ggplot(reefbase, aes(x = Reefs_At_Risk)) +
  geom_histogram(binwidth = 5, color = "black", fill = "light blue") +
  labs(title = "Distribution of Countries % of Reefs at Risk", x = "% of Reefs at Risk per Country", y 
  theme_classic()
```



We can see that the majority of countries in the data set have 100% reefs at risk :(

Clustering

For this part we thought of doing a cluster analysis based on some indicators and values to see the distribution of countries that share characteristics. This is one of many examples that can be done with this extracted data.

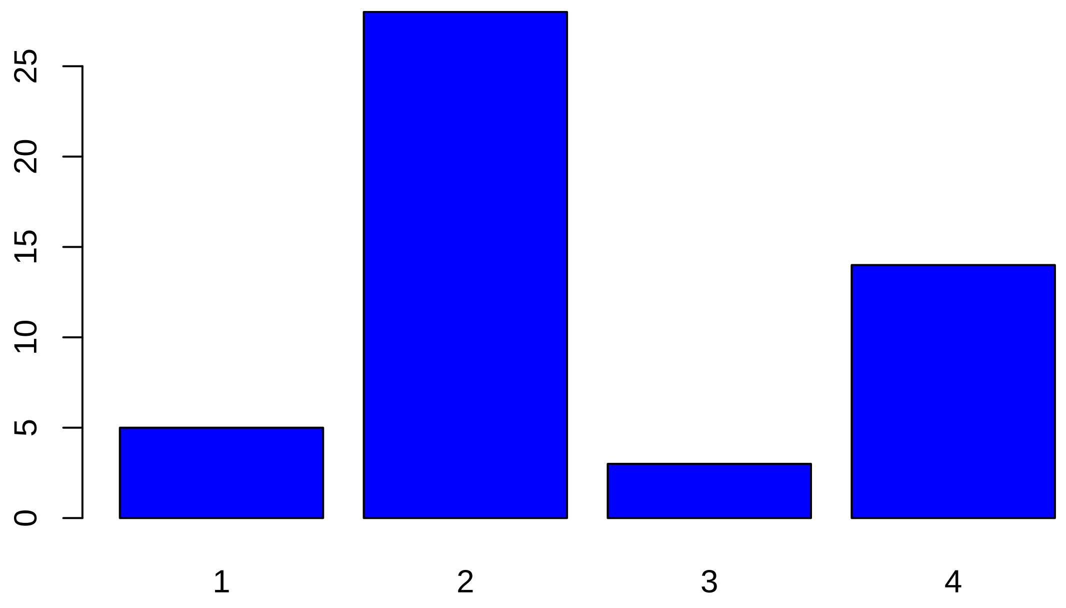
Create indicators and remove NAs First, we created indicators and removes NAs to simplify the analysis.

```
# Taking into account that the loop is too long, you can upload the csv created with the data screaped,
#reefbase = read.csv('reefbase_by_country.csv')
reef_countries = reefbase %>%
  transmute(
    reef_a_pm = Reef_Area/Marine_Area,
    mangrove_a_pm = Mangrove_Area,
    fishing_a_pm = Inshore_Fishing_Area,
    Reefs_At_Risk,
    Fish_Consumption,
    country) %>%
  drop_na()

reef_trns = reef_countries %>% select(-country)
```

Cluster model

We ran the cluster model, specifying 4 groups of countries and setting a seed for interpretability.

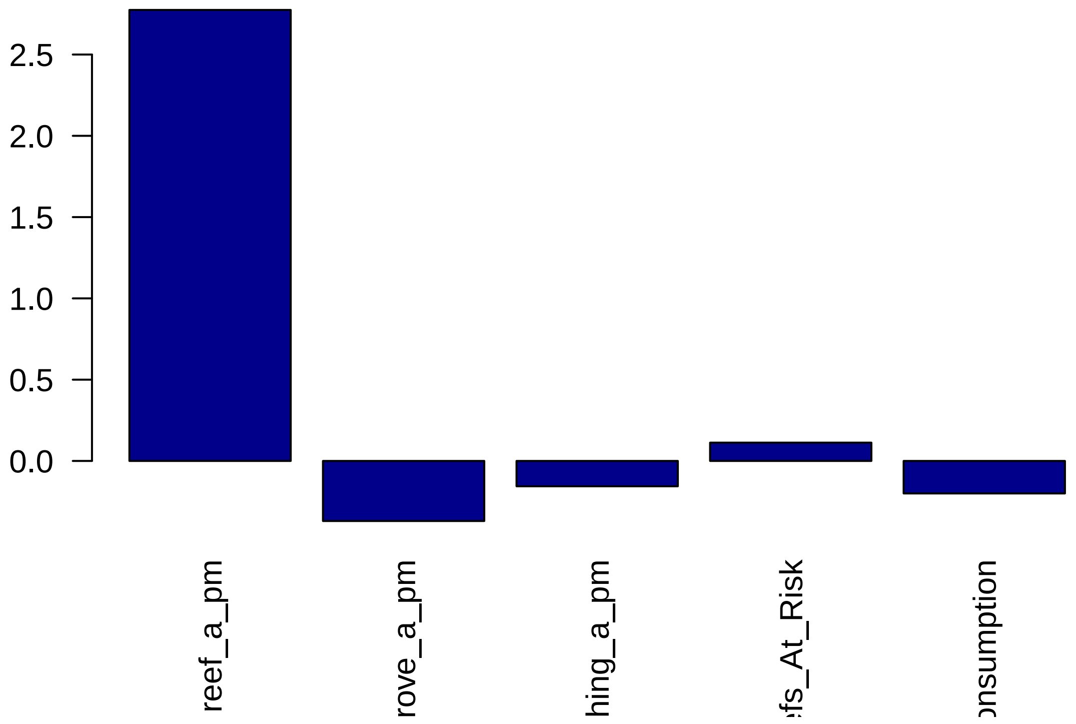
```
set.seed(3522)

k = 4

name.model = kmeans(scale(reef_trns), centers=k, nstart = 1000 )

groups = name.model$cluster
barplot(table(groups), col="blue")
```



We can observe that the majority of countries are located in groups 2 and 4.

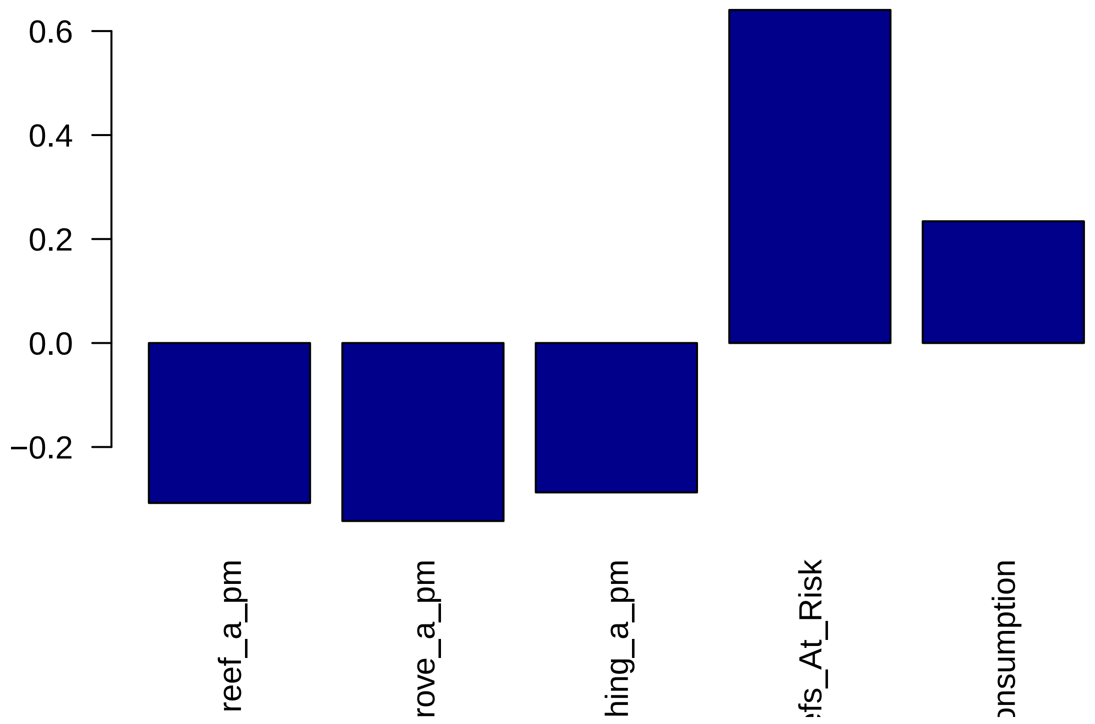
Interpretaion In this part we show how each group differentiates from one another and their specific characteristics.

```
centers=name.model$centers
barplot(centers[1,], las=2, col="darkblue")
```



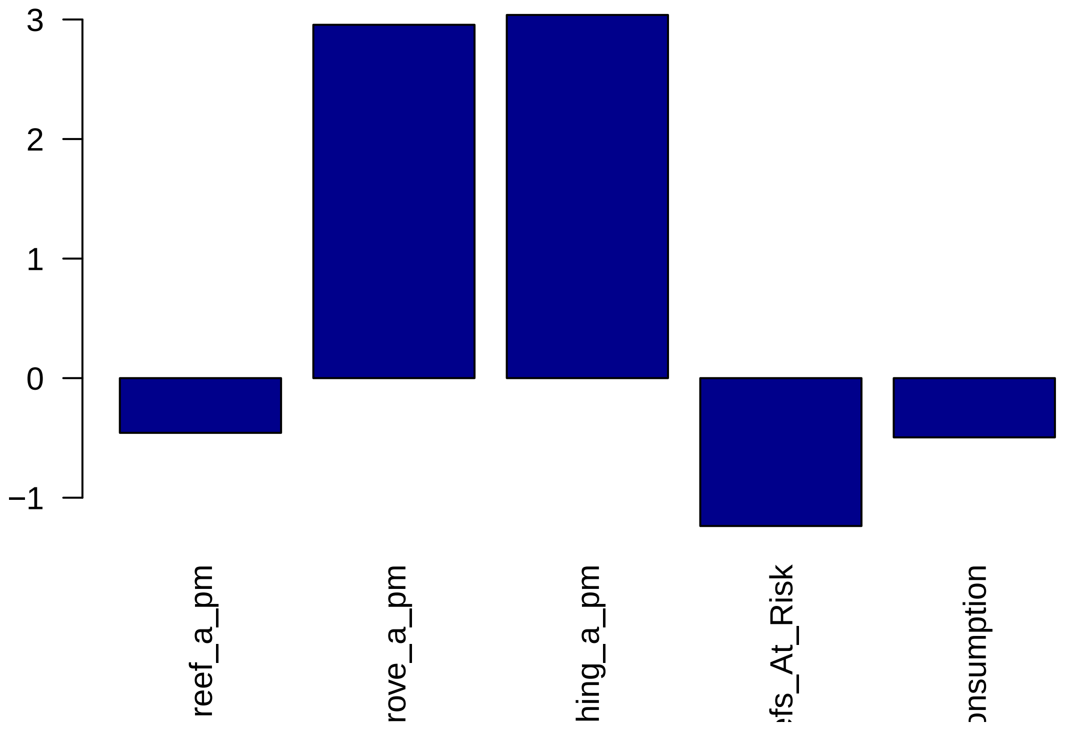
```
#Countries that have more reefs per marine area
```

```
barplot(centers[2,], las=2, col="darkblue")
```



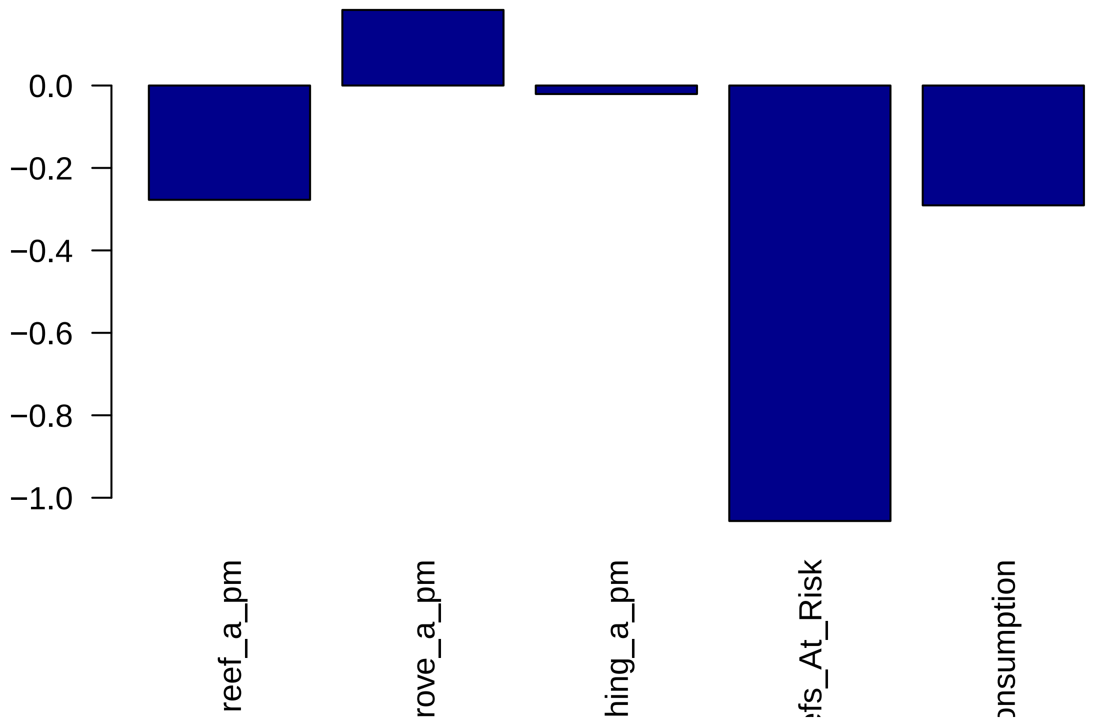
```
# High Fish consumption and Reef at risk
```

```
barplot(centers[3,], las=2, col="darkblue")
```



```
#Less fish consumption, reef at risk and less reef per marine area with higher inshore fishing area
```

```
barplot(centers[4,], las=2, col="darkblue")
```



```r
# Hiher mangrooves and fishing areas
```

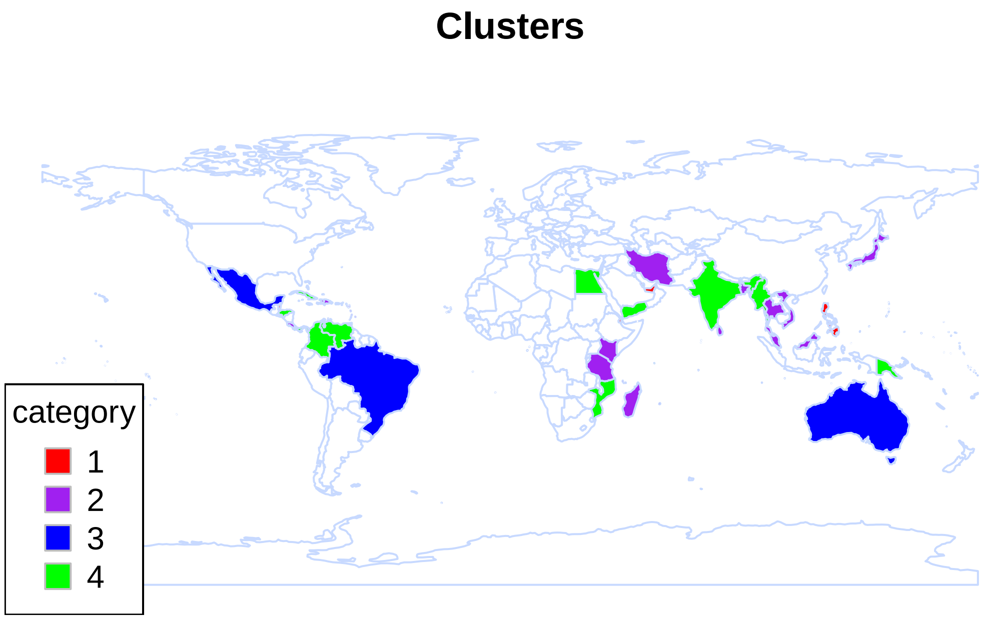
Final Plot

For this final plot we thought of mapping the cluster results. First, we matched the country codes to ISO3 codes using the countrycode() function, then joined the resulting data to a world map using joinCountry-Data2Map(). Finally, we ploted the map with the categories from the cluster analysis.

```r
names=reef_countries$country
map = data.frame(country=names, value=name.model$cluster)

#Convert the country code into iso3c using the function countrycode()
map$country = countrycode(map$country, 'country.name', 'iso3c')
#Create data object supporting the map
matched <- joinCountryData2Map(map, joinCode = "ISO3",
                               nameJoinColumn = "country")
```

```
## 50 codes from your data successfully matched countries in the map
## 0 codes from your data failed to match with a country code in the map
## 193 codes from the map weren't represented in your data
```

```r
#Draw the map
mapCountryData(matched,nameColumnToPlot="value",missingCountryCol = "white",
               borderCol = "#C7D9FF",
               catMethod = "categorical", numCats = 4,
               colourPalette = c('red','purple','blue','green'),
               mapTitle = c("Clusters"), lwd=1)
```



API

To contrast our findings with Natural Language Processing models, we called the CHAT GPT API and requested it to answer some questions about coral reefs. We sent a POST request, indicating the endpoint, credentials, and body.

```
token = read.table("key.txt", sep = '\t', header = TRUE)

call_chatgtp <- function(prompt) {
  response <- POST(
    url = "https://api.openai.com/v1/chat/completions",
    add_headers(Authorization = paste("Bearer", token[1,])),
    content_type_json(),
    encode = "json",
    body = list(
      model = "gpt-3.5-turbo",
      messages = list(list(
        role = "user",
        content = prompt
      ))
    )
  )
  str_trim(content(response)$choices[[1]]$message$content)
}


call_chatgtp("what is the biggest coral reef, and where are the coral reefs in risk?")
```

Conclusions

Web scraping is a powerful technique to collect data from websites, and it can be used to extract information that is not available in structured formats such as APIs or databases. In this project, we used Selenium to scrape data from the ReefBase website, which provides information about coral reefs and related ecosystems around the world. We collected data about the area of different types of ecosystems, as well as indicators of their health and use, for many countries.

We then cleaned and analyzed this data, creating summary statistics and visualizations such as histograms and cluster maps. Finally, we included an API connection to retrieve generated text from a NLP model, created by Open AI.