Final Project

HTHSCI 1M03 - Foundations of Data Science

Roy Luo 400474680

2023-04-10

Introduction:

The data we will be analyzing are two datasets (CSV file 1 and CSV file 2) in the form of CSV files from Kaggle concerning UFO spottings around the world. The CSV files includes data such as the time of the spotting, where the spotting took place, descriptions and durations of the encounter, etc. I personally chose this data and topic because I have always found it so interesting that despite there being so many UFO sightings, there still has not been any confirmation of the existence of extra terrestrials. I thus decided to investigate this topic further and come to a conclusion of why that is by trying to find:

- Which places in the world are more likely to have UFO spottings?
  - Analyzing the locations of UFO sightings can provide insights into the phenomenon of UFOs and the lack of confirmation of extraterrestrial life. This includes identifying geographical patterns of sightings, understanding human perception and reporting bias, investigating potential relationships with military and government installations, studying environmental factors and guiding investigative efforts.
- Do the types and length of UFO spottings vary from place to place?
  - Analyzing the length and type of UFO sightings can help us understand the lack of confirmation of extraterrestrial existence by considering factors such as the quality of evidence, potential for natural or human-made explanations and the need for scientific rigor to prevent a lack of credibility when evaluating the plausibility of potential UFO encounters.
- Are there any other interesting trends to note in historical global UFO spottings?
  - Finding additional trends in UFO sightings can contribute to our understanding of the lack of confirmation of extraterrestrial existence by examining consistency or inconsistency in patterns, considering human perception and reporting bias and avoiding unwanted assumptions or conlusions.

Data Wrangling Plan : File 1

Iteration 1

Phase 1

- Read CSV file into R (specify column types)
- Drop unnecessary columns and duplicate rows
- Change column names if necessary
- Identify uids and verify they are unique
- Sort by uids

Phase 2

Read CSV files into R (specify column types)

```
tib1 <- read_csv("complete.csv", col_types = "ccccciccdd")
```

```
## Warning: One or more parsing issues, call `problems()` on your data frame for details,
## e.g.:
##   dat <- vroom(...)
##   problems(dat)
```

```
tib1 %>% glimpse()
```

```
## Rows: 80,332
## Columns: 11
## $ datetime               <chr> "10/10/1949 20:30", "10/10/1949 21:00", "10/10/~
## $ city                   <chr> "san marcos", "lackland afb", "chester (uk/engl~
## $ state                  <chr> "tx", "tx", NA, "tx", "hi", "tn", NA, "ct", "al~
## $ country                <chr> "us", NA, "gb", "us", "us", "us", "gb", "us", "~
## $ shape                  <chr> "cylinder", "light", "circle", "circle", "light~
## $ `duration (seconds)`   <int> 2700, 7200, 20, 20, 900, 300, 180, 1200, 180, 1~
## $ `duration (hours/min)` <chr> "45 minutes", "1-2 hrs", "20 seconds", "1/2 hou~
## $ comments               <chr> "This event took place in early fall around 194~
## $ `date posted`          <chr> "4/27/2004", "12/16/2005", "1/21/2008", "1/17/2~
## $ latitude               <dbl> 29.88306, 29.38421, 53.20000, 28.97833, 21.4180~
## $ longitude              <dbl> -97.941111, -98.581082, -2.916667, -96.645833, ~
```

Drop unnecessary columns and duplicate rows

```
tib1 %<>% select(!c("comments", "date posted", "duration (hours/min)"))
tib1 %<>% distinct()
```

Change column names if necessary

Let's check the column names.

```
tib1 %>% colnames()
```

```
## [1] "datetime"           "city"               "state"
## [4] "country"            "shape"              "duration (seconds)"
## [7] "latitude"           "longitude"
```

Let's change the column names to something more fitting.

```
tib1 %<>% rename("state/province" = "state", "duration" = "duration (seconds)")
```

Now let's check that they are renamed.

```
tib1 %>% colnames()
```

```
## [1] "datetime"       "city"           "state/province" "country"
## [5] "shape"          "duration"       "latitude"       "longitude"
```

Identify uids and verify they are unique

```
tib1 %>%
  count(datetime, country) %>%
  filter(n > 1)
```

```
## # A tibble: 0 x 3
## # i 3 variables: datetime <chr>, country <chr>, n <int>
```

We see they are unique as they don't appear more than one.

Sort by uids

```
tib1 %<>% select(datetime, country, everything()) %>% arrange(datetime, country)
```

Let's get a final view of tib1 after our first iteration of changes. We can see the tibble has only relevant columns for our project, no duplicate rows, has a uniform naming convention for columns and is sorted by unique identifiers.

```
tib1 %>% glimpse()
```

```
## Rows: 69,586
## Columns: 8
## $ datetime         <chr> "1/1/1910 24:00", "1/1/1944 12:00", "1/1/1947 17:00",~
## $ country          <chr> "us", "us", NA, NA, NA, NA, "us", "us", "us", "us", "~
## $ city             <chr> "kirksville (near)", "san diego", "manama (bahrain)",~
## $ `state/province` <chr> "mo", "ca", NA, "tx", "ne", "ia", "wv", "tx", "tx", "~
## $ shape            <chr> "disk", "cigar", "circle", "cigar", NA, "circle", "di~
## $ duration         <int> 120, 180, 300, 300, 600, 30, 10, 20, 2, 10800, 30, 30~
## $ latitude         <dbl> 40.19472, 32.71528, 26.21667, 33.66013, 41.49254, 42.~
## $ longitude        <dbl> -92.58306, -117.15639, 50.58333, -97.71556, -99.90181~
```

Iteration 2

Phase 1

- Check and clean character columns
- Check and clean numeric columns
- Convert date columns into date datatype
- Convert appropriate columns into factors
- Drop NA values

Phase 2

Check and clean character columns

Let's start with the datetime column and remove all the rows that do not fit the normal date format (month day year hour minute) that is present in that column.

```
tib1 %<>% filter(!is.na(mdy_hm(datetime)))
```

Now let's see the unique entries in the country column.

```
tib1$country %>% unique()
```

```
## [1] "us" NA   "ca" "gb" "au" "de"
```

Let's rename them to lowercase.

```
tib1$country %<>% tolower()
```

Now let's view the entries in the city column.

```
tib1$city %>% glimpse()
```

```
##  chr [1:69586] "kirksville (near)" "san diego" "manama (bahrain)" ...
```

Let's rename the entries to lowercase. We see a lot of the entries end with something in brackets. Let's see what those bracket entries say.

```
tib1$city %<>% tolower()
bracketentries <- str_extract(tib1$city, "\\(.*\\)$")
bracketentries %>% glimpse()
```

```
##  chr [1:69586] "(near)" NA "(bahrain)" NA "(rural)" NA NA NA NA NA NA NA ...
```

We see either it's somewhat irrelevant information, or it contains the name of the country the city is located in. Let's remove the brackets if the text inside does not contain a country using the maps and countrycode libraries, or if the brackets contains the name of the country, let's make the country entry for that row the country in brackets.

```
tib1 %<>% mutate(country = ifelse(is.na(country), str_extract(city, "\\((.*)\\)$"), country))
tib1 %<>% mutate(country = str_remove_all(country, "\\(|\\)"))
tib1 %<>% mutate(city = str_replace(city, "\\s*\\(.*?\\)", ""))
```

This effectively cleans both the city and country columns. Now let's check one more time to make sure all the entries in the city and country columns are legitimate places, or make them NA otherwise. Let's also turn abbreviations of country names into the actual country names (again using the countrycode library).

```
tib1$country %<>% tolower()
tib1 %<>% mutate(country = ifelse(country %in% tolower(world.cities$country) |
  country %in% tolower(codelist$iso2c), country, NA))
tib1 %<>% mutate(city = ifelse(city %in% tolower(world.cities$name), city, NA))
tib1 %<>% mutate(country = countryname(country, destination = tolower("country.name.en")))
```

```
## Warning: There was 1 warning in `mutate()`.
## i In argument: `country = countryname(country, destination =
##   tolower("country.name.en"))`.
## Caused by warning:
## ! Some values were not matched unambiguously: au, ca, canary islands, de, gb, ne
```

Let's do the same thing for the state column.

```
tib1$`state/province` %>% unique()
```

```
##  [1] "mo" "ca" NA   "tx" "ne" "ia" "wv" "in" "fl" "mt" "il" "ar" "az" "nj" "la"
## [16] "md" "mi" "on" "wa" "al" "tn" "ny" "ga" "co" "ct" "id" "ma" "ok" "nc" "ky"
## [31] "ks" "sc" "ri" "va" "pa" "hi" "wi" "oh" "nv" "or" "bc" "me" "ak" "nb" "nh"
## [46] "nm" "de" "qc" "ut" "ms" "vt" "sa" "nt" "pr" "pq" "mn" "ab" "ns" "nd" "sd"
## [61] "sk" "mb" "wy" "dc" "nf" "yt" "pe" "yk"
```

It looks good. Now let's check the last character column which is the shape column.

```
tib1$shape %>% table()
```

```
## .
##   changed  changing   chevron     cigar    circle      cone  crescent     cross
##         1      1709       828      1885      6527       267         2       208
##  cylinder     delta   diamond      disk      dome       egg  fireball     flare
##      1163         7      1062      4714         1       688      5017         1
##     flash formation   hexagon     light     other      oval   pyramid rectangle
##      1150      2061         1     14122      4964      3273         1      1151
##     round    sphere  teardrop  triangle   unknown
##         2      4641       663      6876      4859
```

Some of these entries look similar so let's adjust them so they match better.

```
tib1 %<>% mutate(shape = ifelse(shape == "changing", "changed", shape))
tib1 %<>% mutate(shape = ifelse(shape == "flare", "fireball", shape))
tib1 %<>% mutate(shape = ifelse(shape == "round", "sphere", shape))
tib1 %<>% mutate(shape = ifelse(shape == "pyramid", "triangle", shape))
```

```
tib1 %<>% mutate(shape = ifelse(shape == "delta", "chevron", shape))
tib1 %<>% mutate(shape = ifelse(shape == "dome", "other", shape))
tib1 %<>% mutate(shape = ifelse(shape == "hexagon", "other", shape))
tib1 %<>% mutate(shape = ifelse(shape == "flash", "light", shape))
tib1 %<>% mutate(shape = ifelse(shape == "crescent", "other", shape))
tib1$shape %>% table()
```

```
## .
##   changed   chevron     cigar    circle      cone     cross  cylinder   diamond
##      1710       835      1885      6527       267       208      1163      1062
##      disk       egg  fireball formation     light     other      oval rectangle
##      4714       688      5018      2061     15272      4968      3273      1151
##    sphere  teardrop  triangle   unknown
##      4643       663      6877      4859
```

We can see now all the different shapes of UFO spottings are now part of common categories or the other category.

Check and clean numeric columns

Let's mutate the duration column into an integer data type.

```
tib1 %<>% mutate(duration = as.integer(duration))
```

Let's get rid of outliers in this data.

```
Q1 <- quantile(tib1$duration, 0.25, na.rm = T)
Q3 <- quantile(tib1$duration, 0.75, na.rm = T)
IQR <- Q3 - Q1
tib1 %<>% filter(duration >= Q1 - 1.5 * IQR, duration <= Q3 + 1.5 * IQR)
```

Now let's check the latitude and longitude columns. Let's first make sure they are within the correct bounds.

```
tib1 %<>% filter(latitude >= -90, latitude <= 90, longitude >= -180, longitude <= 180)
```

Now let's make sure all the values of longitude and latitude that have 0 as their values are assigned to NA (it's extremely unlikely all those UFO spottings happen at some arbitrary point above the Atlantic Ocean).

```
tib1 %<>% mutate(latitude = ifelse(latitude == 0 & longitude == 0, NA, latitude))
tib1 %<>% mutate(longitude = ifelse(latitude == 0 & longitude == 0, NA, longitude))
```

Convert date columns into data datatype

Let's convert datetime to a dttm datatype.

```
tib1 %<>% mutate(datetime = as.POSIXct(.$datetime, format = "%m/%d/%Y %H:%M"))
```

Convert appropriate columns into factors

Let's convert the appropriate character columns into factors.

```
tib1 %<>% mutate(shape = as.factor(shape))
tib1 %<>% mutate(city = as.factor(city))
tib1 %<>% mutate(country = as.factor(country))
tib1 %<>% mutate(`state/province` = as.factor(`state/province`))
```

Drop NA values

```
tib1 %<>% drop_na()
tib1 %>% glimpse()
```

```
## Rows: 26,149
## Columns: 8
## $ datetime         <dttm> 1944-01-01 12:00:00, 1957-01-01 21:00:00, 1966-01-01~
## $ country          <fct> United States, United States, United States, United S~
## $ city             <fct> san diego, dallas, flagstaff, burlington, wheaton, se~
## $ `state/province` <fct> ca, tx, az, nj, md, wa, al, tn, wa, ny, az, co, ca, c~
## $ shape            <fct> cigar, unknown, disk, circle, circle, triangle, unkno~
## $ duration         <int> 180, 20, 600, 1200, 60, 30, 120, 30, 300, 120, 300, 1~
## $ latitude         <dbl> 32.71528, 32.78333, 35.19806, 40.07111, 39.03972, 47.~
## $ longitude        <dbl> -117.15639, -96.80000, -111.65056, -74.86528, -77.055~
```

This is our final tib1. We can see now the columns are the appropriate data type, have only clean data and no NA values. Note that the initial CSV file was already in tidy format so we do not need to pivot the data into long format.

Data Wrangling Plan : File 2

Iteration 1

Phase 1

- Read CSV file into R (specify column types)
- Drop unnecessary columns and duplicate rows
- Change column names if necessary
- Identify uids and verify they are unique
- Sort by uids

Phase 2

Read CSV files into R (specify column types)

```
tib2 <- read_csv("ufo_sighting_data.csv", col_types = "ccccciccdd")
```

```
## Warning: One or more parsing issues, call `problems()` on your data frame for details,
## e.g.:
##   dat <- vroom(...)
##   problems(dat)
```

```
tib2 %>% glimpse()
```

```
## Rows: 80,332
## Columns: 11
## $ Date_time                      <chr> "10/10/1949 20:30", "10/10/1949 21:00"~
## $ city                           <chr> "san marcos", "lackland afb", "chester~
## $ `state/province`               <chr> "tx", "tx", NA, "tx", "hi", "tn", NA, ~
## $ country                        <chr> "us", NA, "gb", "us", "us", "us", "gb"~
## $ UFO_shape                      <chr> "cylinder", "light", "circle", "circle~
## $ length_of_encounter_seconds    <int> 2700, 7200, 20, 20, 900, 300, 180, 120~
## $ described_duration_of_encounter <chr> "45 minutes", "1-2 hrs", "20 seconds",~
## $ description                    <chr> "This event took place in early fall a~
## $ date_documented                <chr> "4/27/2004", "12/16/2005", "1/21/2008"~
## $ latitude                       <dbl> 29.88306, 29.38421, 53.20000, 28.97833~
## $ longitude                      <dbl> -97.941111, -98.581082, -2.916667, -96~
```

Drop unnecessary columns and duplicate rows

```
tib2 %<>% select(!c("description", "date_documented", "described_duration_of_encounter"))
tib2 %<>% distinct()
```

Change column names if necessary

Let's first rename all the column names to lower case.

```
tib2 %<>% rename_with(tolower)
```

Let's check the column names.

```
tib2 %>% colnames()
```

```
## [1] "date_time"                  "city"
## [3] "state/province"             "country"
## [5] "ufo_shape"                  "length_of_encounter_seconds"
## [7] "latitude"                   "longitude"
```

Let's change the column names to something more fitting.

```
tib2 %<>% rename("datetime" = "date_time", "shape" = "ufo_shape", "duration" = "length_of_encounter_sec
```

Now let's check that they are renamed.

```
tib2 %>% colnames()
```

```
## [1] "datetime"       "city"           "state/province" "country"
## [5] "shape"          "duration"       "latitude"       "longitude"
```

Identify uids and verify they are unique

```
tib2 %>%
  count(datetime, country) %>%
  filter(n > 1)
```

```
## # A tibble: 0 x 3
## # i 3 variables: datetime <chr>, country <chr>, n <int>
```

We see they are unique as they don't appear more than one.

Sort by uids

```
tib2 %<>% select(datetime, country, everything()) %>% arrange(datetime, country)
```

Let's get a final view of tib2 after our first iteration of changes. We can see the tibble has only relevant columns for our project, no duplicate rows, has a uniform naming convention for columns and is sorted by unique identifiers.

```
tib2 %>% glimpse()
```

```
## Rows: 69,586
## Columns: 8
## $ datetime         <chr> "1/1/1910 24:00", "1/1/1944 12:00", "1/1/1947 17:00",~
## $ country          <chr> "us", "us", NA, NA, NA, NA, "us", "us", "us", "us", "~
## $ city             <chr> "kirksville (near)", "san diego", "manama (bahrain)",~
## $ `state/province` <chr> "mo", "ca", NA, "tx", "ne", "ia", "wv", "tx", "tx", "~
## $ shape            <chr> "disk", "cigar", "circle", "cigar", NA, "circle", "di~
## $ duration         <int> 120, 180, 300, 300, 600, 30, 10, 20, 2, 10800, 30, 30~
## $ latitude         <dbl> 40.19472, 32.71528, 26.21667, 33.66013, 41.49254, 42.~
## $ longitude        <dbl> -92.58306, -117.15639, 50.58333, -97.71556, -99.90181~
```

Iteration 2

Phase 1

- Check and clean character columns
- Check and clean numeric columns
- Convert date columns into date datatype
- Convert appropriate columns into factors
- Drop NA values
- Join tib1 and tib2

Phase 2

Check and clean character columns

Let's start with the datetime column and remove all the rows that do not fit the normal date format (month day year hour minute) that is present in that column.

```
tib2 %<>% filter(!is.na(mdy_hm(datetime)))
```

Now let's see the unique entries in the country column.

```
tib2$country %>% unique()
```

```
## [1] "us" NA   "ca" "gb" "au" "de"
```

Let's rename them to lowercase.

```
tib2$country %<>% tolower()
```

Now let's view the entries in the city column.

```
tib2$city %>% glimpse()
```

```
##  chr [1:69586] "kirksville (near)" "san diego" "manama (bahrain)" ...
```

Let's rename the entries to lowercase. We see a lot of the entries end with something in brackets. Let's see what those bracket entries say.

```
tib2$city %<>% tolower()
bracketentries <- str_extract(tib2$city, "\\(.*\\)$")
bracketentries %>% glimpse()
```

```
##  chr [1:69586] "(near)" NA "(bahrain)" NA "(rural)" NA NA NA NA NA NA NA ...
```

We see either it's somewhat irrelevant information, or it contains the name of the country the city is located in. Let's remove the brackets if the text inside does not contain a country using the maps and countrycode libraries, or if the brackets contains the name of the country, let's make the country entry for that row the country in brackets.

```
tib2 %<>% mutate(country = ifelse(is.na(country), str_extract(city, "\\((.*)\\)$"), country))
tib2 %<>% mutate(country = str_remove_all(country, "\\(|\\)"))
tib2 %<>% mutate(city = str_replace(city, "\\s*\\(.*?\\)", ""))
```

This effectively cleans both the city and country columns. Now let's check one more time to make sure all the entries in the city and country columns are legitimate places, or make them NA otherwise. Let's also turn abbreviations of country names into the actual country names (again using the countrycode library).

```
tib2$country %<>% tolower()
tib2 %<>% mutate(country = ifelse(country %in% tolower(world.cities$country) |
  country %in% tolower(codelist$iso2c), country, NA))
tib2 %<>% mutate(city = ifelse(city %in% tolower(world.cities$name), city, NA))
tib2 %<>% mutate(country = countryname(country, destination = tolower("country.name.en")))
```

```
## Warning: There was 1 warning in `mutate()`.
## i In argument: `country = countryname(country, destination =
##   tolower("country.name.en"))`.
## Caused by warning:
## ! Some values were not matched unambiguously: au, ca, canary islands, de, gb, ne
```

Let's do the same thing for the state column.

```
tib2$`state/province` %>% unique()
```

```
##  [1] "mo" "ca" NA   "tx" "ne" "ia" "wv" "in" "fl" "mt" "il" "ar" "az" "nj" "la"
## [16] "md" "mi" "on" "wa" "al" "tn" "ny" "ga" "co" "ct" "id" "ma" "ok" "nc" "ky"
## [31] "ks" "sc" "ri" "va" "pa" "hi" "wi" "oh" "nv" "or" "bc" "me" "ak" "nb" "nh"
## [46] "nm" "de" "qc" "ut" "ms" "vt" "sa" "nt" "pr" "pq" "mn" "ab" "ns" "nd" "sd"
## [61] "sk" "mb" "wy" "dc" "nf" "yt" "pe" "yk"
```

It looks good. Now let's check the last character column which is the shape column.

```
tib2$shape %>% table()
```

```
## .
##   changed  changing   chevron     cigar    circle      cone  crescent     cross
##         1      1709       828      1885      6527       267         2       208
##  cylinder     delta   diamond      disk      dome       egg  fireball     flare
##      1163         7      1062      4714         1       688      5017         1
##     flash formation   hexagon     light     other      oval   pyramid rectangle
##      1150      2061         1     14122      4964      3273         1      1151
##     round    sphere  teardrop  triangle   unknown
##         2      4641       663      6876      4859
```

Some of these entries look similar so let's adjust them so they match better.

```
tib2 %<>% mutate(shape = ifelse(shape == "changing", "changed", shape))
tib2 %<>% mutate(shape = ifelse(shape == "flare", "fireball", shape))
tib2 %<>% mutate(shape = ifelse(shape == "round", "sphere", shape))
tib2 %<>% mutate(shape = ifelse(shape == "pyramid", "triangle", shape))
tib2 %<>% mutate(shape = ifelse(shape == "delta", "chevron", shape))
tib2 %<>% mutate(shape = ifelse(shape == "dome", "other", shape))
tib2 %<>% mutate(shape = ifelse(shape == "hexagon", "other", shape))
tib2 %<>% mutate(shape = ifelse(shape == "flash", "light", shape))
tib2 %<>% mutate(shape = ifelse(shape == "crescent", "other", shape))
tib2$shape %>% table()
```

```
## .
##   changed   chevron     cigar    circle      cone     cross  cylinder   diamond
##      1710       835      1885      6527       267       208      1163      1062
##      disk       egg  fireball formation     light     other      oval rectangle
##      4714       688      5018      2061     15272      4968      3273      1151
##    sphere  teardrop  triangle   unknown
##      4643       663      6877      4859
```

We can see now all the different shapes of UFO spottings are now part of common categories or the other category.

Check and clean numeric columns

Let's mutate the duration column into an integer data type.

```
tib2 %<>% mutate(duration = as.integer(duration))
```

Let's get rid of outliers in this data.

```
Q1 <- quantile(tib2$duration, 0.25, na.rm = T)
Q3 <- quantile(tib2$duration, 0.75, na.rm = T)
IQR <- Q3 - Q1
tib2 %<>% filter(duration >= Q1 - 1.5 * IQR, duration <= Q3 + 1.5 * IQR)
```

Now let's check the latitude and longitude columns. Let's first make sure they are within the correct bounds.

```
tib2 %<>% filter(latitude >= -90, latitude <= 90, longitude >= -180, longitude <= 180)
```

Now let's make sure all the values of longitude and latitude that have 0 as their values are assigned to NA (it's extremely unlikely all those UFO spottings happen at some arbitrary point above the Atlantic Ocean).

```
tib2 %<>% mutate(latitude = ifelse(latitude == 0 & longitude == 0, NA, latitude))
tib2 %<>% mutate(longitude = ifelse(latitude == 0 & longitude == 0, NA, longitude))
```

Convert date columns into data datatype

Let's convert datetime to a dttm datatype and also make a new column called time that only contains the time and not date so that we can use it for plotting and data visualization.

```
tib2 %<>% mutate(datetime = as.POSIXct(.$datetime, format = "%m/%d/%Y %H:%M"))
```

Convert appropriate columns into factors

Let's convert the appropriate character columns into factors.

```
tib2 %<>% mutate(shape = as.factor(shape))
tib2 %<>% mutate(city = as.factor(city))
tib2 %<>% mutate(country = as.factor(country))
tib2 %<>% mutate(`state/province` = as.factor(`state/province`))
```

Drop NA values

```
tib2 %<>% drop_na()
```

Join tib1 and tib2

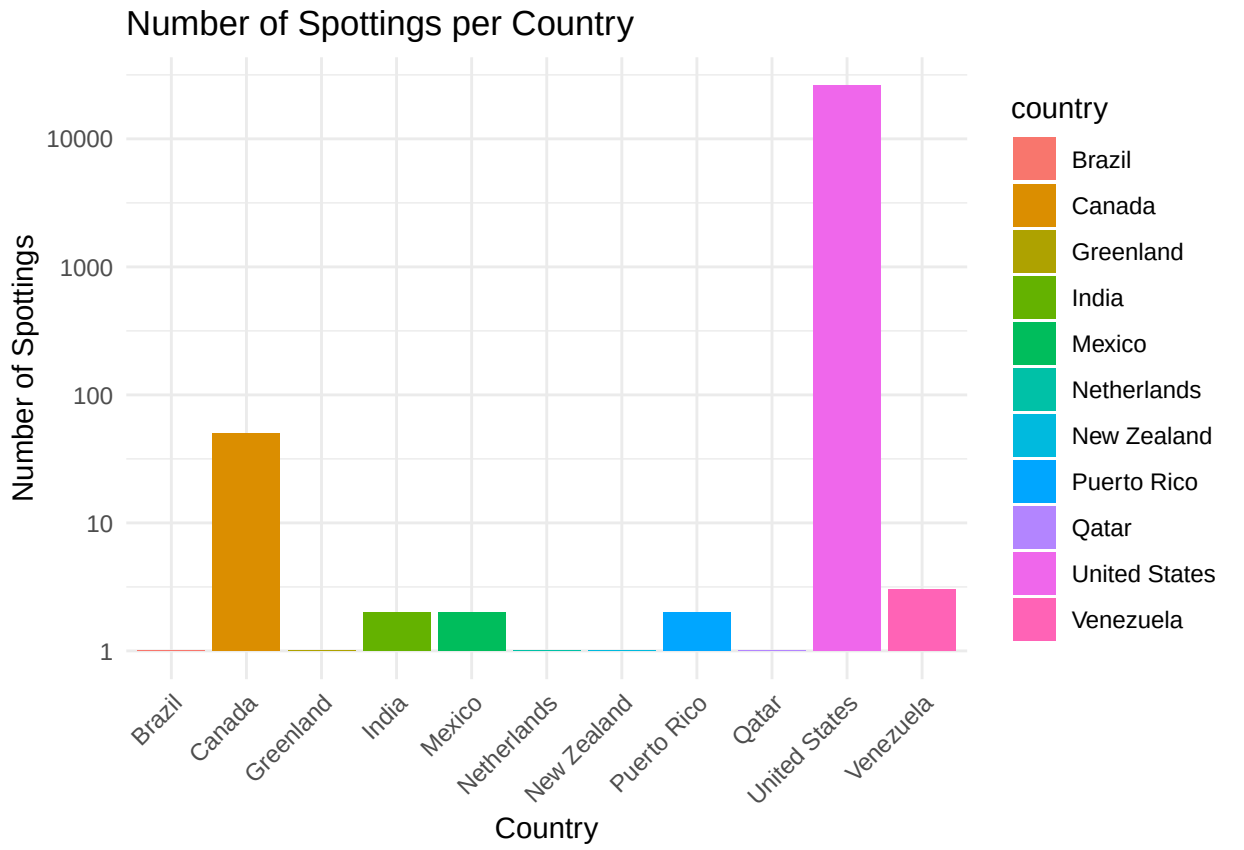
Let's also make sure to get rid of duplicate rows from the two tibbles.

```
tib <- bind_rows(tib1, tib2)
tib %<>% distinct()
```

This is our final tibble. We can see the join worked, and it is now ready to use for data visualization.

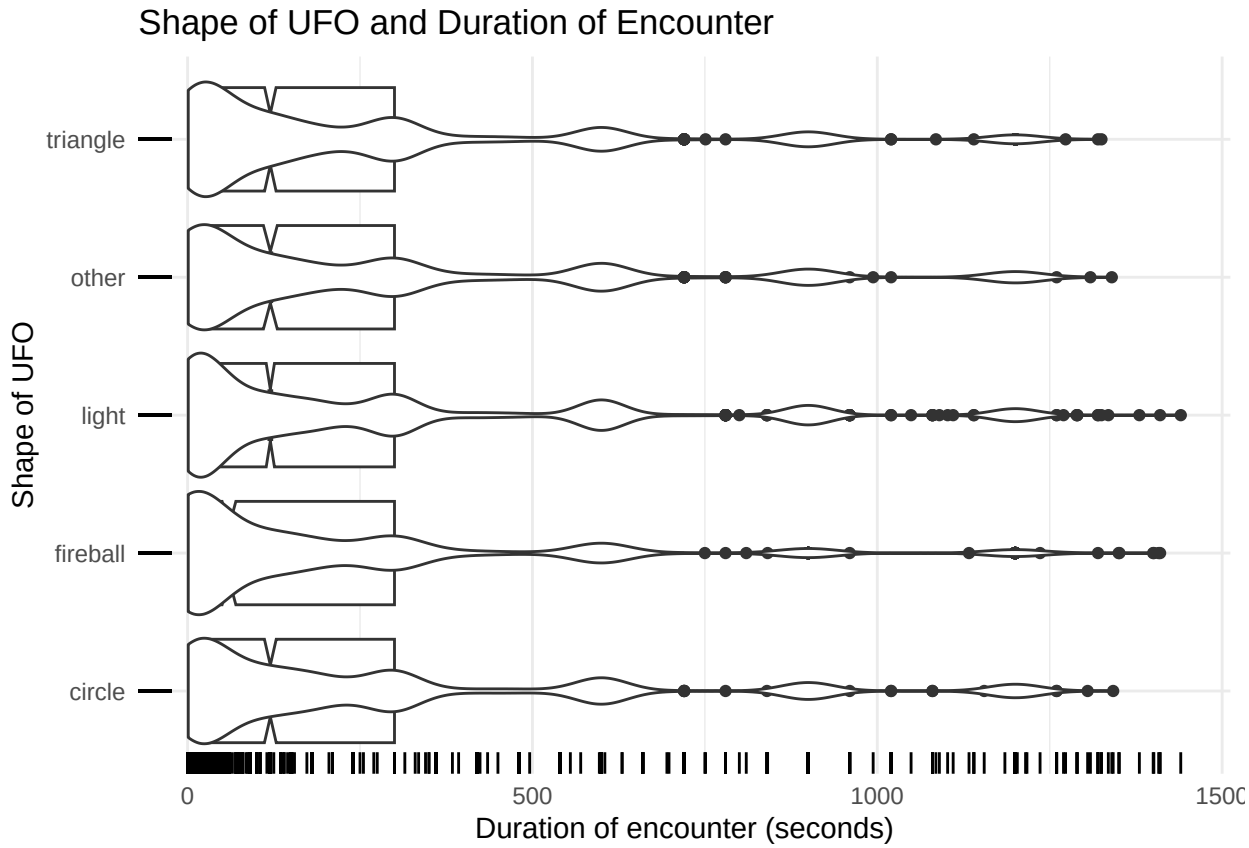
Results/Discussion

```
p1 <- tib %>% ggplot(aes(x = country, fill = country)) +
  geom_bar() +
  labs(x = "Country", y = "Number of Spottings", title = "Number of Spottings per Country") +
  scale_y_continuous(trans = "log10") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, vjust = 1, hjust = 1))
p1
```



The first thing I immediately wanted to find out is what country had the most UFO spottings. This plot shows that U.S.A has by far the most UFO spottings out of any other country in the world. Perhaps this could be due to factors such as its larger population and land area, technological advancement in aviation and space exploration, a strong media and popular culture that features stories about UFOs, and a reporting culture that encourages and collects reports of unusual phenomena. While not all UFO sightings are evidence of extraterrestrial activity as there are often natural or man-made explanations for them, it was still interesting to see the vast majority of the world's UFO spottings happen to the United States.

```r
topshapes <- tib %>%
  group_by(shape) %>%
  summarize(count = n()) %>%
  arrange(desc(count)) %>%
  head(5) %>%
  ungroup()
p2 <- tib %>%
  filter(shape %in% topshapes$shape) %>%
  ggplot(aes(x=duration, y= shape)) +
  geom_boxplot(notch = T) +
  geom_rug() +
  geom_violin() +
  labs(x = "Duration of encounter (seconds)", y = "Shape of UFO", title = "Shape of UFO and Duration of
  theme_minimal()
p2
```

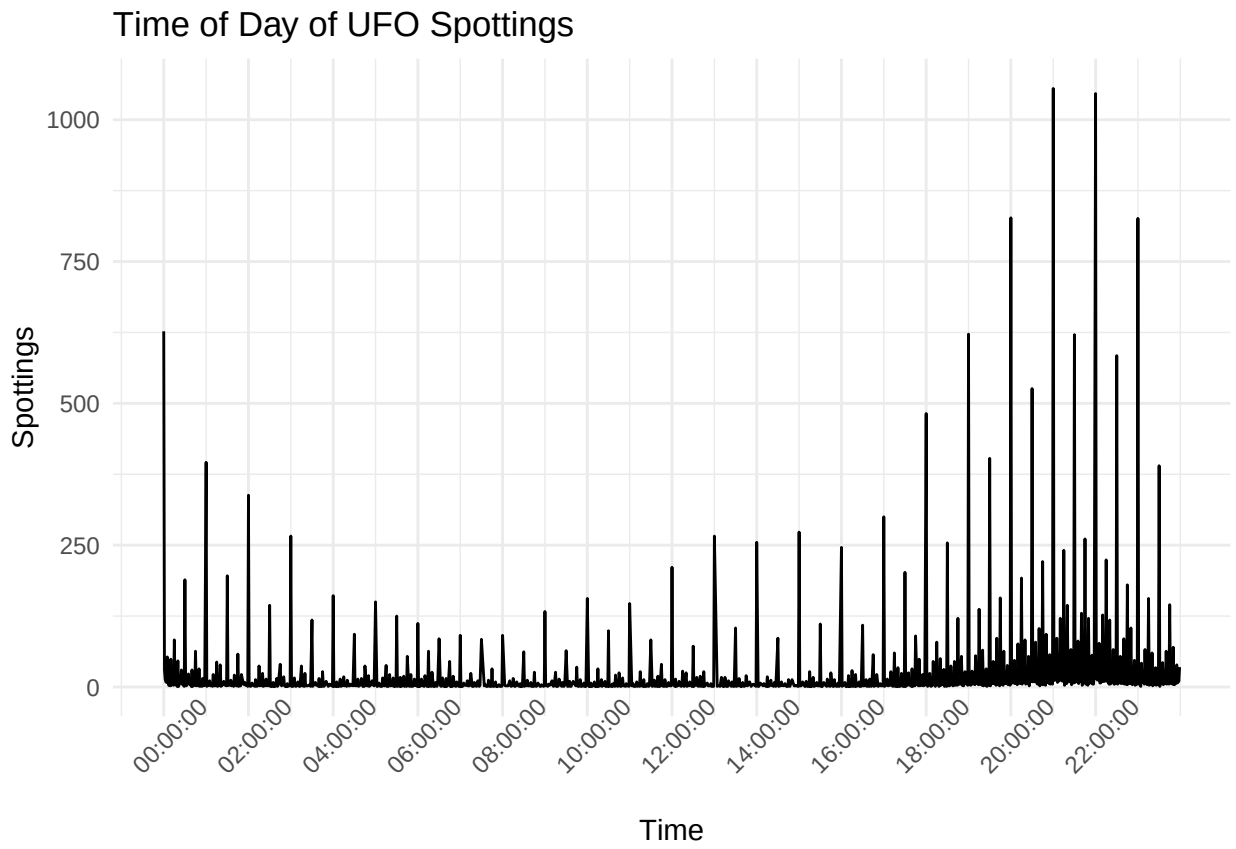


We can see that despite there being a multitude of shapes that UFOs are described as, the duration that they are spotted for is often quite short. We can see that even the 5 most common shapes that UFOs are described as often still are not seen for a very long time. These sightings could be brief because UFOs are typically unknown and unfamiliar objects that can appear suddenly, move quickly, and exhibit unusual behavior. Since they are often seen at a distance or high altitudes in the sky, it would be difficult for the naked human eye to track or observe them. Human perception and attention, reporting delays and other explanations could also contribute to the rarity of UFO sightings.

```
p3 <- tib %>% mutate(time = as_hms(datetime)) %>%
  group_by(time) %>%
  mutate(n = n()) %>%
  ungroup() %>%
  ggplot(aes(x = time, y=n)) +
  geom_line() +
  labs(x = "Time", y = "Spottings", title = "Time of Day of UFO Spottings") +
  scale_x_time(breaks = as.hms(c('0:00:00','2:00:00', '4:00:00','6:00:00', '8:00:00','10:00:00', '12:00
  theme_minimal() +
  theme(axis.text.x=element_text(angle = 45))
```

```
## Warning: ‘as.hms()‘ was deprecated in hms 0.5.0.
## i Please use ‘as_hms()‘ instead.
## This warning is displayed once every 8 hours.
## Call ‘lifecycle::last_lifecycle_warnings()‘ to see where this warning was
## generated.
```

```
p3
```



We can see that the majority of UFO spottings happen during the later hours of the day when it is likely dark outside. This is likely due to reduced visibility, altered human perception, cultural factors, and reporting bias that contributes to an increased likelihood of noticing unusual lights or objects in the sky, and interpreting them as UFOs. If those same encounters happened during the daytime where natural light is plentiful, I would believe that many UFO spottings would instead be planes or clouds or other hallucinations the human mind makes up due to the lack of light.

Conclusion

Despite the large number of reported UFO sightings from around the world, there are several reasons why conclusive proof of extraterrestrial existence has not been found yet. We have even shown that UFO spottings can be attributed to various factors. Many sightings can be explained by natural phenomena or ordinary objects, human perception, reduced visibility at night, cultural influences, and reporting bias. Additionally, cultural beliefs and media portrayals of UFOs, as well as reporting bias, can shape people's expectations and interpretations of sightings. Even how short and brief the vast majority of encounters are speak to how untrustworthy a human's account of a UFO spotting really is. We have found patterns in where these spottings take place, the duration and shape of these spottings and even the time of day of when people see them, yet all of these point out to us that it is far more likely than not that these UFO spottings can be chalked up to rational and logical explanations instead of definitive proof of extraterrestrials.