MSDR Capstone R Notebook

```
# Load necessary libraries
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(lubridate)
```

```
##
## Attaching package: 'lubridate'
```

```
## The following objects are masked from 'package:base':
##
##     date, intersect, setdiff, union
```

```
library(ggplot2)
library(leaflet)
library(htmltools)
```

```
# Load the dataset
noaa_data <- read.csv("noaa.csv")
```

```
# Check data types and missing values
str(noaa_data)
```

```
## 'data.frame':    37331 obs. of  6 variables:
##  $ time          : chr  "2023-02-17 09:37:34.868000+00:00" "2023-02-16 05:37:05.138000
+00:00" "2023-02-15 18:10:10.060000+00:00" "2023-02-15 06:38:09.034000+00:00" ...
##  $ latitude      : num  -6.6 -15.1 12.3 -40.5 45.1 ...
##  $ longitude     : num  132.1 167 123.9 174.6 23.2 ...
##  $ deaths        : num  38.6 36 20.1 74.3 10 ...
##  $ mag           : num  6.1 5.6 6.1 5.7 5.6 6.1 5.6 5.9 5.5 5.5 ...
##  $ locationsource: chr  "us" "us" "us" "us" ...
```

```
summary(noaa_data)
```

```
##      time              latitude         longitude           deaths
##  Length:37331       Min.   :-77.080   Min.   :-180.00   Min.   : -4.00
##  Class :character   1st Qu.:-16.520   1st Qu.: -75.81   1st Qu.: 15.00
##  Mode  :character   Median :  1.153   Median :  98.58   Median : 28.50
##                     Mean   :  5.458   Mean   :  38.88   Mean   : 58.58
##                     3rd Qu.: 33.786   3rd Qu.: 143.35   3rd Qu.: 41.00
##                     Max.   : 87.199   Max.   : 180.00   Max.   :700.00
##                                                         NA's   :134
##       mag        locationsource
##  Min.   :5.500   Length:37331
##  1st Qu.:5.600   Class :character
##  Median :5.800   Mode  :character
##  Mean   :5.949
##  3rd Qu.:6.140
##  Max.   :9.500
##
```

```
colnames(noaa_data)
```

```
## [1] "time"           "latitude"       "longitude"      "deaths"
## [5] "mag"            "locationsource"
```

```
# Define eq_clean_data function
eq_clean_data <- function(data) {
  data <- data %>%
    # Convert the time column to Date class by extracting the date part
    mutate(DATE = as.Date(time, format = "%Y-%m-%d"),
           LATITUDE = as.numeric(latitude),
           LONGITUDE = as.numeric(longitude),
           deaths = as.integer(deaths),  # Convert deaths to integer
           LOCATION_NAME = eq_location_clean(locationsource)) %>%
    # Optionally drop the original columns if they are no longer needed
    select(-time, -locationsource, -latitude, -longitude)

  return(data)
}
```

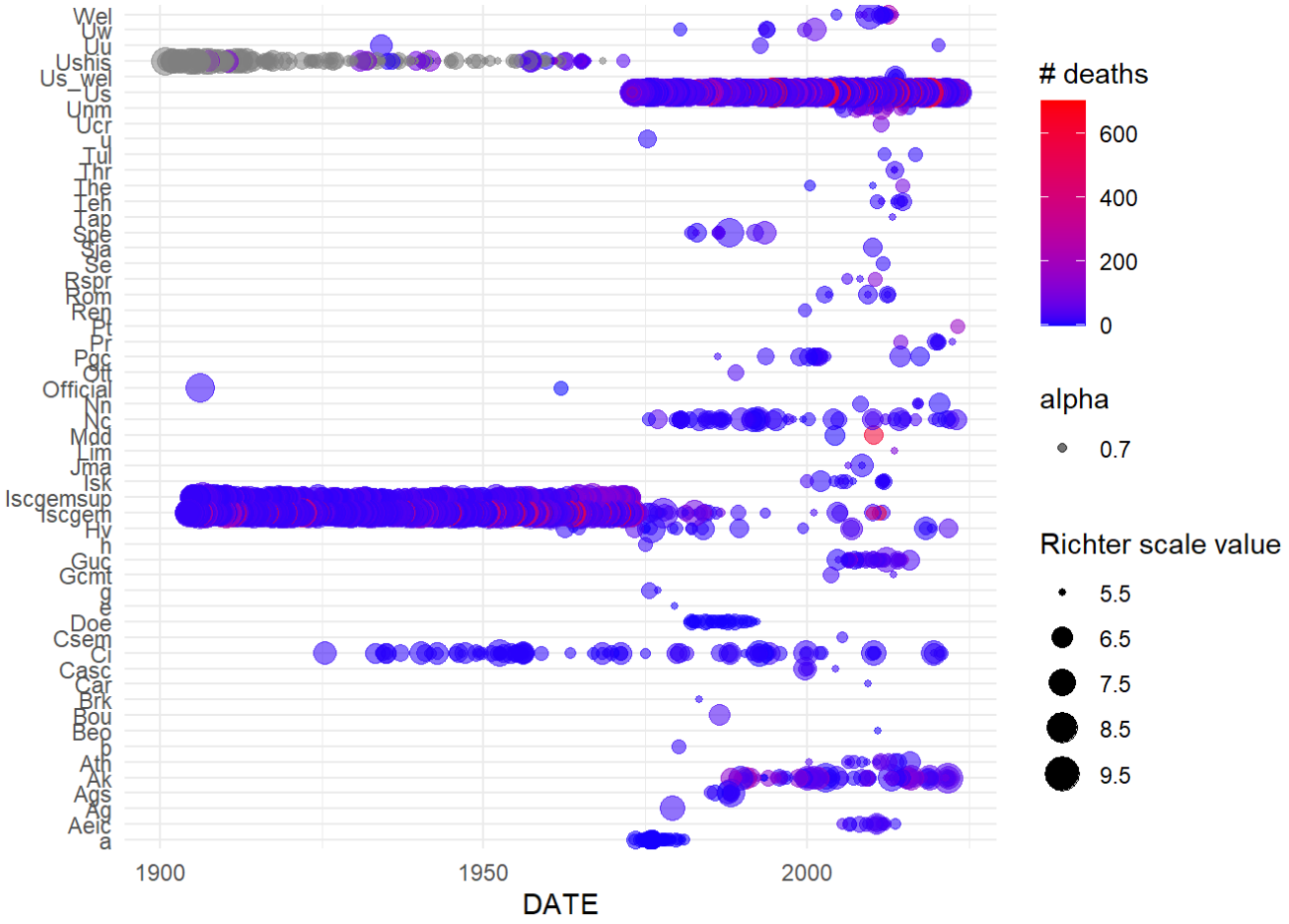
```
# Define eq_location_clean function
eq_location_clean <- function(location_name) {
  cleaned_name <- location_name %>%
    sub("^[^:]+:\\s*", "", .) %>%
    tolower() %>%
    tools::toTitleCase()

  return(cleaned_name)
}
```

```
# Apply data cleaning
cleaned_data <- eq_clean_data(noaa_data)
```

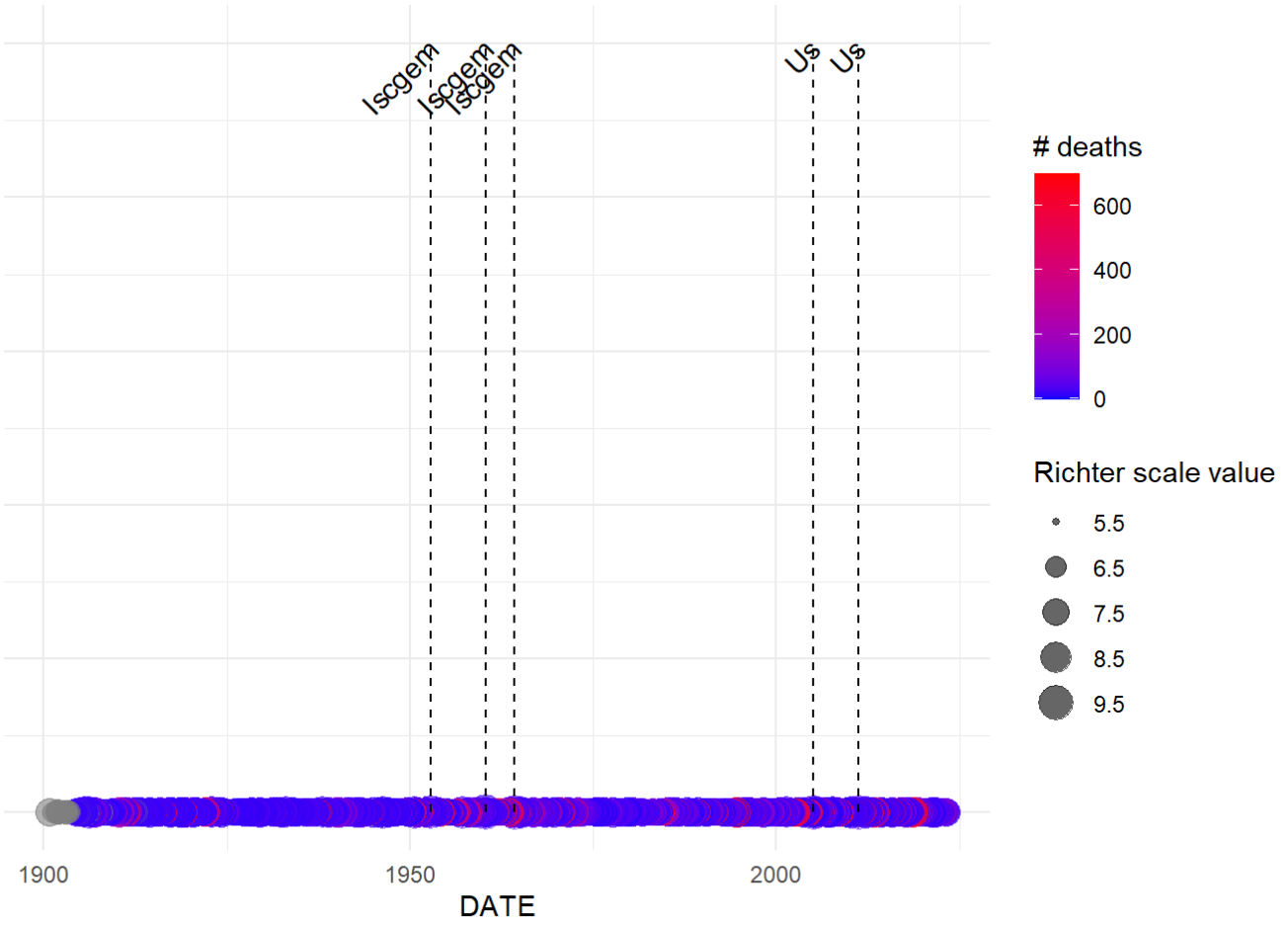
```r
# Check the cleaned data
str(cleaned_data)
```

```
## 'data.frame':    37331 obs. of  6 variables:
##  $ deaths       : int  38 36 20 74 10 374 10 48 10 35 ...
##  $ mag          : num  6.1 5.6 6.1 5.7 5.6 6.1 5.6 5.9 5.5 5.5 ...
##  $ DATE         : Date, format: "2023-02-17" "2023-02-16" ...
##  $ LATITUDE     : num  -6.6 -15.1 12.3 -40.5 45.1 ...
##  $ LONGITUDE    : num  132.1 167 123.9 174.6 23.2 ...
##  $ LOCATION_NAME: chr  "Us" "Us" "Us" "Us" ...
```

```r
# Define a custom geom_timeline function
geom_timeline <- function(data, x, y = NULL, color = "blue", size = 2, alpha = 0.5, ...) {
  ggplot(data, aes(x = {{ x }}, y = {{ y }},
                   color = {{ color }},
                   size = {{ size }},
                   alpha = {{ alpha }})) +
    geom_point() +
    scale_size_continuous(name = "Richter scale value") +
    scale_color_gradient(low = "blue", high = "red", name = "# deaths") +
    theme_minimal() +
    theme(axis.title.y = element_blank()) +
    labs(x = "DATE")
}

# Usage with the cleaned data
# Assuming `cleaned_data` has columns `DATE`, `mag`, `DEATHS`, and optionally `LOCATION_NA
ME`
p <- geom_timeline(cleaned_data,
                   x = DATE,
                   y = LOCATION_NAME,  # Use this if you want separate timelines for each
location
                   color = deaths,
                   size = mag,
                   alpha = 0.7)

# Display the plot
print(p)
```

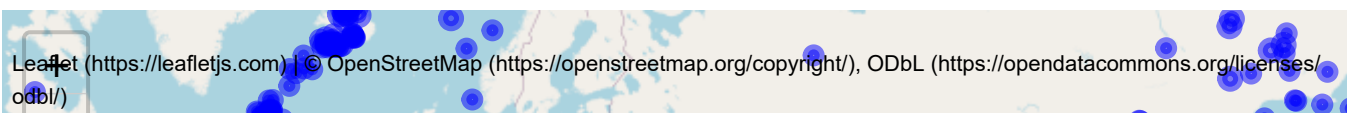

Wel
Uw
Uu
Ushis
Us_wel
Us_Us
Unm
Ucr
u
Tul
Thr
The
Teh
Tap
Spe
Sja
Se
Rspr
Rom
Ren
Pt
Pr
Pgc
Ott
Official
Nn
Nc
Mdd
Lim
Jma
Isk
Iscgemsup
Iscgem
Hv
h
Guc
Gcmt
g
e
Doe
Csem
Cl
Casc
Car
Brk
Bou
Bep
b
Ath
Ak
Ags
Ag
Aeic
a
1900
1950
2000
DATE
deaths
600
400
200
0
alpha
0.7
Richter scale value
5.5
6.5
7.5
8.5
9.5

```r
# Define a custom geom_timeline_label function
geom_timeline_label <- function(data, x, label, n_max = 10, ...) {
  # Filter the top n_max earthquakes by magnitude
  top_earthquakes <- data %>%
    arrange(desc(mag)) %>%
    head(n_max)

  ggplot(data, aes(x = {{ x }}, y = 0)) +
    geom_point(aes(size = mag, color = deaths), alpha = 0.6) +
    geom_segment(data = top_earthquakes,
                 aes(x = {{ x }}, xend = {{ x }},
                     y = 0, yend = 0.5),
                 linetype = "dashed") +
    geom_text(data = top_earthquakes,
              aes(x = {{ x }},
                  y = 0.5,
                  label = {{ label }}),
              angle = 45, hjust = 1) +
    scale_size_continuous(name = "Richter scale value") +
    scale_color_gradient(low = "blue", high = "red", name = "# deaths") +
    theme_minimal() +
    theme(axis.title.y = element_blank(),
          axis.text.y = element_blank(),
          axis.ticks.y = element_blank()) +
    labs(x = "DATE")
}

# Usage with the cleaned data
# Assuming `cleaned_data` has columns `DATE`, `mag`, `DEATHS`, and `LOCATION_NAME`
p <- geom_timeline_label(cleaned_data,
                         x = DATE,
                         label = LOCATION_NAME,
                         n_max = 5)

# Display the plot
print(p)
```




```
# Define the eq_map function
eq_map <- function(data, annot_col) {
  # Check if the required columns are in the data frame
  required_cols <- c("LATITUDE", "LONGITUDE", "mag", annot_col)
  if (!all(required_cols %in% colnames(data))) {
    stop("Data must contain LATITUDE, LONGITUDE, mag, and the specified annotation colum
n.")
  }

  # Create the map
  leaflet(data) %>%
    addTiles() %>%
    addCircles(lng = ~LONGITUDE, lat = ~LATITUDE,
               radius = ~mag * 10000, # Scale the radius for visibility
               popup = as.formula(paste0("~", annot_col)),
               color = "blue",
               fillOpacity = 0.5) %>%
    setView(lng = mean(data$LONGITUDE, na.rm = TRUE),
            lat = mean(data$LATITUDE, na.rm = TRUE),
            zoom = 2)
}
```

```
# Example usage with a hypothetical data frame 'earthquake_data'
eq_map(data = cleaned_data, annot_col = "LOCATION_NAME")
```



```r
# Define the eq_create_label function
eq_create_label <- function(data) {
  data %>%
    rowwise() %>%
    mutate(
      label = paste0(
        if (!is.na(LOCATION_NAME)) paste0("<b>Location:</b> ", eq_location_clean(LOCATION_
NAME), "<br>") else "",
        if (!is.na(mag)) paste0("<b>Magnitude:</b> ", mag, "<br>") else "",
        if (!is.na(deaths)) paste0("<b>Total deaths:</b> ", deaths) else ""
      )
    ) %>%
    pull(label)
}
```

```r
# Example usage with a hypothetical data frame 'earthquake_data'
labels <- eq_create_label(cleaned_data)

# Using these labels in the eq_map function
# eq_map(data = cleaned_data, annot_col = labels)
```