Placement

Yicheng Shen

Aug 6, 2024

Note: the github repo for this work is: https://github.com/sheny2/Placement

Q1

Task 1: Read the data Scrape the wikipedia page on natural disasters

```
url <- "https://en.wikipedia.org/wiki/List_of_natural_disasters_by_death_toll"

webpage <- read_html(url)

tables <- webpage %>% html_nodes("table.wikitable")

# 20th and 21st century data are in the 2nd and 3rd table of this page
table_20th <- tables[[2]] %>% html_table(fill = TRUE)
table_21st <- tables[[3]] %>% html_table(fill = TRUE)

disasters <- rbind(table_20th, table_21st) %>% as_tibble() # merge into one

head(disasters)
```

```
## # A tibble: 6 x 6
##    Year `Death toll` Event                       `Countries affected` Type  Date
##   <int> <chr>        <chr>                       <chr>                <chr> <chr>
## 1  1900 6,000-12,000 1900 Galveston hurricane    United States        Trop~ Sept~
## 2  1901 9,500        1901 eastern United State~ United States        Heat~ June~
## 3  1902 29,000       1902 eruption of Mount Pe~ Martinique           Volc~ Apri~
## 4  1903 3,500        1903 Manzikert earthquake  Turkey               Eart~ Apri~
## 5  1904 400          1904 Sichuan earthquake    China                Eart~ Augu~
## 6  1905 20,000+      1905 Kangra earthquake     India                Eart~ Apri~
```

Task 2: Clean the data Convert the death toll to numbers using the midpoints when a range is given and the bound when an upper or lower bound is given.

```
disasters_clean = disasters
# remove unnecessary footnote, words and symbols
disasters_clean$`Death toll` = disasters$`Death toll` %>%
                                          str_remove_all(",") %>%
                                          str_remove_all("\\[.*?\\]") %>%
                                          str_remove_all("[a-zA-Z]") %>%
                                          str_remove_all("\\(.*?\\)") %>%
                                          str_trim()
```

```r
# which entries need special attention to clean
index_to_clean = is.na(as.numeric(gsub("," ,"", disasters_clean$`Death toll`)))
disasters_clean$`Death toll`[index_to_clean]
```

```
##  [1] "6000-12000"     "20000+"         "12000-15000"    "75000-82000"
##  [5] "6000-8000"      "50000-220000"   "942-1900"       "29978-32610"
##  [9] "2000-10000"     "258707-273407"  "50000-100000+"  "105385-142800"
## [13] "4112+"          "3257-3800"      "2000-8000"      "422499-4000000"
## [17] "3103+"          "6865-9300"      "10700-12000"    "5000+"
## [21] "715+"           "32700-32968"    "2824-5000"      "10000-110000"
## [25] "1023+"          "300000-500000"  "2175-2204"      "8210+"
## [29] "26000-240000"   "242419-655000"  "10000-50000"    "15000-25000"
## [33] "2633-5000"      "25000-50000"    "35000-45000"    "17126-18373"
## [37] "700-800"        "13805-20023"    "86000-87351"    "5749-5778"
## [41] "100000-316000"  "3951+"          "59259-62013"    "320-600"
```

```r
# use regular expression here
convert_death_toll <- function(toll) {
  if (str_detect(toll, "\\d+\\s*-\\s*\\d+")) {
    nums <- str_extract_all(toll, "\\d+")[[1]] %>% as.numeric()
    return(mean(nums))
  } else if (str_detect(toll, "\\d+\\s*-\\s*\\d+")) {
    nums <- str_extract_all(toll, "\\d+")[[1]] %>% as.numeric()
    return(mean(nums))
  } else if (str_detect(toll, "\\d+")) {
    return(as.numeric(str_extract(toll, "\\d+")))
  }
}

death_toll_clean = round(sapply(disasters_clean$`Death toll`, convert_death_toll))
death_toll_clean[index_to_clean] # check the results of those irregular ones
```

```
##     6000-12000          20000+    12000-15000    75000-82000       6000-8000
##           9000           20000          13500          78500           7000
##   50000-220000        942-1900    29978-32610     2000-10000  258707-273407
##         135000            1421          31294           6000         266057
##  50000-100000+  105385-142800          4112+      3257-3800      2000-8000
##          75000          124092           4112           3528           5000
## 422499-4000000           3103+      6865-9300    10700-12000          5000+
##        2211250            3103           8082          11350           5000
##           715+     32700-32968      2824-5000   10000-110000          1023+
##            715           32834           3912          60000           1023
##  300000-500000       2175-2204          8210+   26000-240000  242419-655000
##         400000            2190           8210         133000         448710
##    10000-50000     15000-25000      2633-5000    25000-50000    35000-45000
##          30000           20000           3816          37500          40000
##    17126-18373         700-800    13805-20023    86000-87351       5749-5778
##          17750             750          16914          86676           5764
##  100000-316000           3951+    59259-62013        320-600
##         208000            3951          60636            460
```

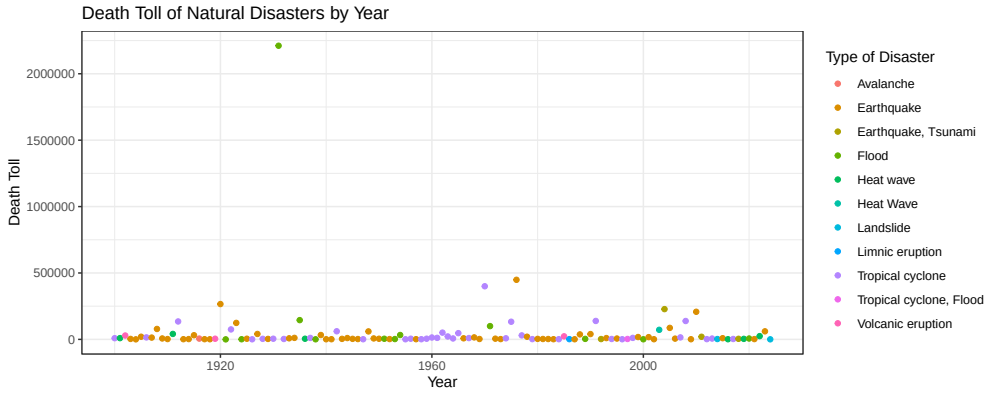
```
disasters_clean$`Death toll` = death_toll_clean
disasters_clean
```

```
## # A tibble: 125 x 6
##     Year `Death toll` Event                      `Countries affected` Type  Date
##    <int>        <dbl> <chr>                      <chr>                <chr> <chr>
##  1  1900         9000 1900 Galveston hurricane   United States        Trop~ Sept~
##  2  1901         9500 1901 eastern United Stat~ United States        Heat~ June~
##  3  1902        29000 1902 eruption of Mount P~ Martinique           Volc~ Apri~
##  4  1903         3500 1903 Manzikert earthquake Turkey               Eart~ Apri~
##  5  1904          400 1904 Sichuan earthquake    China              Eart~ Augu~
##  6  1905        20000 1905 Kangra earthquake     India              Eart~ Apri~
##  7  1906        15000 1906 Hong Kong typhoon     Hong Kong,China    Trop~ Sept~
##  8  1907        13500 1907 Qaratog earthquake    Uzbekistan         Eart~ Octo~
##  9  1908        78500 1908 Messina earthquake    Italy              Eart~ Dece~
## 10  1909         7000 1909 Borujerd earthquake   Iran               Eart~ Janu~
## # i 115 more rows
```

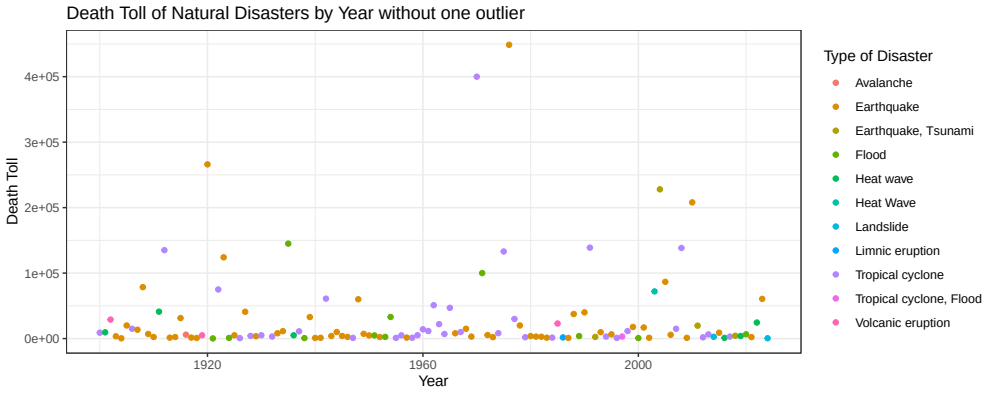
Task 3: Plot the data

Disasters have happened frequently throughout the years. While at one time floods caused one of the highest death tolls, it should be noted that earthquakes are often the leading ones causing fatalities. Both topical cyclones and earthquakes seem to be the most frequent and deadly disasters during the studied period.

```
ggplot(disasters_clean, aes(x = Year, y = `Death toll`, color = Type)) +
  geom_point() +
  labs(title = "Death Toll of Natural Disasters by Year",
       x = "Year",
       y = "Death Toll",
       color = "Type of Disaster")
```




```
ggplot(disasters_clean %>% filter(Event != "1931 China floods"),
       aes(x = Year, y = `Death toll`, color = Type)) +
  geom_point() +
  labs(title = "Death Toll of Natural Disasters by Year without one outlier",
       x = "Year",
       y = "Death Toll",
       color = "Type of Disaster")
```



Q2

First we derive the form of gradient by computing the derivative w.r.t b:

$$\frac{\partial||y-bx||^2}{\partial b} = 2(\frac{\partial \sum_i^n (y_i - bx_i)}{\partial b}) = 2(\frac{-\sum_i^n x_i(y_i - bx_i)}{n})$$

```
# write the gradient descent function
gradient_descent <- function(x, y, learning_rate, num_iter) {
  b <- 0 # start at 0, or somewhere
  n <- length(x)

  for (i in 1:num_iter) {
    gradient <- -2 * sum(x * (y - b * x)) / n # compute gradient
    b <- b - learning_rate * gradient # update here
  }
  return(b)
}

# Test the function using randomly generated normal vectors
set.seed(8848)
n <- 100
x <- rnorm(n)
b <- 5
y <- b * x + rnorm(n)

S <- 1000 # long enough

b_grad <- gradient_descent(x, y, learning_rate = 0.05, num_iter = S) # choose e to be 0.05

# Compare
b_solution <- x %*% y / (norm(x, type="2"))^2
b_solution %>% as.vector()
```

```
## [1] 5.015827
```

```
b_grad # success!
```

```
## [1] 5.015827
```

```r
# To test different learning rates
test_learning_rates <- function(x, y, b, learning_rates, num_iter) {
  results <- data.frame(learning_rate = c(), b_est = c())

  for (e in learning_rates) {
    b_est <- gradient_descent(x, y, e, num_iter)
    results <- rbind(results, data.frame(learning_rate = e, b_est = b_est))
  }

  return(results)
}
```

Findings: The algorithm can perform well as long as we choose a reasonable value for the learning rate.

We can see from below that the learning rate should not be too small. A too-small step size can make it inefficient for the algorithm to explore and reach convergence or get stuck at a local min instead of global mean, thus unable to find the correct solution.

Meanwhile, we also do not want the step size to be too large at each update, which could cause our estimate to oscillate too much and miss the optimal point as well.

```r
learning_rates <- seq(0.001, 0.1, by = 0.001)
results <- test_learning_rates(x, y, b_true, learning_rates, num_iter = S)

# Plot the results
ggplot(results, aes(x = learning_rate, y = b_est)) +
  geom_line() +
  labs(title = "Estimate vs Learning Rate", x = "Learning Rate", y = "Estimate")
```