Exploration and Analysis of NOAA Storm Database

akshayamrit

25/08/2019

Exploration and Analysis of NOAA Storm Database to Identify Most Harmful Events With Respect to Population Health and Economic Consequences

Storm Data is an official publication of the National Oceanic and Atmospheric Administration (NOAA) which documents:
1. The occurrence of storms and other significant weather phenomena having sufficient intensity to cause loss of life, injuries, significant property damage, and/or disruption to commerce.
2. Rare, unusual, weather phenomena that generate media attention, such as snow flurries in South Florida or the San Diego coastal area.
3. Other significant meteorological events, such as record maximum or minimum temperatures or precipitation that occur in connection with another event.
According to NOAA, the data recording starts from Jan 1950. Data which we are using contains data from Jan 1950 to Nov 2011. Storm Data contains Date, Time, Event, Place of occurance, Damage, etc. We will be using this database to identify most harmful events with respect to population health as well as economic consequences.

Data Processing:

Before we get into the analysis, we will check whether we have the data or not. If the data is not present in our working directory, it will be downloaded.

```
if(!file.exists("StormData.csv.bz2")) {
      download.file(url = "https://d396qusza40orc.cloudfront.net/repdata%2Fdata%2FStormData.csv.bz2",
                    destfile = "StormData.csv.bz2", method = "curl")
}
if(!file.exists("StromDataDoc.pdf")){
      download.file(url = "https://d396qusza40orc.cloudfront.net/repdata%2Fpeer2_doc%2Fpd01016005curr.p
                    destfile = "StromDataDoc.pdf", method = "curl")
}
```

We made sure that we have the data so we can start reading the data now. We will be reading the pdf documentation as well to extract categorized event types.

```
library(data.table)
```

```
## Warning: package 'data.table' was built under R version 3.6.1
```

```
readTime <- proc.time()
stormData <- fread("StormData.csv.bz2")
```

```
## Registered S3 method overwritten by 'R.oo':
##   method        from
##   throw.default R.methodsS3
```

```
proc.time()-readTime
```

```
##    user  system elapsed
##   45.25    2.40   52.16
```

```
rm(readTime)
```

We have loaded the data but we need to know what the data looks like and isolate the data needed to answer our questions.

```
dim(stormData)
```

```
## [1] 902297     37
```

```
str(stormData)
```

```
## Classes 'data.table' and 'data.frame':   902297 obs. of  37 variables:
##  $ STATE__   : num  1 1 1 1 1 1 1 1 1 1 ...
##  $ BGN_DATE  : chr  "4/18/1950 0:00:00" "4/18/1950 0:00:00" "2/20/1951 0:00:00" "6/8/1951 0:00:00" .
##  $ BGN_TIME  : chr  "0130" "0145" "1600" "0900" ...
##  $ TIME_ZONE : chr  "CST" "CST" "CST" "CST" ...
##  $ COUNTY    : num  97 3 57 89 43 77 9 123 125 57 ...
##  $ COUNTYNAME: chr  "MOBILE" "BALDWIN" "FAYETTE" "MADISON" ...
##  $ STATE     : chr  "AL" "AL" "AL" "AL" ...
##  $ EVTYPE    : chr  "TORNADO" "TORNADO" "TORNADO" "TORNADO" ...
##  $ BGN_RANGE : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ BGN_AZI   : chr  "" "" "" "" ...
##  $ BGN_LOCATI: chr  "" "" "" "" ...
##  $ END_DATE  : chr  "" "" "" "" ...
##  $ END_TIME  : chr  "" "" "" "" ...
##  $ COUNTY_END: num  0 0 0 0 0 0 0 0 0 0 ...
##  $ COUNTYENDN: logi  NA NA NA NA NA NA ...
##  $ END_RANGE : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ END_AZI   : chr  "" "" "" "" ...
##  $ END_LOCATI: chr  "" "" "" "" ...
##  $ LENGTH    : num  14 2 0.1 0 0 1.5 1.5 0 3.3 2.3 ...
##  $ WIDTH     : num  100 150 123 100 150 177 33 33 100 100 ...
##  $ F         : int  3 2 2 2 2 2 2 1 3 3 ...
##  $ MAG       : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ FATALITIES: num  0 0 0 0 0 0 0 0 1 0 ...
##  $ INJURIES  : num  15 0 2 2 2 6 1 0 14 0 ...
##  $ PROPDMG   : num  25 2.5 25 2.5 2.5 2.5 2.5 2.5 25 25 ...
##  $ PROPDMGEXP: chr  "K" "K" "K" "K" ...
##  $ CROPDMG   : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ CROPDMGEXP: chr  "" "" "" "" ...
##  $ WFO       : chr  "" "" "" "" ...
##  $ STATEOFFIC: chr  "" "" "" "" ...
##  $ ZONENAMES : chr  "" "" "" "" ...
##  $ LATITUDE  : num  3040 3042 3340 3458 3412 ...
##  $ LONGITUDE : num  8812 8755 8742 8626 8642 ...
##  $ LATITUDE_E: num  3051 0 0 0 0 ...
```

```
##  $ LONGITUDE_: num  8806 0 0 0 0 ...
##  $ REMARKS   : chr  "" "" "" "" ...
##  $ REFNUM    : num  1 2 3 4 5 6 7 8 9 10 ...
##  - attr(*, ".internal.selfref")=<externalptr>
```

```
length(unique(stormData$EVTYPE))
```

```
## [1] 985
```

By running the above code we observe that:
* We have several columns which are not needed for our analysis.
* We have several rows which does not contain any data on population health or economic consequences.
* There are **985** types of events listed in the data provided but there are only **48** events classified in the documentation.

We need to fix the above problems before moving on to the analysis part. To do so, we need to:
* Remove unwanted columns from the table. We will remove all the columns which won't be necessary for fnding out the loss of or damage of human lives and loss of property.
* Remove rows containing redundant information. These rows include the data for events which did not cause any damage.
* Fix event names in data table to match with the event names provided in the documentation. To do this, we will be reading the PDF documentation which we have already downloaded in the first step. Now, as we did have to read the documentation to understand the data table, we started by looking up for the exact section where we will find the names of all the event type ie. '2.1.1 Storm Data Event Table'. After finding this value, we have extracted the lines which contained the event names and modified it according to our needs. Now, we compare these values to the event types present in the data table to reduce the number of unique variables as much as possible. We have used 'amatch' function to perform this activity. This function uses fuzzy logic to approximately match provided strings. We can change how close the match should be by adjusting 'max.distance' argument but for our current problem, we found out that it works best if we leave it at default which is '0.1'. If we increase it any further, it tends to match values which are not actually the same.

```
# Removing unwanted columns from table:
library(tidyverse)
```

```
## Warning: package 'tidyverse' was built under R version 3.6.1
```

```
## -- Attaching packages ---------------------------------------------------------------------------
```

```
## v ggplot2 3.2.1     v purrr   0.3.2
## v tibble  2.1.3     v dplyr   0.8.3
## v tidyr   0.8.3     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.4.0
```

```
## Warning: package 'ggplot2' was built under R version 3.6.1
```

```
## Warning: package 'tidyr' was built under R version 3.6.1
```

```
## Warning: package 'dplyr' was built under R version 3.6.1
```

```
## Warning: package 'forcats' was built under R version 3.6.1
```

```
## -- Conflicts -------------------------------------------------------------------------------------
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
stormData <- stormData %>% select(c(BGN_DATE, COUNTY, COUNTYNAME, STATE, EVTYPE, FATALITIES, INJURIES, I

# Removing redundant rows:
stormData <- stormData %>% filter(FATALITIES != 0 | INJURIES != 0 | PROPDMG != 0 | CROPDMG != 0)

# Changing format of Columns:
library(lubridate)
```

```
## Warning: package 'lubridate' was built under R version 3.6.1
```

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

```
## The following object is masked from 'package:base':
##
##     date
```

```
stormData$BGN_DATE <-  mdy_hms(stormData$BGN_DATE)
stormData$EVTYPE <- str_squish(stormData$EVTYPE)
stormData$PROPDMGEXP <- as.factor(str_to_upper(stormData$PROPDMGEXP))
stormData$CROPDMGEXP <- as.factor(str_to_upper(stormData$CROPDMGEXP))

# Using EXP columns to modify DMG and removing them.
stormData <- stormData %>%
  mutate(PROPDMG = if_else(PROPDMGEXP %in% c("", "?", "-"), PROPDMG * 0, PROPDMG)) %>%
  mutate(PROPDMG = if_else(PROPDMGEXP %in% "+", PROPDMG * 1, PROPDMG)) %>%
  mutate(PROPDMG = if_else(PROPDMGEXP %in% "H", PROPDMG * 100, PROPDMG)) %>%
  mutate(PROPDMG = if_else(PROPDMGEXP %in% "K", PROPDMG * 1000, PROPDMG)) %>%
  mutate(PROPDMG = if_else(PROPDMGEXP %in% "M", PROPDMG * 1000000, PROPDMG)) %>%
  mutate(PROPDMG = if_else(PROPDMGEXP %in% "B", PROPDMG * 1000000000, PROPDMG))

stormData <- stormData %>%
  mutate(CROPDMG = if_else(CROPDMGEXP %in% c("", "?", "-"), CROPDMG * 0, CROPDMG)) %>%
  mutate(CROPDMG = if_else(CROPDMGEXP %in% "+", CROPDMG * 1, CROPDMG)) %>%
  mutate(CROPDMG = if_else(CROPDMGEXP %in% "H", CROPDMG * 100, CROPDMG)) %>%
  mutate(CROPDMG = if_else(CROPDMGEXP %in% "K", CROPDMG * 1000, CROPDMG)) %>%
  mutate(CROPDMG = if_else(CROPDMGEXP %in% "M", CROPDMG * 1000000, CROPDMG)) %>%
  mutate(CROPDMG = if_else(CROPDMGEXP %in% "B", CROPDMG * 1000000000, CROPDMG))

stormData <- stormData %>% select(-c(PROPDMGEXP, CROPDMGEXP))

# Extracting list of events from Documentation.
library(pdftools)
```

```
## Warning: package 'pdftools' was built under R version 3.6.1
```

```
documentation.events <- pdf_text("StromDataDoc.pdf") %>% read_lines()
firstLine <- grep("2.1.1   Storm Data Event Table", documentation.events)
documentation.events <- documentation.events[(firstLine + 2): (firstLine + 25)]
rm(firstLine)
documentation.events <- documentation.events %>%
  str_replace_all(pattern = "Z$", replacement = "") %>%
  str_replace_all(pattern = "C$", replacement = "") %>%
  str_replace_all(pattern = "M$", replacement = "") %>%
  str_replace_all(pattern = " Z ", replacement = ".") %>%
  str_replace_all(pattern = " C ", replacement = ".") %>%
  str_replace_all(pattern = " M ", replacement = ".") %>%
  strsplit(split = ".", fixed = TRUE) %>%
  unlist() %>%
  str_squish()

# Modifying event types in stormData table to match with values present in documentation
for(i in 1:length(documentation.events)){
  stormData[agrepl(pattern = documentation.events[i], x = stormData$EVTYPE, ignore.case = TRUE),5] = do
}
rm(i)
```

Data Analysis:

We have processed the data as much as possible, now we will be analyzing the data to answer our questions. Before starting that, let's look at how the data looks after all the processing.

```
str(stormData)
```

```
## 'data.frame':    254633 obs. of  9 variables:
##  $ BGN_DATE  : POSIXct, format: "1950-04-18" "1950-04-18" ...
##  $ COUNTY    : num  97 3 57 89 43 77 9 123 125 57 ...
##  $ COUNTYNAME: chr  "MOBILE" "BALDWIN" "FAYETTE" "MADISON" ...
##  $ STATE     : chr  "AL" "AL" "AL" "AL" ...
##  $ EVTYPE    : chr  "Tornado" "Tornado" "Tornado" "Tornado" ...
##  $ FATALITIES: num  0 0 0 0 0 0 0 0 1 0 ...
##  $ INJURIES  : num  15 0 2 2 2 6 1 0 14 0 ...
##  $ PROPDMG   : num  25000 2500 25000 2500 2500 2500 2500 2500 25000 25000 ...
##  $ CROPDMG   : num  0 0 0 0 0 0 0 0 0 0 ...
```

```
length(unique(stormData$EVTYPE))
```

```
## [1] 234
```

We were able to reduce the number of event types to **234** but we know that the number of events should be **48**. It is difficult to rename every event type manually so we will try to figure out whether it's necessary to answer our question. We need to figure out how many of those events need to be taken care of to be of any importance to us. ### Identify Most Harmful Events With Respect to Population Health. To figure out whether we need to worry about the typos which are still present in the table, we will extract the necessary data from the table which we have already created. To do that, we need to follow these steps:
1. Group the data by event types and get their corresponding damage on human health.
2. Group every event which is not listed in the documentation into one category and compare its value to others.

```
# Creating table to be used to answer our first query.
stormData.harm <- stormData %>%
  group_by(year = year(BGN_DATE), EVTYPE) %>%
  summarize(pop.harm = sum(FATALITIES)+sum(INJURIES)) %>%
  arrange(year)

# Determining whether typos in events will have impact on our result
stormData.harm.analysis <- stormData.harm %>%
  group_by(EVTYPE) %>%
  summarize(pop.harm = sum(pop.harm)) %>%
  arrange(desc(pop.harm))
stormData.harm.analysis <- stormData.harm.analysis %>%
  mutate(EVTYPE = if_else(!(str_to_upper(EVTYPE) %in% str_to_upper(documentation.events)), "Others", EV
  group_by(EVTYPE) %>%
  summarize(pop.harm = sum(pop.harm)) %>%
  arrange(desc(pop.harm))
```

Identify events that has the greatest economic consequences.

These steps will be performed again, but we will focus economic consequences to answer our second part of the question.

```
# Creating table to be used to answer our first query.
stormData.damage <- stormData %>%
  group_by(year = year(BGN_DATE), EVTYPE) %>%
  summarize(eco.cq = sum(PROPDMG)+sum(CROPDMG)) %>%
  arrange(year)

# Determining whether typos in events will have impact on our result
stormData.damage.analysis <- stormData.damage %>%
  group_by(EVTYPE) %>%
  summarize(eco.cq = sum(eco.cq)) %>%
  arrange(desc(eco.cq))
stormData.damage.analysis <- stormData.damage.analysis %>%
  mutate(EVTYPE = if_else(!(str_to_upper(EVTYPE) %in% str_to_upper(documentation.events)), "Others", EV
  group_by(EVTYPE) %>%
  summarize(eco.cq = sum(eco.cq)) %>%
  arrange(desc(eco.cq))
```

Results:

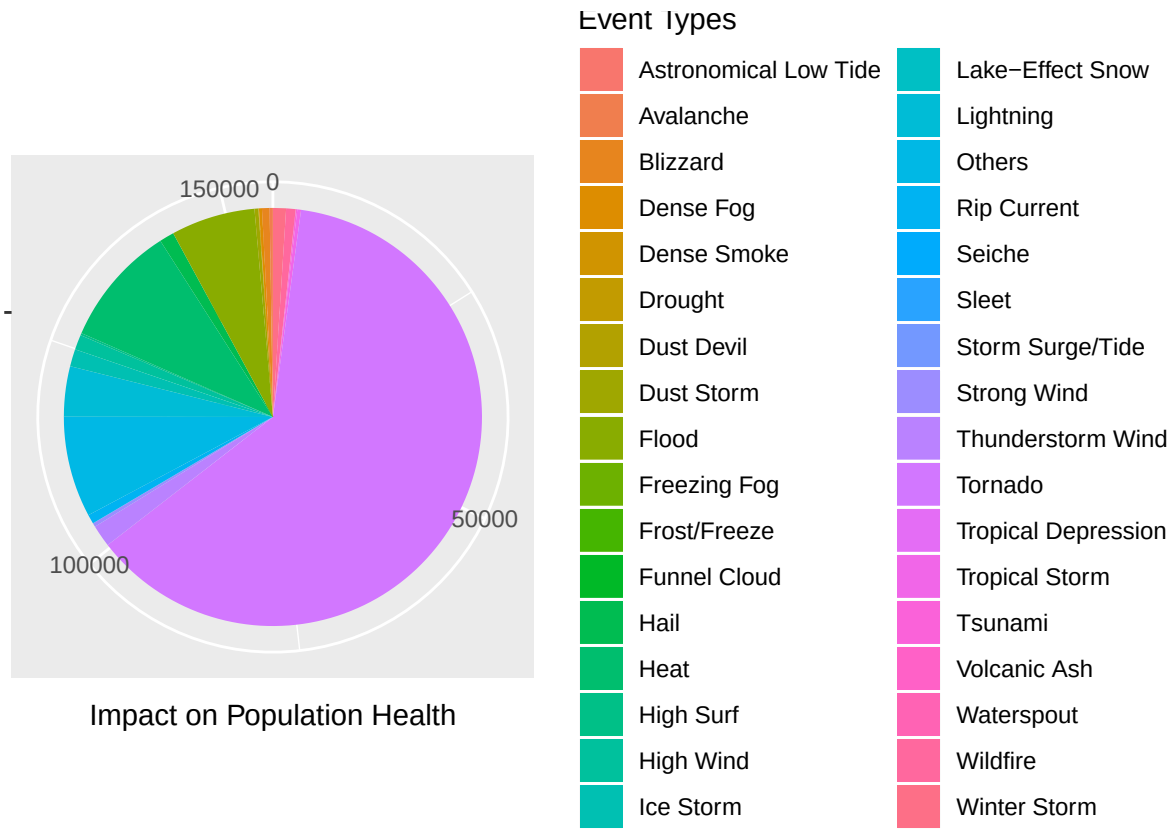
Which types of event is most harmful to population health?

```
head(stormData.harm.analysis, n = 10)
```

```
## # A tibble: 10 x 2
##    EVTYPE            pop.harm
##    <chr>                <dbl>
##  1 Tornado              97068
##  2 Heat                 14607
```

```
##  3 Others               12144
##  4 Flood                10127
##  5 Lightning             6049
##  6 Thunderstorm Wind     2649
##  7 Ice Storm             2081
##  8 High Wind             1801
##  9 Hail                  1766
## 10 Winter Storm          1570
```

```
# Visualizing table
g.harm <- ggplot(data = stormData.harm.analysis, aes(x = "", y = pop.harm, fill = EVTYPE)) + geom_bar(s
g.harm + coord_polar("y", start = 0) + labs(x = "", y = "Impact on Population Health", fill = "Event Ty
```



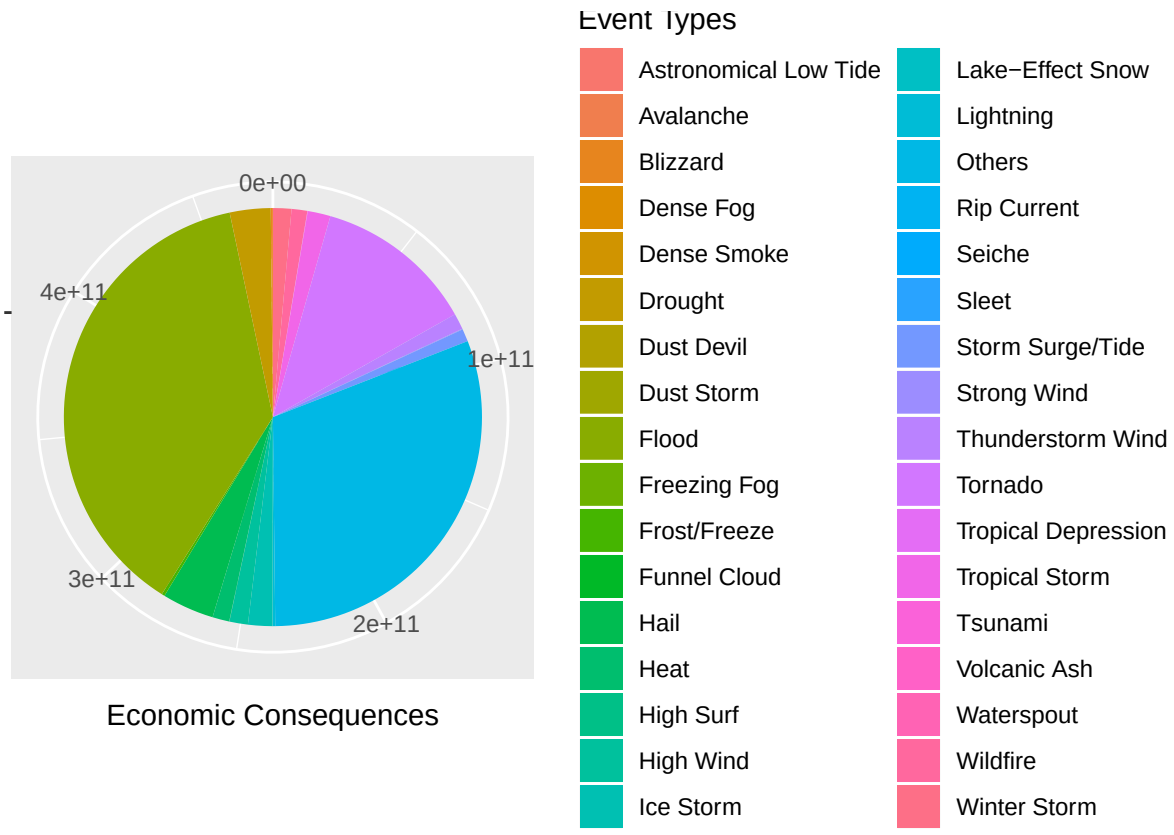
Using the pie chart and the table above, we can conlcude that Tornado causes most harm to the population health. For the period of year 1950 to year 2011, Tornado has harmed **9,7068** lives accross all states. ### Which type of event has the greatest economic consequece?

```
head(stormData.damage.analysis, n = 10)
```

```
## # A tibble: 10 x 2
##    EVTYPE                eco.cq
##    <chr>                  <dbl>
##  1 Flood          179909849807.
##  2 Others         146462556735
##  3 Tornado         59010559546.
```

```
##  4 Hail            19160928866.
##  5 Drought         15018927780
##  6 Ice Storm        8968141360
##  7 Tropical Storm   8409286550
##  8 High Wind        6787142925
##  9 Winter Storm     6781441251
## 10 Heat             6225900932.
```

```
# Visualizing table
g.eco <- ggplot(data = stormData.damage.analysis, aes(x = "", y = eco.cq, fill = EVTYPE)) + geom_bar(sta
g.eco + coord_polar("y", start = 0) + labs(x = "", y = "Economic Consequences", fill = "Event Types")
```



Using the table and pie chart above, we can conclude that Flood has the greatest impact on economy. For the period of year 1950 to year 2011, Flood has damaged property worth **$179,909,849,807** across all states.