The Relative Impact of Adverse Weather Events on Health, Crop and Property Losses

Rusell Glynn

11/1/2021

Synopsis:

Adverse weather events are tracked by the U.S National Oceanic and Atmospheric Administration. This information is stored in a publicly available storm database. Using this raw data, we will answer the following two research questions:

1. What types of events (EVTYPE) are most harmful with respect to population health?

2 Across the US, what types of events have the greatest economic impact as measured by property and crop damages?

The raw data has 902297 observations stored in 37 variables. Since only data entered after January, 1996, has complete information on all weather event types (EVTYPE), we will only observations entered after that date. Further, only 8 of the 37 variables have information relevant to the research questions and we will subset the data to only include information in these 8 variables. The population health impact of events will be measured by 'FATALATIES' and 'INJURIES'. The economic impact of weather events is recorded as property and crop dollar losses in 4 separate variables (PROPDMG, PROPDMGEXP, CROPDMG, CROPDMEXP). Data entry errors in the EVTYPE variable complicated the analysis. Using the NOAA data dictionary to inform our decisions, we reclassified EVTYPE aberrant entries into the official listed categories as completely as possible. Only 350 of 201318 records were not classified in the final data set.

Data Processing

Load R 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(readr)
library(lubridate)
```

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

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

```
library(naniar)
library(stringdist)
library(fuzzyjoin)
library(stringr)
```

Copy file and store locally. Load file into memory as 'storm'.

```
url <- 'https://d396qusza40orc.cloudfront.net/repdata%2Fdata%2FStormData.csv.bz2'
dest <- "C:/Users/Russell/projects/storm/storm_data.csv.bz2"
download.file(url,destfile = dest)
storm <- read_csv('storm_data.csv.bz2')
```

```
##
## -- Column specification --------------------------------------------------------
## cols(
##   .default = col_double(),
##   BGN_DATE = col_character(),
##   BGN_TIME = col_character(),
##   TIME_ZONE = col_character(),
##   COUNTYNAME = col_character(),
##   STATE = col_character(),
##   EVTYPE = col_character(),
##   BGN_AZI = col_logical(),
##   BGN_LOCATI = col_logical(),
##   END_DATE = col_logical(),
##   END_TIME = col_logical(),
##   COUNTYENDN = col_logical(),
##   END_AZI = col_logical(),
##   END_LOCATI = col_logical(),
##   PROPDMGEXP = col_character(),
##   CROPDMGEXP = col_logical(),
##   WFO = col_logical(),
##   STATEOFFIC = col_logical(),
##   ZONENAMES = col_logical(),
##   REMARKS = col_logical()
## )
## i Use `spec()` for the full column specifications.
```

Explore storm data and variable names.

```
str(storm)
```

```
## spec_tbl_df [902,297 x 37] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
##  $ STATE__   : num [1:902297] 1 1 1 1 1 1 1 1 1 1 ...
##  $ BGN_DATE  : chr [1:902297] "4/18/1950 0:00:00" "4/18/1950 0:00:00" "2/20/1951 0:00:00" "6/8/1951 0
##  $ BGN_TIME  : chr [1:902297] "0130" "0145" "1600" "0900" ...
##  $ TIME_ZONE : chr [1:902297] "CST" "CST" "CST" "CST" ...
##  $ COUNTY    : num [1:902297] 97 3 57 89 43 77 9 123 125 57 ...
##  $ COUNTYNAME: chr [1:902297] "MOBILE" "BALDWIN" "FAYETTE" "MADISON" ...
##  $ STATE     : chr [1:902297] "AL" "AL" "AL" "AL" ...
##  $ EVTYPE    : chr [1:902297] "TORNADO" "TORNADO" "TORNADO" "TORNADO" ...
##  $ BGN_RANGE : num [1:902297] 0 0 0 0 0 0 0 0 0 0 ...
##  $ BGN_AZI   : logi [1:902297] NA NA NA NA NA NA ...
##  $ BGN_LOCATI: logi [1:902297] NA NA NA NA NA NA ...
##  $ END_DATE  : logi [1:902297] NA NA NA NA NA NA ...
##  $ END_TIME  : logi [1:902297] NA NA NA NA NA NA ...
##  $ COUNTY_END: num [1:902297] 0 0 0 0 0 0 0 0 0 0 ...
##  $ COUNTYENDN: logi [1:902297] NA NA NA NA NA NA ...
##  $ END_RANGE : num [1:902297] 0 0 0 0 0 0 0 0 0 0 ...
##  $ END_AZI   : logi [1:902297] NA NA NA NA NA NA ...
##  $ END_LOCATI: logi [1:902297] NA NA NA NA NA NA ...
##  $ LENGTH    : num [1:902297] 14 2 0.1 0 0 1.5 1.5 0 3.3 2.3 ...
##  $ WIDTH     : num [1:902297] 100 150 123 100 150 177 33 33 100 100 ...
##  $ F         : num [1:902297] 3 2 2 2 2 2 2 1 3 3 ...
##  $ MAG       : num [1:902297] 0 0 0 0 0 0 0 0 0 0 ...
##  $ FATALITIES: num [1:902297] 0 0 0 0 0 0 0 0 1 0 ...
##  $ INJURIES  : num [1:902297] 15 0 2 2 2 6 1 0 14 0 ...
##  $ PROPDMG   : num [1:902297] 25 2.5 25 2.5 2.5 2.5 2.5 2.5 25 25 ...
##  $ PROPDMGEXP: chr [1:902297] "K" "K" "K" "K" ...
##  $ CROPDMG   : num [1:902297] 0 0 0 0 0 0 0 0 0 0 ...
##  $ CROPDMGEXP: logi [1:902297] NA NA NA NA NA NA ...
##  $ WFO       : logi [1:902297] NA NA NA NA NA NA ...
##  $ STATEOFFIC: logi [1:902297] NA NA NA NA NA NA ...
##  $ ZONENAMES : logi [1:902297] NA NA NA NA NA NA ...
##  $ LATITUDE  : num [1:902297] 3040 3042 3340 3458 3412 ...
##  $ LONGITUDE : num [1:902297] 8812 8755 8742 8626 8642 ...
##  $ LATITUDE_E: num [1:902297] 3051 0 0 0 0 ...
##  $ LONGITUDE_: num [1:902297] 8806 0 0 0 0 ...
##  $ REMARKS   : logi [1:902297] NA NA NA NA NA NA ...
##  $ REFNUM    : num [1:902297] 1 2 3 4 5 6 7 8 9 10 ...
##  - attr(*, "problems")= tibble [5,255,570 x 5] (S3: tbl_df/tbl/data.frame)
##   ..$ row     : int [1:5255570] 1671 1673 1674 1675 1678 1679 1680 1681 1682 1683 ...
##   ..$ col     : chr [1:5255570] "WFO" "WFO" "WFO" "WFO" ...
##   ..$ expected: chr [1:5255570] "1/0/T/F/TRUE/FALSE" "1/0/T/F/TRUE/FALSE" "1/0/T/F/TRUE/FALSE" "1/0/T
##   ..$ actual  : chr [1:5255570] "NG" "NG" "NG" "NG" ...
##   ..$ file    : chr [1:5255570] "'storm_data.csv.bz2'" "'storm_data.csv.bz2'" "'storm_data.csv.bz2'"
##  - attr(*, "spec")=
##   .. cols(
##   ..   STATE__ = col_double(),
##   ..   BGN_DATE = col_character(),
##   ..   BGN_TIME = col_character(),
```

```
##   ..   TIME_ZONE = col_character(),
##   ..   COUNTY = col_double(),
##   ..   COUNTYNAME = col_character(),
##   ..   STATE = col_character(),
##   ..   EVTYPE = col_character(),
##   ..   BGN_RANGE = col_double(),
##   ..   BGN_AZI = col_logical(),
##   ..   BGN_LOCATI = col_logical(),
##   ..   END_DATE = col_logical(),
##   ..   END_TIME = col_logical(),
##   ..   COUNTY_END = col_double(),
##   ..   COUNTYENDN = col_logical(),
##   ..   END_RANGE = col_double(),
##   ..   END_AZI = col_logical(),
##   ..   END_LOCATI = col_logical(),
##   ..   LENGTH = col_double(),
##   ..   WIDTH = col_double(),
##   ..   F = col_double(),
##   ..   MAG = col_double(),
##   ..   FATALITIES = col_double(),
##   ..   INJURIES = col_double(),
##   ..   PROPDMG = col_double(),
##   ..   PROPDMGEXP = col_character(),
##   ..   CROPDMG = col_double(),
##   ..   CROPDMGEXP = col_logical(),
##   ..   WFO = col_logical(),
##   ..   STATEOFFIC = col_logical(),
##   ..   ZONENAMES = col_logical(),
##   ..   LATITUDE = col_double(),
##   ..   LONGITUDE = col_double(),
##   ..   LATITUDE_E = col_double(),
##   ..   LONGITUDE_ = col_double(),
##   ..   REMARKS = col_logical(),
##   ..   REFNUM = col_double()
##   .. )
```

```
names(storm)
```

```
##  [1] "STATE__"    "BGN_DATE"   "BGN_TIME"   "TIME_ZONE"  "COUNTY"
##  [6] "COUNTYNAME" "STATE"      "EVTYPE"     "BGN_RANGE"  "BGN_AZI"
## [11] "BGN_LOCATI" "END_DATE"   "END_TIME"   "COUNTY_END" "COUNTYENDN"
## [16] "END_RANGE"  "END_AZI"    "END_LOCATI" "LENGTH"     "WIDTH"
## [21] "F"          "MAG"        "FATALITIES" "INJURIES"   "PROPDMG"
## [26] "PROPDMGEXP" "CROPDMG"    "CROPDMGEXP" "WFO"        "STATEOFFIC"
## [31] "ZONENAMES"  "LATITUDE"   "LONGITUDE"  "LATITUDE_E" "LONGITUDE_"
## [36] "REMARKS"    "REFNUM"
```

Select the variables of interest to answer questions.

We only need variables that have to do with event type health(injuries, fatalities), property damage, and crop damage. Similarly, only data recorded from Jan. 1996 onward contains all event types. We will exclude earlier data from our analysis. We will also eliminate all rows without fatalities, injuries, crop or property damage.

```r
# select 8 variables on interest.

storm2 <- storm %>% select(BGN_DATE, EVTYPE, FATALITIES, INJURIES, PROPDMG, PROPDMGEXP, CROPDMG, CROPDM

# filter to exclude records entered before January 1, 1996

storm2$BGN_DATE <- as.POSIXlt(storm2$BGN_DATE, format = '%m/%d/%Y %H:%M:%S')
storm2 <- storm2 %>% filter(BGN_DATE>= as.Date('1996/01/01', format = '%Y/%m/%d'))

# include only records with values other than 0 in the 4 variables with health effects and economic cos

storm2 <- storm2 %>% filter(FATALITIES != 0 | INJURIES != 0 | PROPDMG != 0 | CROPDMG != 0)
```

Calculate property damage and crop damage from PROPDMG, PROPDMGEXP, CROPDMG, and CROPDMGEXP.

Create two new variables with the calculated property damage cost (propcost) and crop damage cost (cropcost)

```r
# Create funtion to convert data in PROPDMGEXP and CROPDMGEXP to number

mult <- function(x) {
    if(x %in% c("0", "5", "6", "4",  "2",  "3", "7", "1", "8")) {
       r <- (10^as.numeric(x))
    }
    else if(x %in% c('b', 'B')){
        r <- 1000000000
    }
    else if(x %in% c('M', 'm')){
        r <- 1000000
    }
    else if(x %in% c('K', 'k')) {
        r <- 1000
    }
    else if(x %in% c('H', 'h')){
        r <- 100
    }
    else {
       r <- 1
    }
}

# Convert property PROPDMGEXP data into numbers and multiply by PROPDMG

facprop <- sapply(storm2$PROPDMGEXP, mult)
propcost <- facprop*storm2$PROPDMG

# convert crop CROPDMGEXP data into numbers and multiply by CROPDMG

faccrop <- sapply(storm2$CROPDMGEXP, mult)
cropcost <- faccrop * storm2$CROPDMG

# Create two new columns in storm2 dataframe with calculated property_cost and crop_cost
```

```
storm2 <- storm2 %>% mutate(property_cost = propcost,
                            crop_cost = cropcost)

# Create report with totals for each impact health and economic variable

report <- storm2 %>% summarise(fatalaties = sum(FATALITIES),
                               injuries = sum(INJURIES),
                               total_property_cost = sum(property_cost),
                               total_crop_cost = sum(crop_cost))

report
```

```
## # A tibble: 1 x 4
##   fatalaties injuries total_property_cost total_crop_cost
##        <dbl>    <dbl>               <dbl>           <dbl>
## 1       8732    57975        366767615380        1201853.
```

Reclassify EVTYPES into official categories

```
# EVTYPEs from data dictionary. In the data, most of these values are in entered in capital letters.

# Load official events from data dictionary  and convert to all capitals

events <- c('Astronomical Low Tide', 'Avalanche', 'Blizzard', 'Coastal Flood', 'Cold/Wind Chill', 'Debri

events <- sapply(events, toupper)
events <- table(events)


# Convert all EVTYPEs to upper case before matching to events table.

storm2$EVTYPE <- sapply(storm2$EVTYPE, toupper)

# Reclassify events with fuzzy matching using minimum maxDist and look at summary report of
# incorrectly classified events

s <- amatch(storm2$EVTYPE, events, method = 'osa', maxDist = 1)

# r is the number of records (NA) not matched with minimum maxDist = 1

r <-which(is.na(s))

# Prepare a summary report of these unmatched records in these unmatched records?

l <- storm2$EVTYPE[r]

l <- data.frame(value = l, stringsAsFactors = T)

m <- l %>% group_by(value) %>% summarize(n = n()) %>% arrange(desc(n))
m
```

```
## # A tibble: 183 x 2
##    value                 n
##    <fct>             <int>
##  1 TSTM WIND         61778
##  2 THUNDERSTORM WIND 43097
##  3 HAIL              22679
##  4 FLASH FLOOD       19012
##  5 TORNADO           12366
##  6 LIGHTNING         11152
##  7 FLOOD              9513
##  8 HIGH WIND          5402
##  9 STRONG WIND        3369
## 10 WINTER STORM       1460
## # ... with 173 more rows
```

There are 61778 records misclassified as 'TSTM' and additional records misclassified with 'TSTM' in the variable values. We will reassign these and other misclassified records to appropriate events values.

```
# Store storm EVTYPE in new variable evtype to preserve original data.

evtype <- storm2$EVTYPE

# change 2 cases 'NON-TSTM WIND and NON TSTM WIND to 'STRONG WIND'. Change 'WIND' to 'STRONG WIND'.

a <- str_detect(evtype, 'NON-TSTM')
storm2$EVTYPE[a] <- 'STONG WIND'
a <- str_detect(evtype, 'NON TSTM')
evtype[a] <- 'STRONG WIND'
a <- str_detect(evtype, 'WIND')
evtype[a] <- 'STRONG WIND'

# Change all other values containing 'TSTM' to 'THUNDERSTORM WIND'.
# change all values containing 'MICROBURST' to 'THUNDERSTORM WIND'
a <- str_detect(evtype, 'TSTM')
evtype[a] <- 'THUNDERSTORM WIND'
a <- str_detect(evtype, 'MICROBURST')
evtype[a] <- 'THUNDERSTORM WIND'

# Change all values containing 'FLD' or 'FLOOD' To 'FLOOD'

a <- str_detect(evtype, 'FLD')
evtype[a] <- 'FLOOD'
a <-  str_detect(evtype, 'FLOOD')
evtype[a] <- 'FLOOD'

# Change all values containing 'FIRE' to 'WILDFIRE'

a <- str_detect(evtype, 'FIRE')
evtype[a] <- 'WILDFIRE'

# Change all values containing 'HURRICANE' or   'TYPHOON' to '  'HURRICANE (TYPHOON)'.

a <-  str_detect(evtype, 'HURRICANE')
evtype[a] <- 'HURRICANE (TYPHOON)'
```

```
a <-  str_detect(evtype, 'TYPHOON')
evtype[a] <- 'HURRICANE (TYPHOON)'

# Change all values with 'COLD' to 'EXTREME COLD/WIND CHILL'

a <-  str_detect(evtype, 'COLD')
evtype[a] <- 'EXTREME COLD/WIND CHILL'

# Change 'STORM SURGE' AND 'SURF' to 'STORM SURGE/TIDE',

a <-  str_detect(evtype, 'STORM SURGE')
evtype[a] <- 'STORM SURGE/TIDE'
a <-  str_detect(evtype, 'SURF')
evtype[a] <- 'STORM SURGE/TIDE'

# Change 'LIGHT SNOW' to 'WINTER WEATHER', Change 'SNOW' to 'HEAVY SNOW'
a <-  str_detect(evtype, 'LIGHT SNOW')
evtype[a] <- 'WINTER WEATHER'
a <-  str_detect(evtype, 'SNOW')
evtype[a] <- 'HEAVY SNOW'

# Change 'FOG' to 'DENSE FOG'.

a <-  str_detect(evtype, 'FOG')
evtype[a] <- 'DENSE FOG'

# Change 'WINTER WEATHER/MIX' to 'WINTER WEATHER'
a <-  str_detect(evtype, 'WINTER')
evtype[a] <- 'WINTER WEATHER'

# Change 'COASTAL STORM' to 'MARINE THUNDERSTORM WIND'
a <-  str_detect(evtype, 'COASTAL STORM')
evtype[a] <- 'MARINE THUNDERSTORM WIND'

# Change 'FREEZ' to 'FROST/FREEZE'
a <-  str_detect(evtype, 'FREEZ')
evtype[a] <- 'FROST/FREEZE'

# change 'WINT' to 'WINTER WEATHER'
a <-  str_detect(evtype, 'WINTRY MIX')
evtype[a] <- 'WINTER WEATHER'

# Change 'GLAZE' to 'ICE STORM'
a <-  str_detect(evtype, 'GLAZE')
evtype[a] <- 'ICE STORM'

# Rerun match analysis on on the cleaned data

s <- amatch(evtype, events, method = 'osa', maxDist = 1)

# r is the number of records (NA) not matched with minimum maxDist = 1

r <-which(is.na(s))
```

```
# Prepare a summary report of these unmatched records in these unmatched records?

l <- evtype[r]

l <- data.frame(value = l, stringsAsFactors = T)

m1 <- l %>% group_by(value) %>% summarize(n = n()) %>% arrange(desc(n))
m1
```

```
## # A tibble: 79 x 2
##    value               n
##    <fct>           <int>
##  1 STRONG WIND    114745
##  2 FLOOD           29519
##  3 HAIL            22679
##  4 TORNADO         12366
##  5 LIGHTNING       11152
##  6 WINTER WEATHER   2152
##  7 HEAVY SNOW       1284
##  8 WILDFIRE         1229
##  9 HEAVY RAIN       1047
## 10 EXCESSIVE HEAT    685
## # ... with 69 more rows
```

Summarize the total property damage and crop damage, injuries, fatalaties from the mismatched records to determine if we have failed to classify significant events.

```
storm3 <- storm2
storm3$EVTYPE <- evtype

mismatch <- storm3[r,]
report2 <- mismatch %>% summarize(cropcost = sum(crop_cost),
                                  propcost = sum(property_cost),
                                  fatalities = sum(FATALITIES),
                                  injuries = sum(INJURIES))
report2
```

```
## # A tibble: 1 x 4
##   cropcost     propcost fatalities injuries
##      <dbl>        <dbl>      <dbl>    <dbl>
## 1 1201853. 366767615380       8732    57975
```

Our miscellaneous mismatch group only accounts for less than 1% of our total values in all 4 categories. This should not affect our subsequent analysis.

```
report3 <- storm3 %>% group_by(EVTYPE) %>% summarize(crop_cost = sum(crop_cost),
                                                     property_cost = sum(property_cost),
                                                     fatalities = sum(FATALITIES),
                                                     injuries = sum(INJURIES)) %>%
arrange(desc(injuries))
```