Analysis of U.S. Weather Events (1996-2011)

Synopsis

The purpose of this report is to analyze data from the NOAA Storm Database to determine which events are most harmful to population health and which events have the greatest economic consequences. Population health is defined as the sum total of casualties and injuries caused by the weather event. Economic consequences are defined as the sum total of property and crop damage caused by the weather event.

From the data, I found that tornados, excessive heat, and floods have caused the most fatalities/injuries since 1996. Hurricanes, storm surges, and floods have caused the most property/crop damage since 1996.

Data Processing

The raw data used for this analysis was downloaded from the following url: Storm Data

Reading in the data I first read in the entire data set directly from the zipped csv file provided at the link above. The file contains 37 columns, eight of which we need to perform our analysis. Therefore we ignore all other columns and reset the column names to a more readable format. I read the date column in as a year value and ignore any other information. The event type is converted to all uppercase as it's read in.

Formatted Column Name

Description

year

The year the weather event occurred.

type

The type of the weather event.

fatalities

The number of fatalities caused by the weather event.

injuries

The number of injuries caused by the weather event.

propertyDamage

The cost in dollars of property damage caused by the weather event.

propertyDamageScale

The scale of the property damage caused by the weather event.

K = thousands of dollars

M = millions of dollars

B = billions of dollars

cropDamage

The cost in dollars of crop damage caused by the weather event.

cropDamageScale

The scale of the crop damage caused by the weather event.

K = thousands of dollars

M = millions of dollars

B = billions of dollars

```
# Set up date processing and convert all event types to upper case
library("lubridate")

setClass("myDate")
setClass("typeUpper")
setAs("character", "myDate", function(from) as.numeric(year(mdy_hms(from))))
setAs("character", "typeUpper", function(from) toupper(from))


# Filter out unnecessary columns
columnsToRead = c("NULL",
                  "myDate",
                  rep("NULL", 5),
                  "typeUpper",
                  rep("NULL", 14),
                  "numeric",    "numeric",  "numeric",   "character",  "numeric",    "character",
                  rep("NULL", 9))

# read data
eventData = read.csv("repdata-data-StormData.csv.bz2", header=TRUE, colClasses=columnsToRead, comment.cl
colnames(eventData) <-
 c("year", "type", "fatalities", "injuries", "propertyDamage", "propertyDamageScale", "cropDamage", "cr
```

Cleaning the data There are several things I do to clean up the raw data set. First, I filter out all weather events that happened before 1996. According to the NOAA, 1996 is the first year that information on all event types started to be collected. It also makes sense to limit the data to recent years as weather patterns change and dollar values in the data have not been adjusted for inflation.

```
eventData <- eventData[eventData$year >= 1996, ]
```

Next I eliminate events that do not have any reported fatalities/injuries and property/crop damage.

```
eventData <- eventData[eventData$fatalities != 0 | eventData$fatalities != 0 |
                       eventData$propertyDamage != 0 | eventData$cropDamage != 0, ]
```

Next I replace each pair of property and crop damage columns with a single column that has the numeric value representing each pair.

```
# return actual numeric value for damage data
# expects a two-column dataframe that contains the damage and damage scale columns
getNumericValue <- function(damageColumns)
{
    if ("" == damageColumns[2]) return(as.numeric(damageColumns[1]))
    if ("K" == damageColumns[2]) return(1000 * as.numeric(damageColumns[1]))
    if ("M" == damageColumns[2]) return(1000000 * as.numeric(damageColumns[1]))
    if ("B" == damageColumns[2]) return(1000000000 * as.numeric(damageColumns[1]))
}

propertyDamage <- apply(eventData[, c("propertyDamage", "propertyDamageScale")], 1, getNumericValue)
cropDamage <- apply(eventData[, c("cropDamage", "cropDamageScale")], 1, getNumericValue)
eventData <- cbind(eventData[, 1:4], propertyDamage, cropDamage)
```

The data now has the following format:

```
head(eventData)
```

```
##        year         type fatalities injuries propertyDamage cropDamage
## 248768 1996 WINTER STORM          0        0         380000      38000
## 248769 1996      TORNADO          0        0         100000          0
## 248770 1996    TSTM WIND          0        0           3000          0
## 248771 1996    TSTM WIND          0        0           5000          0
## 248772 1996    TSTM WIND          0        0           2000          0
## 248774 1996    HIGH WIND          0        0         400000          0
```

Cleaning Event Types The NOAA documentation specifies the following 48 event types:

```
validEventTypes <-
    as.vector(sapply(c("Astronomical Low Tide",   "Avalanche",           "Blizzard",
                       "Coastal Flood",           "Cold/Wind Chill",     "Debris Flow",
                       "Dense Fog",               "Dense Smoke",         "Drought",
                       "Dust Devil",              "Dust Storm",          "Excessive Heat",
                       "Extreme Cold/Wind Chill", "Flash Flood",         "Flood",
                       "Frost/Freeze",            "Funnel Cloud",        "Freezing Fog",
                       "Hail",                    "Heat",                "Heavy Rain",
                       "Heavy Snow",              "High Surf",           "High Wind",
                       "Hurricane (Typhoon)",     "Ice Storm",           "Lake-Effect Snow",
                       "Lakeshore Flood",         "Lightning",           "Marine Hail",
                       "Marine High Wind",        "Marine Strong Wind",  "Marine Thunderstorm Wind",
                       "Rip Current",             "Seiche",              "Sleet",
                       "Storm Surge/Tide",        "Strong Wind",         "Thunderstorm Wind",
                       "Tornado",                 "Tropical Depression", "Tropical Storm",
                       "Tsunami",                 "Volcanic Ash",        "Waterspout",
                       "Wildfire",                "Winter Storm",        "Winter Weather"), toupper))
```

After displaying both the percentage of rows that contain invalid event values and the actual invalid values, it is clear the event data needs cleaning.

```
invalidEventTypes <- eventData[!(eventData$type %in% validEventTypes), ]
print(paste("The percentage of events that have invalid event types = ",
            round(nrow(invalidEventTypes) / nrow(eventData) * 100, 2)))
```

```
## [1] "The percentage of events that have invalid event types =  32.87"
```

```
print(unique(invalidEventTypes$type))
```

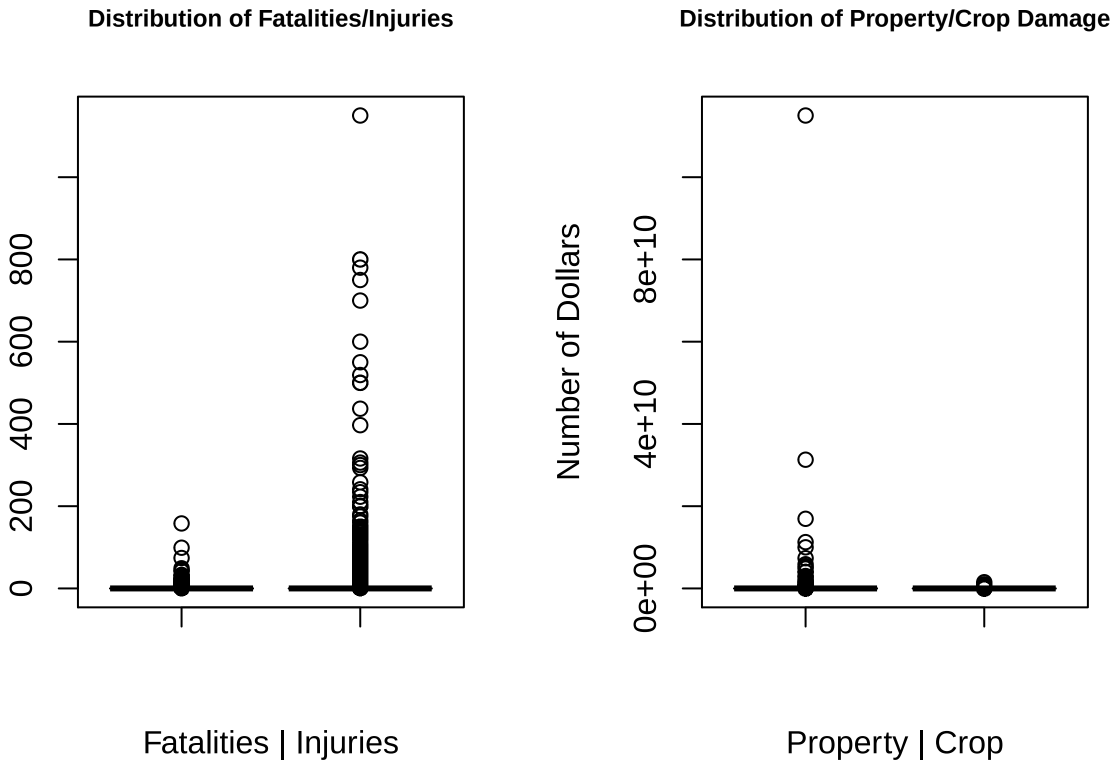
```
##   [1] "TSTM WIND"                 "FREEZING RAIN"
##   [3] "EXTREME COLD"              "TSTM WIND/HAIL"
##   [5] "RIP CURRENTS"              "OTHER"
##   [7] "WILD/FOREST FIRE"          "STORM SURGE"
##   [9] "ICE JAM FLOOD (MINOR"      "URBAN/SML STREAM FLD"
##  [11] "ROUGH SURF"                "HEAVY SURF"
##  [13] "MARINE ACCIDENT"           "FOG"
##  [15] "FREEZE"                    "DRY MICROBURST"
##  [17] "WINDS"                     "COASTAL STORM"
##  [19] "EROSION/CSTL FLOOD"        "RIVER FLOODING"
##  [21] "DAMAGING FREEZE"           "HURRICANE"
##  [23] "BEACH EROSION"             "HEAVY RAIN/HIGH SURF"
##  [25] "UNSEASONABLE COLD"         "EARLY FROST"
##  [27] "WINTRY MIX"                "COASTAL FLOODING"
##  [29] "LANDSLUMP"                 "TIDAL FLOODING"
##  [31] "STRONG WINDS"              "EXTREME WINDCHILL"
##  [33] "GLAZE"                     "EXTENDED COLD"
##  [35] "WHIRLWIND"                 "HEAVY SNOW SHOWER"
##  [37] "LIGHT SNOW"                "MIXED PRECIP"
##  [39] "COLD"                      "FREEZING SPRAY"
##  [41] "DOWNBURST"                 "MUDSLIDES"
##  [43] "MICROBURST"                "SNOW"
##  [45] "SNOW SQUALLS"              "WIND DAMAGE"
##  [47] "LIGHT SNOWFALL"            "FREEZING DRIZZLE"
##  [49] "GUSTY WIND/RAIN"           "GUSTY WIND/HVY RAIN"
##  [51] "WIND"                      "COLD TEMPERATURE"
##  [53] "COLD AND SNOW"             "RAIN/SNOW"
##  [55] "TSTM WIND (G45)"           "GUSTY WINDS"
##  [57] "GUSTY WIND"                "TSTM WIND 40"
##  [59] "TSTM WIND 45"              "HARD FREEZE"
##  [61] "TSTM WIND (41)"            "RIVER FLOOD"
##  [63] "TSTM WIND (G40)"           "MUD SLIDE"
##  [65] "SNOW AND ICE"              "AGRICULTURAL FREEZE"
##  [67] "SNOW SQUALL"               "ICY ROADS"
##  [69] "HYPOTHERMIA/EXPOSURE"      "LAKE EFFECT SNOW"
##  [71] "MIXED PRECIPITATION"       "BLACK ICE"
##  [73] "COASTALSTORM"              "DAM BREAK"
##  [75] "BLOWING SNOW"              "FROST"
##  [77] "GRADIENT WIND"             "UNSEASONABLY COLD"
##  [79] "TSTM WIND AND LIGHTNING"   "WET MICROBURST"
##  [81] "MUDSLIDE"                  "HEAVY SURF AND WIND"
##  [83] "TYPHOON"                   "LANDSLIDES"
##  [85] "HIGH SWELLS"               "HIGH WINDS"
##  [87] "SMALL HAIL"                "UNSEASONAL RAIN"
##  [89] "COASTAL FLOODING/EROSION"  " TSTM WIND (G45)"
##  [91] "TSTM WIND  (G45)"          "HIGH WIND (G40)"
##  [93] "TSTM WIND (G35)"           "COASTAL EROSION"
##  [95] "UNSEASONABLY WARM"         "COASTAL  FLOODING/EROSION"
##  [97] "HYPERTHERMIA/EXPOSURE"     "ROCK SLIDE"
##  [99] "GUSTY WIND/HAIL"           "HEAVY SEAS"
## [101] " TSTM WIND"                "LANDSPOUT"
## [103] "RECORD HEAT"               "EXCESSIVE SNOW"
## [105] "FLOOD/FLASH/FLOOD"         "WIND AND WAVE"
## [107] "FLASH FLOOD/FLOOD"         "LIGHT FREEZING RAIN"
## [109] "ICE ROADS"                 "HIGH SEAS"
## [111] "RAIN"                        "ROUGH SEAS"
## [113] "TSTM WIND G45"               "NON-SEVERE WIND DAMAGE"
## [115] "THUNDERSTORM WIND (G40)"     "LANDSLIDE"
## [117] "HIGH WATER"                  " FLASH FLOOD"
## [119] "LATE SEASON SNOW"            "THUNDERSTORM"
## [121] "FALLING SNOW/ICE"            "NON-TSTM WIND"
## [123] "BLOWING DUST"                "   HIGH SURF ADVISORY"
## [125] "WINTER WEATHER MIX"          "COLD WEATHER"
## [127] "ICE ON ROAD"                 "DROWNING"
## [129] "MARINE TSTM WIND"            "HURRICANE/TYPHOON"
## [131] "WINTER WEATHER/MIX"          "ASTRONOMICAL HIGH TIDE"
## [133] "HEAVY SURF/HIGH SURF"
```

Classifying invalid event types is difficult and cannot be done perfectly. For example, does “GUSTY WIND/HAIL” get classified under a wind event type or hail type? I reviewed the invalid event types and classified them as best I could with the rules listed below. After making the substitutions, I then removed any events classified as “OTHER” (both those originally classified as “OTHER” and ones I classified as “OTHER”).

```
convertToValidEventType <- function(eventType)
{
    if (grepl("MARINE TSTM WIND", eventType)) return("MARINE THUNDERSTORM WIND")
    if (grepl("TSTM WIND", eventType)) return("THUNDERSTORM WIND")
    if (grepl("THUNDERSTORM", eventType)) return("THUNDERSTORM WIND")
    if (grepl("GUSTY WIND", eventType)) return("HIGH WIND")
    if (!(grepl("^EXTREME", eventType)) & grepl("COLD", eventType)) return("COLD/WIND CHILL")
    if (grepl("^EXTREME", eventType)) return("EXTREME COLD/WIND CHILL")
    if (grepl("EXPOSURE", eventType)) return("EXTREME COLD/WIND CHILL")
    if (grepl("^FOG", eventType)) return("DENSE FOG")
    if (grepl("ROAD", eventType)) return("WINTER WEATHER")
    if (grepl("WEATHER", eventType)) return("WINTER WEATHER")
    if (grepl("BLACK ICE", eventType)) return("WINTER WEATHER")
    if (grepl("LAKE EFFECT SNOW", eventType)) return("LAKE-EFFECT SNOW")
    if (!(grepl("^LAKE", eventType)) & grepl("SNOW", eventType)) return("HEAVY SNOW")
    if (grepl("SURF", eventType)) return("HIGH SURF")
    if (grepl("FREEZING SPRAY", eventType)) return("HIGH SURF")
    if (grepl("FREEZING RAIN", eventType)) return("SLEET")
    if (grepl("MIX", eventType)) return("SLEET")
    if (grepl("FREEZ", eventType) & !(grepl("FOG", eventType))) return("FROST/FREEZE")
    if (grepl("FROST", eventType)) return("FROST/FREEZE")
    if (grepl("ASTRONOMICAL", eventType)) return("ASTRONOMICAL LOW TIDE")
    if (grepl("COAST", eventType)) return("COASTAL FLOOD")
    if (grepl("EROSION", eventType)) return("COASTAL FLOOD")
    if (grepl("FLASH", eventType)) return("FLASH FLOOD")
    if (grepl("HURRICANE", eventType)) return("HURRICANE (TYPHOON)")
    if (grepl("TYPHOON", eventType)) return("HURRICANE (TYPHOON)")
    if (grepl("CURRENTS", eventType)) return("RIP CURRENT")
    if (grepl("FIRE", eventType)) return("WILDFIRE")
    if (grepl("TIDAL", eventType)) return("STORM SURGE/TIDE")
    if (grepl("SURGE", eventType)) return("STORM SURGE/TIDE")
    if (grepl("BURST", eventType)) return("HEAVY RAIN")
    if (grepl("RAIN", eventType)) return("HEAVY RAIN")
    if (grepl(" SEAS", eventType)) return("MARINE HIGH WIND")
    if (grepl("SWELLS", eventType)) return("MARINE HIGH WIND")
    if (grepl("WIND AND WAVE", eventType)) return("MARINE HIGH WIND")
    if (grepl("RIVER", eventType)) return("FLOOD")
    if (grepl("DAM", eventType)) return("FLOOD")
    if (grepl("HIGH WATER", eventType)) return("FLOOD")
    if (grepl("HEAT", eventType)) return("EXCESSIVE HEAT")
    if (grepl("BLOWING DUST", eventType)) return("DUST STORM")
    if (grepl("LANDSPOUT", eventType)) return("DUST DEVIL")
    if (grepl("^WIND", eventType)) return("HIGH WIND")
    if (grepl("WINDS", eventType)) return("HIGH WIND")
    if (grepl("GRADIENT WIND", eventType)) return("HIGH WIND")
    if (grepl("G40", eventType)) return("HIGH WIND")
    if (grepl("WHIRLWIND", eventType)) return("TORNADO")
    if (grepl("SMALL HAIL", eventType)) return("HAIL")
    if (grepl("ICE JAM", eventType)) return("OTHER")
    if (grepl("URBAN", eventType)) return("OTHER")
    if (grepl("SLIDE", eventType)) return("OTHER")
    if (grepl("SLUMP", eventType)) return("OTHER")
    if (grepl("GLAZE", eventType)) return("OTHER")
    if (grepl("DROWN", eventType)) return("OTHER")
    if (grepl("UNSEASONABLY", eventType)) return("OTHER")
    if (grepl("ACCIDENT", eventType)) return("OTHER")

    return(eventType)
}

eventData$type <- sapply(eventData$type, convertToValidEventType)
eventData <- eventData[eventData$type != "OTHER", ]
```

Outliers Finally, I checked to see if there were any outliers in the data. I generated a panel plot that shows box plots for each vector of health/damage data:

```
par(mfrow = c(1, 2))
boxplot(eventData$fatalities, eventData$injuries,
        main="Distribution of Fatalities/Injuries",
        xlab="Fatalities | Injuries",
        cex.main=0.75)

boxplot(eventData$propertyDamage, eventData$cropDamage,
        main="Distribution of Property/Crop Damage",
        xlab="Property | Crop",
        ylab="Number of Dollars",
        cex.main=0.75)
```



It appears there is an outlier in the property damage for a single event. It appears likely that the data set records the damage in billions of dollars rather than millions of dollars. I determined the event for this outlier and converted the property damage to millions of dollars.

```
maxIndex = which.max(eventData$propertyDamage)
print(eventData[maxIndex,])
```

```
##        year  type fatalities injuries propertyDamage cropDamage
## 605953 2006 FLOOD          0        0       1.15e+11   32500000
```

```
eventData$propertyDamage[maxIndex] <- eventData$propertyDamage[maxIndex] / 1000
```

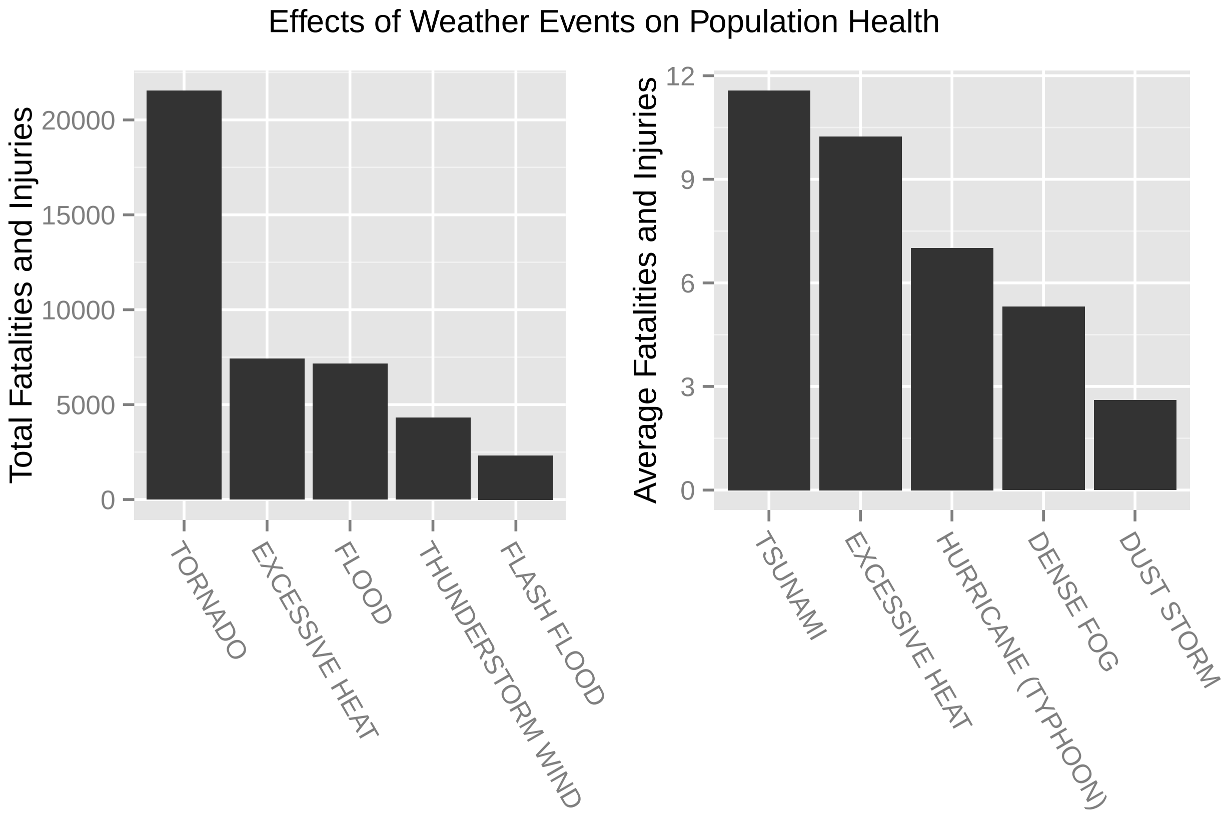
Results

First I plotted data to show the effects of weather events on population health. I created a panel plot that shows both the top 5 event types in terms of total fatalities/injuries and the top 5 event types in terms of average fatalities/injuries.

```
library(ggplot2)
library(gridExtra)
```

```
## Loading required package: grid
```

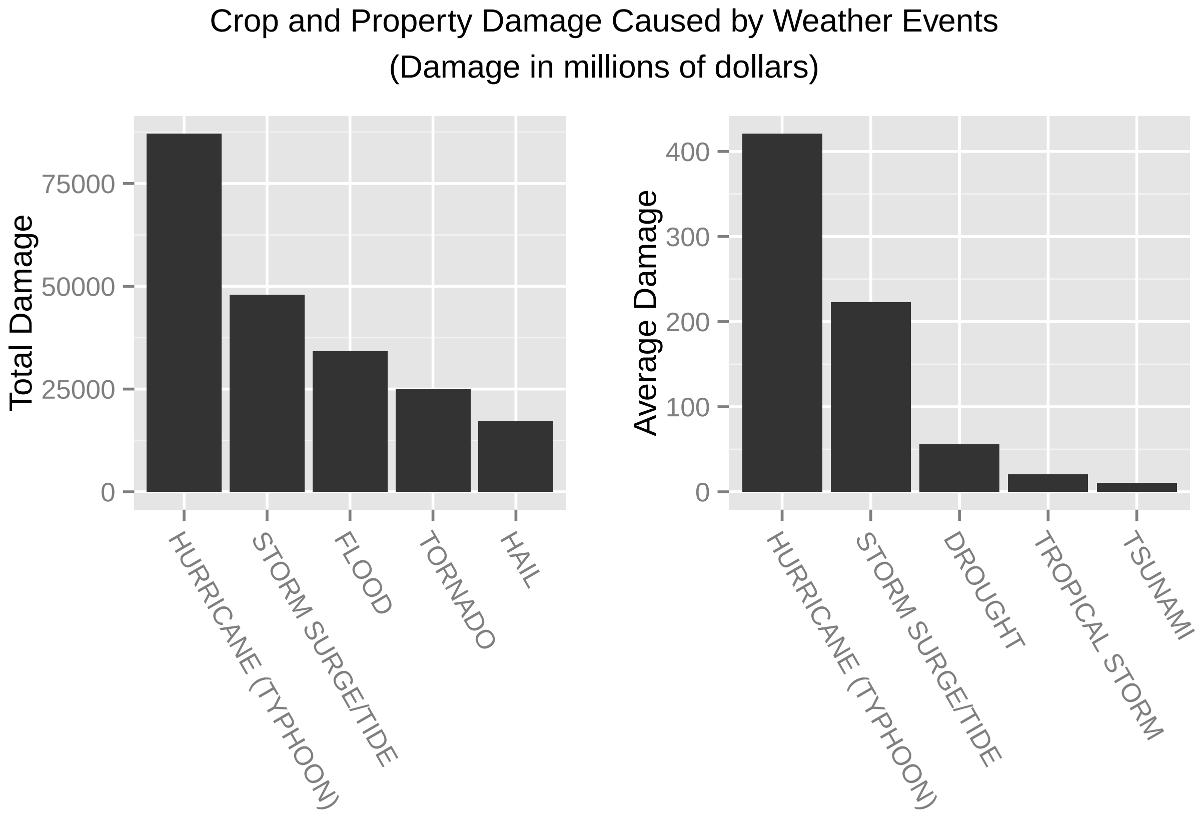
```
eventData$type <- factor(eventData$type)
healthDataSum <- setNames(aggregate(eventData$fatalities + eventData$injuries,
                                    by=eventData[c("type")],
                                    FUN=sum),
                          c("type", "amount"))
healthDataSum <- healthDataSum[order(-healthDataSum$amount),][1:5,]

healthDataAvg <- setNames(aggregate(eventData$fatalities + eventData$injuries,
                                    by=eventData[c("type")],
                                    FUN=mean),
                          c("type", "amount"))
healthDataAvg <- healthDataAvg[order(-healthDataAvg$amount),][1:5,]

p1 <- ggplot(healthDataSum,
             aes(reorder(healthDataSum$type, order(healthDataSum$amount, decreasing=TRUE)),
                 healthDataSum$amount)) +
      xlab("") +
      ylab("Total Fatalities and Injuries") +
      geom_bar(stat="identity") +
      theme(axis.text.x = element_text(angle = -60, hjust = 0))

p2 <- ggplot(healthDataAvg,
            aes(reorder(healthDataAvg$type, order(healthDataAvg$amount, decreasing=TRUE)),
                healthDataAvg$amount)) +
      xlab("") +
      ylab("Average Fatalities and Injuries") +
      geom_bar(stat="identity") +
      theme(axis.text.x = element_text(angle = -60, hjust = 0))

grid.arrange(p1, p2, ncol=2,
             main="Effects of Weather Events on Population Health")
```



The data shows that tornados stand out as the weather event that causes the most fatalities and injuries. Excessive heat also shows up in both plots, which should make it a focus for areas of the country where excessive heat can be an issue.

Next, I plot the data to show the property and crop damage caused by weather events. Again, I create a panel plot that shows the top 5 events in terms of total damage and average damage.

```r
library(ggplot2)
library(gridExtra)

damageDataSum <- setNames(aggregate(eventData$propertyDamage / 1000000 + eventData$cropDamage / 1000000
                                 by=eventData[c("type")],
                                 FUN=sum),
                          c("type", "amount"))
damageDataSum <- damageDataSum[order(-damageDataSum$amount),][1:5,]

damageDataAvg <- setNames(aggregate(eventData$propertyDamage / 1000000 + eventData$cropDamage / 1000000
                                 by=eventData[c("type")],
                                 FUN=mean),
                          c("type", "amount"))
damageDataAvg <- damageDataAvg[order(-damageDataAvg$amount),][1:5,]

p1 <- ggplot(damageDataSum,
             aes(reorder(damageDataSum$type, order(damageDataSum$amount, decreasing=TRUE)),
                 damageDataSum$amount)) +
      xlab("") +
      ylab("Total Damage") +
      geom_bar(stat="identity") +
      theme(axis.text.x = element_text(angle = -60, hjust = 0))

p2 <- ggplot(damageDataAvg,
             aes(reorder(damageDataAvg$type, order(damageDataAvg$amount, decreasing=TRUE)),
                 damageDataAvg$amount)) +
      xlab("") +
      ylab("Average Damage") +
      geom_bar(stat="identity") +
      theme(axis.text.x = element_text(angle = -60, hjust = 0))

grid.arrange(p1, p2, ncol=2,
             main=paste("Crop and Property Damage Caused by Weather Events",
                        "(Damage in millions of dollars)",
                        sep="\n"))
```



The data shows that both hurricanes and storm surges stand out as causing the most total and average damage of all of the weather events. It is also interesting to note that floods are in the top three of both health/damage plots in terms of overall effect.

Further Analysis The NOAA database also provides information at the state level. Further research could be done to report on data at the state level and make regional recommendations. Plots could also show yearly trends or plots categorized by season.

Environment The code/output for this report was generated in the following environment:

```
sessionInfo()
```

```
## R version 3.0.3 (2014-03-06)
## Platform: x86_64-apple-darwin10.8.0 (64-bit)
##
## locale:
## [1] en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8
##
## attached base packages:
## [1] grid      stats     graphics  grDevices utils     datasets  methods
## [8] base
##
## other attached packages:
## [1] gridExtra_0.9.1 ggplot2_1.0.0   lubridate_1.3.3
##
## loaded via a namespace (and not attached):
```

```
##  [1] codetools_0.2-8  colorspace_1.2-4 digest_0.6.4     evaluate_0.5.5  
##  [5] formatR_0.10     gtable_0.1.2     htmltools_0.2.4  knitr_1.6       
##  [9] labeling_0.2     MASS_7.3-33      memoise_0.2.1    munsell_0.4.2   
## [13] plyr_1.8.1       proto_0.3-10     Rcpp_0.11.1      reshape2_1.4    
## [17] rmarkdown_0.2.46 scales_0.2.4     stringr_0.6.2    tools_3.0.3     
## [21] yaml_2.1.11
```