Analysis on impact of severe weather events

Maria A. Flores

21st June 2015

Synopsis

Storms and other severe weather events can cause both public health and economic problems for communities and municipalities. Many severe events can result in fatalities, injuries, and property damage, and preventing such outcomes to the extent possible is a key concern.

This analysis uses the U.S. National Oceanic and Atmospheric Administration's (NOAA) storm database, which tracks characteristics of major storms and weather events in the United States and its harmful effects, if any. The data was cleaned and arranged to sumarize the number of fatalities, injuries and economic damages on properties or crops attributable to a weather event. The tackled questions are:

- Across the United States, which types of events are most harmful with respect to population health?
- Across the United States, which types of events have the greatest economic consequences?

Data processing

Environment settings

The first step was to set the locale into English and load some required libraries. In order to check the installed libraries and load or install them if it's not, a new function *loadLibrary()* was created. The working directory was stablished too, but I found it was pointless to add the code for my own directory. If you are going to execute this in your computer, just remember to set your own working directory.

```
Sys.setlocale("LC_ALL","en_GB.UTF-8");

loadLibrary <- function(x)
{
    if (!require(x,character.only = TRUE))
    {
        install.packages(x,dep=TRUE)
        if(!require(x,character.only = TRUE)) stop("Package not found")
    }
}

loadLibrary("stringr");
loadLibrary("tm");
loadLibrary("stringdist");
loadLibrary("dplyr");
```

Download and import the data into R

Check if the file is downloaded and load the data into *stormData*.

```
if (!file.exists("repdata_data_StormData.csv.bz2")){
    fileUrl <- "https://d396qusza40orc.cloudfront.net/repdata%2Fdata%2FStormData.csv.bz2";
    download.file(fileUrl, destfile="repdata_data_StormData.csv.bz2", method="curl");
}

stormData <- read.csv("repdata_data_StormData.csv.bz2");
```

Data cleaning

The events in the database start in the year 1950 and end in November 2011. In the earlier years of the database there are generally fewer events recorded, most likely due to a lack of good records. More recent years should be considered more complete. The quality of the records is variable, and there are some columns needs to be cleared, normalized or reinterpreted before a good analysis could be done.

State

Taking a look at the data evidenced that there are weather events that were reported in states that don't match with any of the 50 USA states codes plus "DC". There are 72 different state codes in the data, so we have some codes that can't be matched in a particular state.

```
states <- as.data.frame(unique(stormData$STATE));
names(states) <- c("originalStates");
statesInUSA <- as.vector(state.abb);
statesInUSA <- as.data.frame(c(statesInUSA, "DC"));
statesInUSA$filler <- 1;
names(statesInUSA) <- c("stateCode","present");
statesChecklist <- merge(states, statesInUSA, by.x="originalStates", by.y="stateCode", all = TRUE);
statesChecklist[is.na(statesChecklist$present),];
```

```
##    originalStates present
## 3              AM      NA
## 4              AN      NA
## 6              AS      NA
## 15             GM      NA
## 16             GU      NA
## 25             LC      NA
## 26             LE      NA
## 27             LH      NA
## 28             LM      NA
## 29             LO      NA
## 30             LS      NA
## 34             MH      NA
## 52             PH      NA
## 53             PK      NA
## 54             PM      NA
## 55             PR      NA
## 56             PZ      NA
```

```
## 60              SL      NA
## 61              ST      NA
## 66              VI      NA
## 72              XX      NA
```

Even though this remark, we know the whole point of the NOAA storm database is to track events that were relevant to the USA, so the state codes were not cleaned, considering all of the events as valid ones.

Event type

Following the NOAA document Storm Data preparation, it could be seen that there are only 48 event types that can be reported in the database (see table 2.1.1). This notably differs from reality, as there are 985 different event types along the database.

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

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

Taking a look at the data, it's possible to appreciate that abbreviations are used (i.e. "TSTM" for "THUNDERSTORM"), the liberal use of plurals that makes different event types for the same facts (i.e. "WIND" and "WINDS"), etc. This highly difficults the analysis, so there is a need to clean and normalize this field. I faced this by the use of an approximate solution that tries to avoid the tedious work of look over the 985 events and try to figure out what is their correspondences in the 48-event-type list.

The first step was to import the 48 event types. This could be done by a variety of ways. I copied the 48 types from the document and created a list of normalized events. Then, a "normalization list" is created. It will work as a translation table, to look up for the normalized event type that corresponds to a certain event type in the database. At this point, we only add to the list the 48 list.

```
normalizedEventTypes <- matrix(toupper(str_trim(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"))), ncol = 1);

colnames(normalizedEventTypes) <- "normalizedEvents";

normalizationList <- matrix(cbind(normalizedEventTypes,
                                  normalizedEventTypes),
                            ncol = 2);
colnames(normalizationList) <- c("originalEvent", "normalizedEvent");
head(normalizationList);
```

```
##      originalEvent           normalizedEvent
## [1,] "ASTRONOMICAL LOW TIDE" "ASTRONOMICAL LOW TIDE"
## [2,] "AVALANCHE"             "AVALANCHE"
## [3,] "BLIZZARD"              "BLIZZARD"
## [4,] "COASTAL FLOOD"         "COASTAL FLOOD"
## [5,] "COLD/WIND CHILL"       "COLD/WIND CHILL"
## [6,] "DEBRIS FLOW"           "DEBRIS FLOW"
```

The list is improved and extended in further parts of this analysis. Instead of trying to find out the correct event type for each one of the 985 events, the list is improved as much as is needed according to the specific analysis.

Magnitudes in the damages amount

In the previously mentioned document *Storm Data preparation* (see section 2.7), the amounts in dollars are followed by an alphabetical character that express the magnitude of the number. There are only three magnitudes mentioned, "K" for thousands, "M" for millions and "B" for billions. But the data in this field is assorted:

```
unique(stormData$PROPDMGEXP);
```

```
##  [1] K M   B m + 0 5 6 ? 4 2 3 h 7 H - 1 8
## Levels:  - ? + 0 1 2 3 4 5 6 7 8 B h H K m M
```

```
unique(stormData$CROPDMGEXP);
```

```
## [1]   M K m B ? 0 k 2
## Levels:  ? 0 2 B k K m M
```

I assume that the same magnitude can be expressed in lowercase or uppercase, so i.e. *m* and *M* both will express "milions". For the other magnitudes, however, it's difficult to know what these means. Let's quantify the number of rows that have non-standard magnitudes versus the total number of rows in the database.

```
dim(stormData);
```

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

```
length(stormData[stormData$PROPDMGEXP %in% (
    c("-", "?", "+", "0", "1", "2", "3", "4", "5", "6", "7", "8", "h", "H")), "REFNUM"]);
```

```
## [1] 321
```

```
length(stormData[stormData$CROPDMGEXP %in% (
    c("?", "0", "2")), "REFNUM"]);
```

```
## [1] 27
```

It seems that only a few rows have incorrect values for the magnitudes, but this rows could have magnitudes of billions and this would distort the results. Before ignoring them, a search for these events was made over the cleaned online database at Storm Events Database. The procedure was to find the event in the search form and compare the amounts taked in account versus the amounts declared. The result was that is possible to extract some conclusions about it:

- H and h means for "hundred"
- Where the magnitude is *?* the amount is 0
- The other magnitudes seems like some error with the data import structure. Sometimes these events can't be found in the online database, so maybe they were ignored. It applies to almost all the amounts with magnitude "+", except in one case, where the amount was interpreted as units.
- For numeric magnitudes, when the event is found, the amount imputed corresponds to the concatenation of the amount field with the value in the magnitude (i.e. PROPDMG = 10.00, PROPDMGEXP = 4, real value was 104) and interpreted as expressed in units.
- For magnitudes not specified (blank/null value, 76 rows) the amounts are small and there is no any clue about magnitude, so these values are ignored.

So, I use this findings to build a table where the multipliers could be found.

```
magnitudesConversion <- as.data.frame(
                        matrix(cbind(c("H", "h",
                                       "K", "k",
                                       "M", "m",
                                       "B", "b",
                                       "-", "+", "?",
                                       "0", "1", "2", "3", "4",
                                       "5", "6", "7", "8", "9"),
                                     as.numeric(c(100, 100,
                                       1000, 1000,
                                       1000000,1000000,
                                       1000000000, 1000000000,
                                       1, 1, 1,
                                       1, 1, 1, 1, 1,
                                       1, 1, 1, 1, 1))),
                               ncol = 2),
                        stringsAsFactors = FALSE);
colnames(magnitudesConversion) <- c("magnitude", "multiplier");
head(magnitudesConversion);
```

```
##   magnitude multiplier
## 1         H        100
## 2         h        100
## 3         K       1000
## 4         k       1000
## 5         M      1e+06
## 6         m      1e+06
```

Data analysis

Definitions

The main purpose of this project is to know what are the most harmful weather events, so this is the criteria for the definition of "harmful" that this analysis uses:

- **Harmful degree for population health:** the number of people affected, counting both injuries and fatalities.
- **Harmful degree for economic consequences:** the amount of dollars in damages, counting both property damages and crop damages.

Harmful events for population health

By mean of the *aggregate()* function, the fatalities and injuries values are aggregated by event type ("EVTYPE" column). Then, both tables are merged in *sumHarmByType*, then the new column "totalAffected" is created, which will give the value for the the degree that a certain type of event is harmful for the people.

```
sumFatalitiesByType <- aggregate(stormData$FATALITIES, by=list(toupper(str_trim(stormData$EVTYPE))), FU
names(sumFatalitiesByType) <- c("eventtype","fatalities");
sumInjuriesByType <- aggregate(stormData$INJURIES, by=list(toupper(str_trim(stormData$EVTYPE))), FUN=su
names(sumInjuriesByType) <- c("eventtype","injuries");
sumHarmByType <- merge(sumFatalitiesByType,sumInjuriesByType, all = TRUE);
sumHarmByType$totalAffected <- sumHarmByType$fatalities + sumHarmByType$injuries;
```

Once we have both values aggregated by the type of event in the database, it's time to use the normalized events table. The purpose is to match as many rows as possible, within the ones with values of injuries or fatalities above 0. The first try uses only the basic table, matching the events in the database that have exact correspondence with the normalized events. A measure of the amount of matches achieved is shown as a percentage.

```
harmfulForHealth <- sumHarmByType[sumHarmByType$totalAffected > 0,];
firstTry <- merge(harmfulForHealth, normalizedEventTypes, by.x = "eventtype", by.y = "normalizedEvents"
sprintf("%1.2f%%",sum(firstTry$totalAffected)/sum(harmfulForHealth$totalAffected)*100);
```

```
## [1] "89.77%"
```

Although 89.77% of direct matching is a good value, I try to increase this value a little more. In order to develop the strategy of matching event types with injuries or fatalities, the function *stringdist()* is used. This function measures the distance between two strings and is used to compare each event type occurrence in the database with the normalized ones. When two strings are considered close enough to say that they are really the same type of event (i.e. "HIGH WIND" and "HIGH WINDS"), the event and its corresponding normalized event is added to the normalization list. The value for the distance (in this case is 2) was selected running the function with different values, starting from 1, then 2 and then 3. With a value of 3 the procedure introduces errors, marking like if it was the same type of event two strings that are not really representing the same event type. So 2 is the best possible value for this procedure.

```
normalizedEventTypes <- as.data.frame(normalizedEventTypes, stringsAsFactors = FALSE);

dim(normalizationList);
```

```
## [1] 48  2
```

```
for(i in harmfulForHealth$eventtype){
    for(j in normalizedEventTypes$normalizedEvents){
        distance <- stringdist(i,j);
        if(distance > 0 & distance <= 2){
            normalizationList <- rbind(normalizationList, c(i,j));
        }
    }
}

dim(normalizationList);
```

```
## [1] 62  2
```

```
normalizationList;
```

```
##       originalEvent              normalizedEvent
##  [1,] "ASTRONOMICAL LOW TIDE"    "ASTRONOMICAL LOW TIDE"
##  [2,] "AVALANCHE"                "AVALANCHE"
##  [3,] "BLIZZARD"                 "BLIZZARD"
##  [4,] "COASTAL FLOOD"            "COASTAL FLOOD"
##  [5,] "COLD/WIND CHILL"          "COLD/WIND CHILL"
##  [6,] "DEBRIS FLOW"              "DEBRIS FLOW"
##  [7,] "DENSE FOG"                "DENSE FOG"
##  [8,] "DENSE SMOKE"              "DENSE SMOKE"
##  [9,] "DROUGHT"                  "DROUGHT"
## [10,] "DUST DEVIL"               "DUST DEVIL"
## [11,] "DUST STORM"               "DUST STORM"
## [12,] "EXCESSIVE HEAT"           "EXCESSIVE HEAT"
## [13,] "EXTREME COLD/WIND CHILL"  "EXTREME COLD/WIND CHILL"
## [14,] "FLASH FLOOD"              "FLASH FLOOD"
## [15,] "FLOOD"                    "FLOOD"
## [16,] "FROST/FREEZE"             "FROST/FREEZE"
## [17,] "FUNNEL CLOUD"             "FUNNEL CLOUD"
## [18,] "FREEZING FOG"             "FREEZING FOG"
## [19,] "HAIL"                     "HAIL"
## [20,] "HEAT"                     "HEAT"
## [21,] "HEAVY RAIN"               "HEAVY RAIN"
## [22,] "HEAVY SNOW"               "HEAVY SNOW"
## [23,] "HIGH SURF"                "HIGH SURF"
## [24,] "HIGH WIND"                "HIGH WIND"
## [25,] "HURRICANE (TYPHOON)"      "HURRICANE (TYPHOON)"
## [26,] "ICE STORM"                "ICE STORM"
## [27,] "LAKE-EFFECT SNOW"         "LAKE-EFFECT SNOW"
## [28,] "LAKESHORE FLOOD"          "LAKESHORE FLOOD"
## [29,] "LIGHTNING"                "LIGHTNING"
## [30,] "MARINE HAIL"              "MARINE HAIL"
## [31,] "MARINE HIGH WIND"         "MARINE HIGH WIND"
## [32,] "MARINE STRONG WIND"       "MARINE STRONG WIND"
## [33,] "MARINE THUNDERSTORM WIND" "MARINE THUNDERSTORM WIND"
## [34,] "RIP CURRENT"              "RIP CURRENT"
## [35,] "SEICHE"                   "SEICHE"
## [36,] "SLEET"                    "SLEET"
## [37,] "STORM SURGE/TIDE"         "STORM SURGE/TIDE"
## [38,] "STRONG WIND"              "STRONG WIND"
## [39,] "THUNDERSTORM WIND"        "THUNDERSTORM WIND"
## [40,] "TORNADO"                  "TORNADO"
## [41,] "TROPICAL DEPRESSION"      "TROPICAL DEPRESSION"
## [42,] "TROPICAL STORM"           "TROPICAL STORM"
## [43,] "TSUNAMI"                  "TSUNAMI"
## [44,] "VOLCANIC ASH"             "VOLCANIC ASH"
## [45,] "WATERSPOUT"               "WATERSPOUT"
## [46,] "WILDFIRE"                 "WILDFIRE"
## [47,] "WINTER STORM"             "WINTER STORM"
## [48,] "WINTER WEATHER"           "WINTER WEATHER"
## [49,] "AVALANCE"                 "AVALANCHE"
```

```
## [50,] "FLASH FLOODS"             "FLASH FLOOD"
## [51,] "HEAVY RAINS"              "HEAVY RAIN"
## [52,] "HIGH WINDS"               "HIGH WIND"
## [53,] "LIGHTNING."               "LIGHTNING"
## [54,] "RIP CURRENTS"             "RIP CURRENT"
## [55,] "STRONG WINDS"             "STRONG WIND"
## [56,] "THUNDERSTORM  WINDS"      "THUNDERSTORM WIND"
## [57,] "THUNDERSTORM WINDS"       "THUNDERSTORM WIND"
## [58,] "THUNDERSTORM WINDSS"      "THUNDERSTORM WIND"
## [59,] "THUNDERSTORMS WINDS"      "THUNDERSTORM WIND"
## [60,] "THUNDERTORM WINDS"        "THUNDERSTORM WIND"
## [61,] "WILD FIRES"               "WILDFIRE"
## [62,] "WINTER STORMS"            "WINTER STORM"
```

Starting from the 48 direct matches, the normalization list is increased with some more values until 62 matches. Let's see how this had increased the percentage of values covered.

```
secondTry <- merge(harmfulForHealth,
                   normalizationList,
                   by.x = "eventtype",
                   by.y = "originalEvent");

sprintf("%1.2f%%",(sum(secondTry$totalAffected)/sum(harmfulForHealth$totalAffected))*100);
```

```
## [1] "91.09%"
```

The coverage is improved until 91.09%. This still could be improved. In order to have a guidance of what are the better strategy to improve, I create a check list to see the values matched, not matched and their values for the total amount of people affected (*totalAffected*). The better strategy will be the one that finds a match on the event types with higher *totalAffected* values.

```
checklistSecondTry <- merge(harmfulForHealth,
                            normalizationList,
                            by.x = "eventtype",
                            by.y = "originalEvent",
                            all = TRUE);
checklistSecondTry <- arrange(checklistSecondTry, desc(totalAffected));

head(arrange(as.data.frame(cbind(checklistSecondTry$eventtype,
                as.character(checklistSecondTry$normalizedEvent),
                as.character(checklistSecondTry$totalAffected)),
          ncol = 3),
        desc(checklistSecondTry$totalAffected)));
```

```
##                V1              V2    V3
## 1         TORNADO         TORNADO 96979
## 2 EXCESSIVE HEAT EXCESSIVE HEAT  8428
## 3       TSTM WIND            <NA>  7461
## 4           FLOOD           FLOOD  7259
## 5       LIGHTNING       LIGHTNING  6046
## 6            HEAT            HEAT  3037
```

Taking a look at the results, the strategy from here it's to insert in the normalization list the correct matches for the event types with higher values of people affected. For missing values until 100 affected people, the substitution is applied. In this matching some guesses are made, i.e., is assumed that "GLAZE" corresponds to the normalized event type "FROST/FREEZE". Moreover, if there are two events consigned, the preference is for the first mentioned. Of course, some mistakes could be done with this suppositions.

```
normalizationList <- rbind(normalizationList,
    matrix(cbind(checklistSecondTry[grep("^TSTM WIND",
                                         checklistSecondTry$eventtype),
                                    1],"THUNDERSTORM WIND"), ncol = 2));

normalizationList <- rbind(normalizationList,
                           matrix(cbind(checklistSecondTry[grep("^THUNDERSTORM WIND",
                                                                checklistSecondTry$eventtype),
                                                           1],"THUNDERSTORM WIND"), ncol = 2));

normalizationList <- rbind(normalizationList, c("HURRICANE/TYPHOON","HURRICANE (TYPHOON)"));
normalizationList <- rbind(normalizationList, c("FOG","DENSE FOG"));
normalizationList <- rbind(normalizationList, c("WILD/FOREST FIRE","WILDFIRE"));
normalizationList <- rbind(normalizationList, c("HEAT WAVE","EXCESSIVE HEAT"));
normalizationList <- rbind(normalizationList, c("EXTREME COLD","COLD/WIND CHILL"));
normalizationList <- rbind(normalizationList, c("EXTREME HEAT","EXCESSIVE HEAT"));
normalizationList <- rbind(normalizationList, c("GLAZE","FROST/FREEZE"));
normalizationList <- rbind(normalizationList, c("ICE","FROST/FREEZE"));
normalizationList <- rbind(normalizationList, c("WIND","HIGH WIND"));
normalizationList <- rbind(normalizationList, c("HURRICANE","HURRICANE (TYPHOON)"));
normalizationList <- rbind(normalizationList, c("TSTM WIND/HAIL","THUNDERSTORM WIND"));
normalizationList <- rbind(normalizationList, c("WINTER WEATHER/MIX","WINTER WEATHER"));
normalizationList <- rbind(normalizationList, c("URBAN/SML STREAM FLD","FLOOD"));
normalizationList <- rbind(normalizationList, c("COLD","COLD/WIND CHILL"));
normalizationList <- rbind(normalizationList, c("HEAVY SURF/HIGH SURF","HIGH SURF"));
normalizationList <- rbind(normalizationList, c("FLASH FLOODING","FLASH FLOOD"));

normalizationList <- unique.matrix(normalizationList);
```

Let's see the improvement made by these matches.

```
checklistSecondTry <- merge(harmfulForHealth,
                            normalizationList,
                            by.x = "eventtype",
                            by.y = "originalEvent",
                            all.x = TRUE);
checklistSecondTry <- arrange(checklistSecondTry, desc(totalAffected));

secondTry <- merge(harmfulForHealth,
                   normalizationList,
                   by.x = "eventtype",
                   by.y = "originalEvent");

sprintf("%1.2f%%",
        (sum(secondTry$totalAffected) / sum(harmfulForHealth$totalAffected))*100);
```

```
## [1] "99.08%"
```

At this point, the 99.08% of the values are covered. As the amount of time and assumptions that requires increases a lot and the improvement margin is so little, it's a good moment to stop the matches. The values not matched are marked as "OTHERS". Now, the data is prepared again, aggregating it by the new normalized event type.

```
levels(checklistSecondTry$normalizedEvent) <- c(levels(checklistSecondTry$normalizedEvent),"OTHERS");
checklistSecondTry$normalizedEvent[is.na(checklistSecondTry$normalizedEvent)] <- "OTHERS";

mostHarmfulHealth <- aggregate(list(checklistSecondTry$fatalities,
                                    checklistSecondTry$injuries,
                                    checklistSecondTry$totalAffected),
                               by=list(checklistSecondTry$normalizedEvent),
                               FUN=sum);
names(mostHarmfulHealth) <- c("eventtype","fatalities","injuries", "peopleAffected");
mostHarmfulHealth <- arrange(mostHarmfulHealth, peopleAffected);
```

With this data, we can select the types of events most harmful for the population health. It's possible to build a list of the most harmful events. A percentage of coverage is created in order to see what percentage of affected people is due to each event type.

```
mostHarmfulHealth$perF <- sprintf("%1.2f%%",(mostHarmfulHealth$fatalities / sum(mostHarmfulHealth$fatali
mostHarmfulHealth$perI <- sprintf("%1.2f%%",(mostHarmfulHealth$injuries / sum(mostHarmfulHealth$injuries
mostHarmfulHealth$perT <- sprintf("%1.2f%%",(mostHarmfulHealth$peopleAffected /
                                             sum(mostHarmfulHealth$peopleAffected)*100));
```

Greatest economic consequences

In this case, the first step is to apply the multipliers in order to achieve the real values in units ($). As was mentioned above, the values with numercial magnitude were really an error, so I correct this restoring the original value informed, multiplying by 10 and adding the last digit. The other magnitudes have the correct values, so it's possible to obtain the amount in units using the multipliers.

Beyond this point, the procedure is analog to the one followed for the previous analysis on population health. By mean of the *aggregate()* function, the property and crop damages are aggregated by event type ("EVTYPE" column). Then, both tables are merged in *sumDamagesByType*, then the new column "totalDamages" is created, which will give the value for the degree that a certain type of event is harmful for the properties or crops.

```
propertyDamages <- stormData[stormData$PROPDMGEXP != "" & stormData$PROPDMG > 0,
                      c("EVTYPE","PROPDMG","PROPDMGEXP")];
propertyDamages$PROPDMG[propertyDamages$PROPDMGEXP %in% c(0,1,2,3,4,5,6,7,8,9)] <-
    (propertyDamages$PROPDMG[propertyDamages$PROPDMGEXP %in% c(0,1,2,3,4,5,6,7,8,9)] * 10) +
    as.numeric(as.character(propertyDamages$PROPDMGEXP[propertyDamages$PROPDMGEXP %in% c(0,1,2,3,4,5,6,

propertyDamages <- merge(propertyDamages, magnitudesConversion,
                 by.x = "PROPDMGEXP", by.y = "magnitude",
                 all.x = TRUE);
propertyDamages$realPropertyDamage <- propertyDamages$PROPDMG * as.numeric(propertyDamages$multiplier);

sumPropertyDamages <- aggregate(propertyDamages$realPropertyDamage,
                        by=list(toupper(str_trim(propertyDamages$EVTYPE))), FUN=sum);
names(sumPropertyDamages) <- c("eventtype","propertyDamages");
```

```
cropDamages <- stormData[stormData$CROPDMGEXP != "" & stormData$CROPDMG > 0,
                     c("EVTYPE","CROPDMG","CROPDMGEXP")];
cropDamages$CROPDMG[cropDamages$CROPDMGEXP %in% c(0,1,2,3,4,5,6,7,8,9)] <-
    (cropDamages$CROPDMG[cropDamages$CROPDMGEXP %in% c(0,1,2,3,4,5,6,7,8,9)] * 10) +
    as.numeric(as.character(cropDamages$CROPDMGEXP[cropDamages$CROPDMGEXP %in% c(0,1,2,3,4,5,6,7,8,9)]))

cropDamages <- merge(cropDamages, magnitudesConversion,
                 by.x = "CROPDMGEXP", by.y = "magnitude",
                 all.x = TRUE);
cropDamages$realCropDamage <- cropDamages$CROPDMG * as.numeric(cropDamages$multiplier);

sumCropDamages <- aggregate(cropDamages$realCropDamage, by=list(toupper(str_trim(cropDamages$EVTYPE))),
names(sumCropDamages) <- c("eventtype","cropDamages");

sumDamagesByType <- merge(sumPropertyDamages, sumCropDamages,
                          by.x = "eventtype", by.y = "eventtype",
                          all = TRUE);
sumDamagesByType$propertyDamages[is.na(sumDamagesByType$propertyDamages)] <- 0;
sumDamagesByType$cropDamages[is.na(sumDamagesByType$cropDamages)] <- 0;
sumDamagesByType$totalDamages<- sumDamagesByType$propertyDamages + sumDamagesByType$cropDamages;
sum(sumDamagesByType$totalDamages);
```

```
## [1] 476422914708
```

Once we have both values aggregated by the type of event in the database, it's time to use the normalized events table. The purpose is to match as many rows as possible, within the ones with values of damages above 0. The first try uses only the basic table, matching the events in the database that have exact correspondence with the normalized events. A measure of the amount of matches achieved is shown as a percentage.

```
damageFirstTry <- merge(sumDamagesByType,
                        normalizationList,
                        by.x = "eventtype",
                        by.y = "originalEvent");
sprintf("%1.2f%%",
        (sum(damageFirstTry$totalDamages) / sum(sumDamagesByType$totalDamages))*100);
```

```
## [1] "85.97%"
```

Although 85.97% of direct matching is a good value, I try to increase this value a little more following the same strategy as in the previous section but using the values that we matched there.

```
for(i in sumDamagesByType$eventtype){
    for(j in normalizedEventTypes$normalizedEvents){
        distance <- stringdist(i,j);
        if(distance > 0 & distance <= 2){
            normalizationList <- rbind(normalizationList, c(i,j));
        }
    }
}
```

The check list for the damages data is created. Taking a look at the data, we can see some possible improvements. Values with "/" in the event type are trying to be matched with the normalized events. This

is done because of the values with multiple events. We use the first of the events mentioned as the tentative matching value.

```
damagesCheckList <- merge(sumDamagesByType,
                          normalizationList,
                          by.x = "eventtype",
                          by.y = "originalEvent",
                          all.x = TRUE);
damagesCheckList <- arrange(damagesCheckList, desc(totalDamages));

for(i in sumDamagesByType$eventtype){
    for(j in normalizedEventTypes$normalizedEvents){
        unlist(strsplit(i,"/"))[1]
        distance <- stringdist(unlist(strsplit(i,"/"))[1],j);
        if(distance > 0 & distance <= 2){
            normalizationList <- rbind(normalizationList, c(i,j));
        }
    }
}

normalizationList <- unique.matrix(normalizationList);
```

I use regular expressions and replacements to try to match as many values as possible.

```
normalizationList <- rbind(normalizationList,
                           matrix(cbind
                                  (damagesCheckList[
                                      grep("^HURRICANE",
                                           damagesCheckList$eventtype),
                                      1],
                                   "HURRICANE (TYPHOON)"),
                                  ncol = 2));

normalizationList <- rbind(normalizationList,
                           matrix(cbind
                                  (damagesCheckList[
                                      grep("(^FLOOD)",
                                           damagesCheckList$eventtype),
                                      1],
                                   "FLOOD"),
                                  ncol = 2));

normalizationList <- rbind(normalizationList,
                           matrix(cbind
                                  (damagesCheckList[
                                      grep("FREEZE",
                                           damagesCheckList$eventtype),
                                      1],
                                   "FROST/FREEZE"),
                                  ncol = 2));

normalizationList <- rbind(normalizationList,
                           matrix(cbind(damagesCheckList[grep("^THUNDERSTORM WIND",
                                                              damagesCheckList$eventtype),
```

```
                                                    1],"THUNDERSTORM WIND"), ncol = 2));

normalizationList <- rbind(normalizationList,
                           matrix(cbind(damagesCheckList[grep("^THUDERSTORM WIND",
                                                              damagesCheckList$eventtype),
                                                    1],"THUNDERSTORM WIND"), ncol = 2));

normalizationList <- rbind(normalizationList,
                           matrix(cbind(damagesCheckList[grep("^THUNDEERSTORM WIND",
                                                              damagesCheckList$eventtype),
                                                    1],"THUNDERSTORM WIND"), ncol = 2));

normalizationList <- rbind(normalizationList,
                           matrix(cbind(damagesCheckList[grep("^FLASH FLOOD",
                                                              damagesCheckList$eventtype),
                                                    1],"FLASH FLOOD"), ncol = 2));

normalizationList <- rbind(normalizationList,
                           matrix(cbind(damagesCheckList[grep("^HAIL",
                                                              damagesCheckList$eventtype),
                                                  1],"HAIL"), ncol = 2));

normalizationList <- rbind(normalizationList, c("STORM SURGE","STORM SURGE/TIDE"));
normalizationList <- rbind(normalizationList, c("RIVER FLOOD","FLOOD"));
normalizationList <- rbind(normalizationList, c("HURRICANE OPAL","HURRICANE (TYPHOON)"));
normalizationList <- rbind(normalizationList, c("HEAVY RAIN/SEVERE WEATHER","HEAVY RAIN"));
normalizationList <- rbind(normalizationList, c("TORNADOES, TSTM WIND, HAIL","TORNADO"));
normalizationList <- rbind(normalizationList, c("HEAVY RAIN/SEVERE WEATHER","HEAVY RAIN"));
normalizationList <- rbind(normalizationList, c("SEVERE THUNDERSTORM","TORNADO"));
normalizationList <- rbind(normalizationList, c("TYPHOON","HURRICANE (TYPHOON)"));
normalizationList <- rbind(normalizationList, c("LANDSLIDE","LANDSLIDE"));
normalizationList <- rbind(normalizationList, c("HAILSTORM","HAIL"));
normalizationList <- rbind(normalizationList, c("RIVER FLOODING","FLOOD"));
normalizationList <- rbind(normalizationList, c("EXCESSIVE WETNESS","EXCESSIVE WETNESS"));
normalizationList <- rbind(normalizationList, c("COASTAL FLOODING","FLOOD"));
normalizationList <- rbind(normalizationList, c("MAJOR FLOOD","FLOOD"));
normalizationList <- rbind(normalizationList, c("FROST","FROST/FREEZE"));
normalizationList <- rbind(normalizationList, c("COLD AND WET CONDITIONS","EXTREME COLD/WIND CHILL"));
normalizationList <- rbind(normalizationList, c("WINTER STORM HIGH WINDS","WINTER STORM"));
normalizationList <- rbind(normalizationList, c("RECORD COLD","EXTREME COLD/WIND CHILL"));
normalizationList <- rbind(normalizationList, c("WATERSPOUT/TORNADO","WATERSPOUT"));
normalizationList <- rbind(normalizationList, c("EARLY FROST","FROST/FREEZE"));
normalizationList <- rbind(normalizationList, c("SEVERE THUNDERSTORM WINDS","THUNDERSTORM WIND"));

normalizationList <- unique.matrix(normalizationList);
```

Let's see how this had increased the percentage of values covered.

```
damagesCheckList <- merge(sumDamagesByType,
                          normalizationList,
                          by.x = "eventtype",
                          by.y = "originalEvent",
                          all.x = TRUE);
```

```
damagesCheckList <- arrange(damagesCheckList, desc(totalDamages));

sprintf("%1.2f%%",
        (sum(damagesCheckList$totalDamages[!is.na(damagesCheckList$normalizedEvent)])
        / sum(sumDamagesByType$totalDamages))*100);
```

```
## [1] "99.92%"
```

At this point, the 99.92% of the values are covered. As the amount of time and assumptions that requires increases a lot and the improvement margin is so little, it's a good moment to stop the matches. The values not matched are marked as "OTHERS". Now, the data is prepared again, aggregating it by the new normalized event type.

```
levels(damagesCheckList$normalizedEvent) <- c(levels(damagesCheckList$normalizedEvent),"OTHERS");
damagesCheckList$normalizedEvent[is.na(damagesCheckList$normalizedEvent)] <- "OTHERS";

mostDamaging <- aggregate(list(damagesCheckList$propertyDamages,
                               damagesCheckList$cropDamages,
                               damagesCheckList$totalDamages),
                               by=list(damagesCheckList$normalizedEvent),
                               FUN=sum);
names(mostDamaging) <- c("eventtype","propertyDamages","cropDamages", "totalDamages");
mostDamaging <- arrange(mostDamaging, totalDamages);
```

With this data, I can select the types of events most damaging for the properties and crops. It's possible to build a list of the most damaging events. A percentage of coverage is created in order to see what percentage of damage is due to each event type.

```
mostDamaging$perP <- sprintf("%1.2f%%",(mostDamaging$propertyDamages / sum(mostDamaging$propertyDamages)
mostDamaging$perC <- sprintf("%1.2f%%",(mostDamaging$cropDamages / sum(mostDamaging$cropDamages))*100);
mostDamaging$perT <- sprintf("%1.2f%%",(mostDamaging$totalDamages / sum(mostDamaging$totalDamages))*100)
```

Results

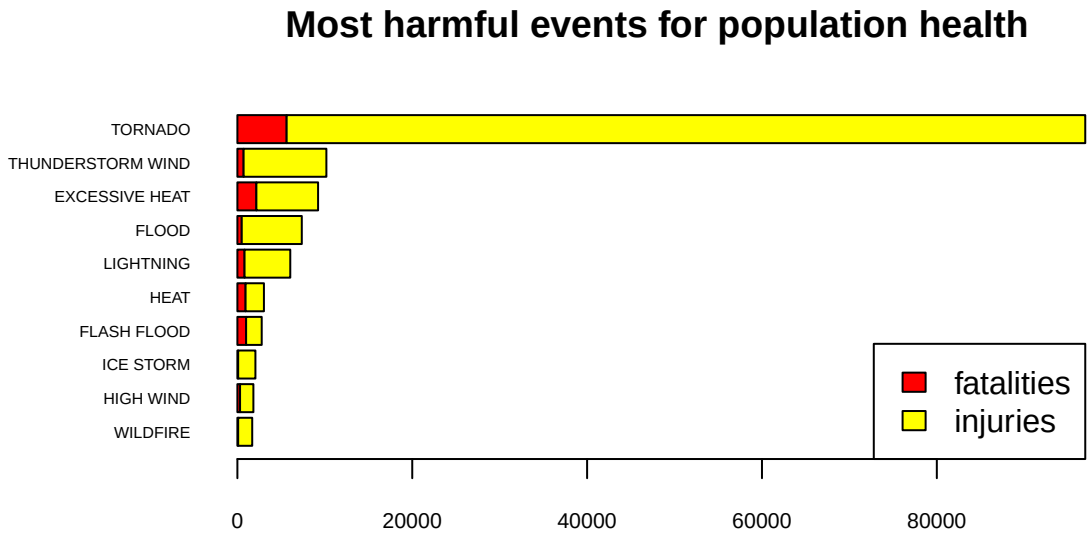
The top 10 of **harmful for the population** event types are concentrating around the 90% of all victims, both injuries and fatalities. I consider this fact a good sign for selecting this 10 events as the most harmful for people health.

```
top10 <- top_n(as.tbl(mostHarmfulHealth), 10, wt = peopleAffected);
top10 <- arrange(top10, perT);
options(dplyr.width = Inf);

arrange(top10, desc(peopleAffected));
```

```
## Source: local data frame [10 x 7]
##
##            eventtype fatalities injuries peopleAffected   perF   perI
## 1            TORNADO       5633    91346          96979 37.19% 65.00%
## 2  THUNDERSTORM WIND        710     9469          10179  4.69%  6.74%
## 3     EXCESSIVE HEAT       2171     7059           9230 14.33%  5.02%
```

```
## 4              FLOOD        498     6868           7366  3.29%  4.89%
## 5          LIGHTNING        817     5230           6047  5.39%  3.72%
## 6               HEAT        937     2100           3037  6.19%  1.49%
## 7        FLASH FLOOD        999     1785           2784  6.60%  1.27%
## 8          ICE STORM         89     1975           2064  0.59%  1.41%
## 9          HIGH WIND        306     1525           1831  2.02%  1.09%
## 10          WILDFIRE         90     1606           1696  0.59%  1.14%
##      perT
## 1  62.30%
## 2   6.54%
## 3   5.93%
## 4   4.73%
## 5   3.88%
## 6   1.95%
## 7   1.79%
## 8   1.33%
## 9   1.18%
## 10  1.09%
```

```r
mostHarmfulHealthPlot <- matrix(rbind(top10$fatalities,
                                      top10$injuries),
                                nrow = 2);
rownames(mostHarmfulHealthPlot) <- c("fatalities", "injuries");
colnames(mostHarmfulHealthPlot) <- top10$eventtype;

par(oma=c(2,2,2,2));
barplot(mostHarmfulHealthPlot,
        horiz = TRUE,
        beside = FALSE,
        names.arg = colnames(mostHarmfulHealthPlot),
        legend.text = rownames(mostHarmfulHealthPlot),
        args.legend = list(x = "bottomright"),
        main="Most harmful events for population health",
        col = heat.colors(2),
        cex.names = 0.5,
        cex.axis = 0.7,
        las = 1);
```



As can be seen, the most harmful events are, by large, the tornadoes.

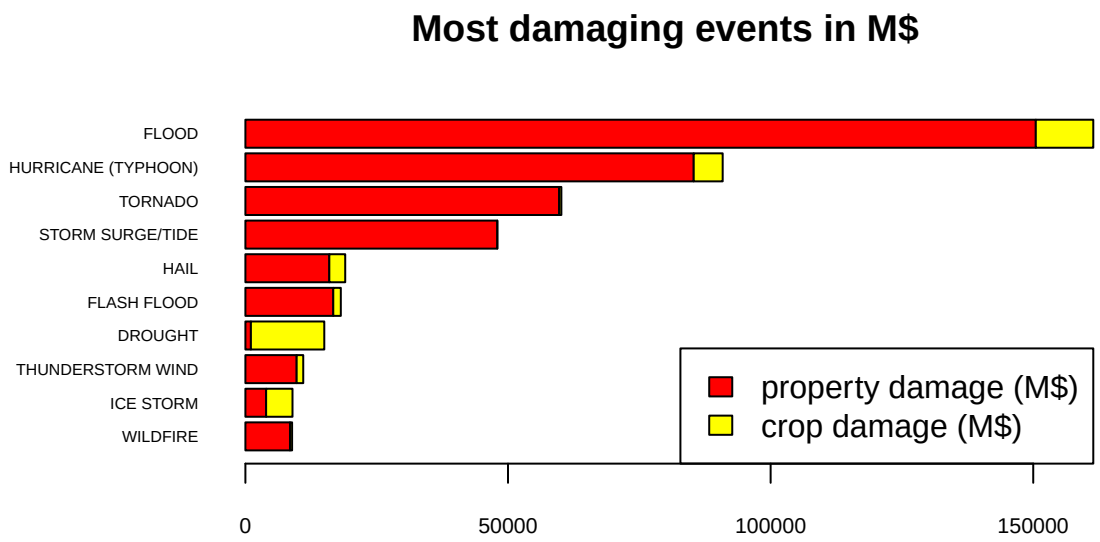
In the case of the **economic impact**, the top 10 of damaging events for the properties/crops are concentrating around the 85-90% of all damages. I consider this fact a good sign for being selected this 10 events as the most damaging for the properties or crops.

```
top10E <- top_n(as.tbl(mostDamaging), 10, wt = totalDamages);
top10E <- arrange(top10E, totalDamages);
options(dplyr.width = Inf);

arrange(top10E, desc(totalDamages));
```

```
## Source: local data frame [10 x 7]
##
##              eventtype propertyDamages cropDamages totalDamages   perP
## 1                FLOOD    150489038622 10944736050 161433774672 35.22%
## 2  HURRICANE (TYPHOON)     85356410010  5516117800  90872527810 19.97%
## 3              TORNADO     59742523412   417655710  60160179122 13.98%
## 4     STORM SURGE/TIDE     47964724000      855000  47965579000 11.22%
## 5                 HAIL     15979772432  3026844800  19006617232  3.74%
## 6          FLASH FLOOD     16732872132  1437163150  18170035282  3.92%
## 7              DROUGHT      1046106000 13972566000  15018672000  0.24%
## 8    THUNDERSTORM WIND      9761727331  1253409700  11015137031  2.28%
## 9            ICE STORM      3944928310  5022113500   8967041810  0.92%
## 10            WILDFIRE      8491543500   402769630   8894313130  1.99%
##      perC   perT
## 1  22.29% 33.88%
## 2  11.23% 19.07%
## 3   0.85% 12.63%
## 4   0.00% 10.07%
## 5   6.16%  3.99%
## 6   2.93%  3.81%
## 7  28.45%  3.15%
## 8   2.55%  2.31%
## 9  10.23%  1.88%
## 10  0.82%  1.87%
```

```
mostDamagingPlot <- matrix(rbind(top10E$propertyDamages / 1000000,
                                 top10E$cropDamages / 1000000),
                           nrow = 2);
rownames(mostDamagingPlot) <- c("property damage (M$)", "crop damage (M$)");
colnames(mostDamagingPlot) <- top10E$eventtype;

par(oma=c(2,2,2,2));
barplot(mostDamagingPlot,
        horiz = TRUE,
        beside = FALSE,
        names.arg = colnames(mostDamagingPlot),
        legend.text = rownames(mostDamagingPlot),
        args.legend = list(x = "bottomright"),
        main="Most damaging events in M$",
        col = heat.colors(2),
        cex.names = 0.5,
```

```
        cex.axis = 0.7,
        las = 1);
```



As can be seen, the most damaging events are the floods, followed by the hurricanes, the tornadoes and the storm surges/tides.