Natural catastrope in United states and their implications on Human life and economy

Shashikant Singh Kunwar

1. Synopsis

Every year different weather events cost a country both in terms of human lives as well as capital. In this work we look at the U.S National Oceanic and Atmospheric Administration (NOAA) storm database that is collected starting 1950 until November 2011. The data has information about different instances of the weather event and the damages due to each one of them. In the first part of this work we find out which weather event leads to largest number of fatalities and injuries. Our results are presented as a bar plot displaying number of total fatality and total injury for each event. In the second part using the data of damages to properties and damages to crop we calculate the total damage due to each event. One can see some of these events can cause staggering amount in billions. The results for this is presented as a bar plot showing the cost of damages (in dollars) for each of the event. Over the years tornadoes have the highest fatality as well as injuries. On the other hand, damages due to flood cost heavily to economy over the years.

2. Data Download

We start by first downloading and saving the data. The data can be downloaded from the given URL or downloaded locally first and then loaded into the code. However, this goes against our philosophy of minimizing human interference. So the code first downloads the data (only if it's not present locally) once the data is downloaded then it is loaded into a data frame. If the data is already loaded it will not be downloaded again and we use cache=TRUE so the operations are only performed if the function is changes. We can check the names of various column in the dataset using the `names` command.

```
library(plyr)

# url of the data
url<-"https://d396qusza40orc.cloudfront.net/repdata%2Fdata%2FStormData.csv.bz2"

# download file if it doesn't exists
if (!file.exists("stormdata.csv")){
download.file(url=url,destfile='./stormdata.csv',method="curl")}

# load file into dataframe
data<-read.csv('stormdata.csv')

# column names of the data
names(data)
```

```
##  [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"
```

```
# dimensionality of the data
dimd <-dim(data)
dimd
```

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

Once the data is loaded we can check the size of data using the dim command. In this case the dimensionality of data is 902297, 37 and it'a quite big data set. We can get more information about the data using str function. This tells us the datatype of data in each column as well as it shows first few values of each column.

```
# str to get some more information about the data
str(data)
```

```
## '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 ...
```

We want to answer two question using this data set and they can be summarized as follows.

1. First which weather event costed greatly to human health over the years ?
2. Second which weather event cost more to economy in terms of damages to properties and crops.

To answer these two question we will separate our analysis in two different sections. One section will present steps required to get the damage by each event for human health and the other section will shows results of the financial damage done by each one of these weather event. First we will start with damage to human health.

3. Human cost of weather events.

These weather events can lead to deaths and injuries. In this part we will focus on finding out how much fatalities and injuries each of these weather event leads to. First we can just find out what is the total number of fatalities and injuries due to these weather events and they are given by Total fatalities and Total injuries.

```
df<-data[,c("EVTYPE","FATALITIES","INJURIES")]

# total fatality and injuries due to extreme weather conditions
rbind("Total fatalities:"=sum(df$FATALITIES),"Total injuries:"=sum(df$INJURIES))
```

```
##                      [,1]
## Total fatalities:  15145
## Total injuries:   140528
```

3.1. Data Processing

Now we want to answer the question how much each weather event contribute to these fatalities and injuries. First we know there are 902297 datasets. Thus, we have this many values of weather events, so just before jumping into calculating total number for each event and plotting the result a smart choice will be to see how many unique values of weather events are actually in our data set.

```
# unique values of weather events
evtype<-unique(df$EVTYPE)

# total number of events
uniqueevent<-length(evtype)
uniqueevent
```

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

If one prints the total number of unique values of the event type it's close to 985 but if one looks carefully we can see repetitions due to typos, case sensitivity, abbreviations etc. Thus we need to do some more work to filter the data.

```
# first converting column's value to lower case and removing
# trailing and leading whitespaces
df$EVTYPE<-trimws(tolower(df$EVTYPE))

# getting unique value of weather events
evtype<-unique(df$EVTYPE)
filteredval<-length(evtype)

# total number of events
filteredval
```

```
## [1] 890
```

So we convert all event names into lowercase, remove white space and then find the unique values. After this first careful analysis the number of unique values are 890. Less than before but still difficult to work with.

We want to see how many type each event is reported a good way to do it just get the frequency count of each event once all the names are converted into lower case.

```
countev<-count(df,"EVTYPE")
head(countev[order(countev$freq,decreasing = TRUE),],10)
```

```
##                 EVTYPE   freq
## 204               hail 288661
## 771          tstm wind 219946
## 677  thunderstorm wind  82564
## 750            tornado  60652
## 130        flash flood  54278
## 146              flood  25327
## 703 thunderstorm winds  20843
## 312          high wind  20214
## 410          lightning  15755
## 266         heavy snow  15708
```

We only print top 10 events based on the frequency of occurrence. So our analysis shows hail has highest occurrance. We can also see something unusual there are two different version of thunderstorm winds in only 10 rows of the data set and this is not particular to only thunderstorm but other event as well. So we need to do a much more involved filtering analysis to get the data correctly. The analysis should put all these events into same category this will ensure we get a correct number and not underestimate of the actual number.

Before going further we can imagine a situation where an event leads to no injury and fatality so rather than working with all the data we can only working with subset of data that is non-zero. This can significantly reduce the size of data that we need to work with and can help further.

```
## converting data type of columns of the data sets to number.
df$FATALITIES<-as.numeric(df$FATALITIES)
df$INJURIES<-as.numeric(df$INJURIES)

# grouping data according to event name and summing event with same name
aggresult<-aggregate(df[,c("FATALITIES","INJURIES")],
                     by=list(EVENT=df$EVTYPE),FUN=sum)

# storing only data that has non zero value for either FATALITIES or INJURIES
aggresult<-aggresult[(aggresult$FATALITIES !=0 | aggresult$INJURIES != 0),]

# renaming column of the aggregated dataset.
names(aggresult)<-c("EVENT","TOTALFATALITY","TOTALINJURY")

# check to see if we didn't miss any entry that has non-zero value.
print(c(sum(aggresult$TOTALFATALITY),sum(aggresult$TOTALINJURY)))
```

```
## [1]  15145 140528
```

```
# printing first 10 entries in the aggregated dataset
head(aggresult,10)
```

```
##                     EVENT TOTALFATALITY TOTALINJURY
## 10                avalance             1           0
## 11               avalanche           224         170
## 19               black ice             1          24
## 20                blizzard           101         805
## 32            blowing snow             2          14
## 37              brush fire             0           2
```

```
## 41            coastal flood            3           2
## 42         coastal flooding            3           0
## 43 coastal flooding/erosion            0           5
## 44            coastal storm            3           2
```

First we converts FATALITIES and INJURIES columns into numeric data type then perform a grouping of all the similar event and get only the data that is non-zero. To see if during subsetting we didn't loose any information we can again calculate the TOTALFATALITY and TOTALINJURY and they match with what we had before and this significantly reduces the dimensionality of data we have to work with. We also print top 10 result of the reduced data set. Now we can start working with how to filter the event type and store the unique values.

```
library(stringdist)
library(hash)

#  list of the events
evlist<-aggresult$EVENT

# returns all the strings starting with same alphabet
samealphabt<-function(x,nameev){
  nalpha<-nameev[grepl(paste("^",x,sep=""),nameev)]
  nalpha
}


# copying the list and removing spaces,special characters from the strings
# this helps in the processing
evlist2<-evlist
temp<-gsub(" ","",gsub("/","",evlist2))
temp<-gsub("\\(|\\)","",temp)
temp<-gsub("-","",temp)
```

We first remove punctuation from our list of the event (evlist). After that we calculate the distance between each event alphabetically. We only consider two string to be similar if the distance between them is less than or equal to 3. However, this may lead to error if we compare strings with small number of alphabets. So we will also check manually if we get the result that is expected. The similar results are stored in a hash (dictionary in python) container that has key value pair. So the key will match to the entry in the original event list and replace it with the value.

```
# container to store key-value pair, only strings that have distance of less than
# 3 are assumed close and if there are wrong key value pair they are removed.
simval<-hash()
for(let in letters){
  sal<-samealphabt(let,temp)
  for(i in 1:length(sal)){
     wrd<-list()
     for(j in i:length(sal)){
          if(i!=j){
              dst<-stringdist(sal[i],sal[j])
              if(isTRUE(dst <= 3)){simval[sal[j]]<-sal[i]}
                  }
                        }
}
 }
```

If we get some entries wrong we will delete the key value pair from the hash. This process is repeated until

we have a smaller subset of group left that we can categorize manually. This significantly reduced the list of weather event.

```
# list of keys of wrong key-value pair in the list
wrong_key<-c('fog','frost','heat','heavysnow','heavysurf','high','highwinds'
             ,'thunderstorm','winterstorm')

# deleting wrong matches
del(wrong_key,simval)


# replacing strings in the list with the similar string based on distance
for (j in temp){
     if (j %in% keys(simval)){
    temp[which(j==temp)]<-simval[[j]]}
}



# list of common occurring words that can be filtered using grep not distance
listgrp<-c('coastalflood',"drought","drymicroburst","flashflood","freezing",
           "heavysnow","heavysurf","hurricane","hypothermia","icestorm",
           "ripcurrent","snow","thunderstorm","tornado","tropicalstorm",
           "tstm","winterstorm","winterweather")

# container to store key-value pair found using grep
simval2<-hash()
for (let in listgrp){
    sal<-grep(let,temp)
    for(j in unique(temp[sal])){
        simval2[j]<-let
    }}

# replacing key-value pair in the container manually
simval2['tornadoes,tstmwind,hail']<-'tornado'

# replacing strings in the list with the similar string (obtained using grep)
for (j in temp){
     if (j %in% keys(simval2)){
    temp[which(j==temp)]<-simval2[[j]]}
}

# container to store key value pair of the mapping
simval3<-hash()
```

Once we have smaller list of word we can change entries that are similar and can be put in a group. After this we introduce a new column in the aggregated data set and this column contains the new event catgories. This number is smaller and we can use it to plot. The elements in the same category are summed and thus we get the total number corresponding to that event.

```
# There are a lot of similar words that can be grouped together however to do
# so I have to explicit create the mapping from the former to latter.

simval3["brushfire"]<-"wildfires"
```

```
simval3["coldtemperature"]<-"cold"
simval3["coldwave"]<-"cold"
simval3["coldweather"]<-"cold"
simval3["coldwindchill"]<-"cold"
simval3["coldwinds"]<-"cold"

simval3["dustdevil"]<-"duststorm"
simval3["densefog"]<-"fog"
simval3["drymircoburstwinds"]<-"drymicroburst"
simval3["drowning"]<-"marineaccident"

simval3["excessiveheat"]<-"extremeheat"
simval3["extendedcold"]<-"extremecold"
simval3["extremecoldwindchill"]<-"extremecold"
simval3["extremewindchill"]<-"extremecold"
simval3["excessiverainfall"]<-"heavyrain"

simval3["fogandcoldtemperatures"]<-"cold"
simval3["floodriverflood"]<-"flood"
simval3["flood&heavyrain"]<-"flood"
simval3["funnelcloud"]<-'tornado'
simval3["frost"]<- "blackice"
simval3["freezing"]<-"freeze"

simval3["gustywind"]<-"strongwind"
simval3["glaze"]<-"blackice"

simval3["highwind"]<-"strongwind"
simval3["heatwave"]<-"heat"
simval3["highwinds"]<-"strongwind"
simval3["highwindandseas"]<-"strongwind"
simval3["heavyseas"]<-"highseas"
simval3["highwater"]<-"highseas"
simval3["high"]<-"highseas"
simval3["hyperthermiaexposure"]<-"hypothermia"
simval3["iceonroad"]<-"iceroads"

simval3["lowtemperature"]<-"cold"
simval3["lightninginjury"]<-"lightning"

simval3["marinemishap"]<-"marineaccident"
simval3["marinehighwind"]<-"marinestrongwind"
simval3["mudslide"]<-"landslide"
simval3["mixedprecip"]<-"blizzard"

simval3["recordcold"]<-"extremecold"
simval3["recordheat"]<-"extremeheat"
simval3["rapidlyrisingwater"]<-"tsunami"
simval3["roguewave"]<-"roughseas"
simval3["recordexcessiveheat"]<-"recordheat"

simval3["sleet"]<-"hail"
simval3["smallhail"]<-"hail"
```

```
simval3["stormsurgetide"]<-"highseas"
simval3["stormsurge"]<-"highseas"

simval3["torrentialrainfall"]<-"heavyrain"

simval3["unseasonablywarm"]<-"unseasonablywarm"
simval3["unseasonablycold"]<-"unseasonablycold"
simval3["urbansmlstreamfld"]<-"urbansmlstreamfld"
simval3["urbanandsmallstreamfloodin"]<-"urbansmlstreamfld"
simval3["unseasonablywarmanddry"]<-"unseasonablywarm"

simval3["wind"]<-"winstorm"
simval3["warmweather"]<-"heat"
simval3["wintrymix"]<-"cold"
simval3["winterweather"]<-"cold"
simval3["winstorm"]<-"windstorm"
simval3["waterspout"]<-"tornado"
simval3["wildforestfire"]<-"wildfires"

# replacing similar strings with one unique value
for (j in temp){
     if (j %in% keys(simval3)){
    temp[which(j==temp)]<-simval3[[j]]}
}

# printing old list and new list. The number of unique element in the new list
# is less than 205 (length of original list).
print(cbind(evlist,temp)[1:10,])
```

```
##       evlist                     temp
##  [1,] "avalance"                 "avalance"
##  [2,] "avalanche"                "avalance"
##  [3,] "black ice"                "blackice"
##  [4,] "blizzard"                 "blizzard"
##  [5,] "blowing snow"             "snow"
##  [6,] "brush fire"               "wildfires"
##  [7,] "coastal flood"            "coastalflood"
##  [8,] "coastal flooding"         "coastalflood"
##  [9,] "coastal flooding/erosion" "coastalflood"
## [10,] "coastal storm"            "coastalstorm"
```

```
aggresult$EVENT2<-as.character(temp)

head(aggresult,5)
```

```
##           EVENT TOTALFATALITY TOTALINJURY   EVENT2
## 10     avalance             1           0 avalance
## 11    avalanche           224         170 avalance
## 19    black ice             1          24 blackice
## 20     blizzard           101         805 blizzard
## 32 blowing snow             2          14     snow
```

3.2. Results

Once we have reduced the space of unique weather event we aggregate the result for both TOTALFATALITY and TOTALINJURY into two different dataset at this point we can again check if we didn't miss any data by printing sum of fatalities and injuries over all the weather event and they are same as the one in original dataset. Thus, we can be sure that we didn't miss any dataset.

```
result1<-aggregate(aggresult[,c("TOTALFATALITY","TOTALINJURY")]
                     ,by=list(EVENT2=aggresult$EVENT2),FUN = sum)

resultF<-result1[order(result1$TOTALFATALITY,decreasing = TRUE),
                 c("EVENT2","TOTALFATALITY")]
resultI<-result1[order(result1$TOTALINJURY,decreasing = TRUE),
                 c("EVENT2","TOTALINJURY")]

#cbind(resultF,resultI)

cbind(TOTALFATALITY=sum(result1$TOTALFATALITY),
      TOTALINJURY=sum(result1$TOTALINJURY))
```

```
##      TOTALFATALITY TOTALINJURY
## [1,]         15145      140528
```

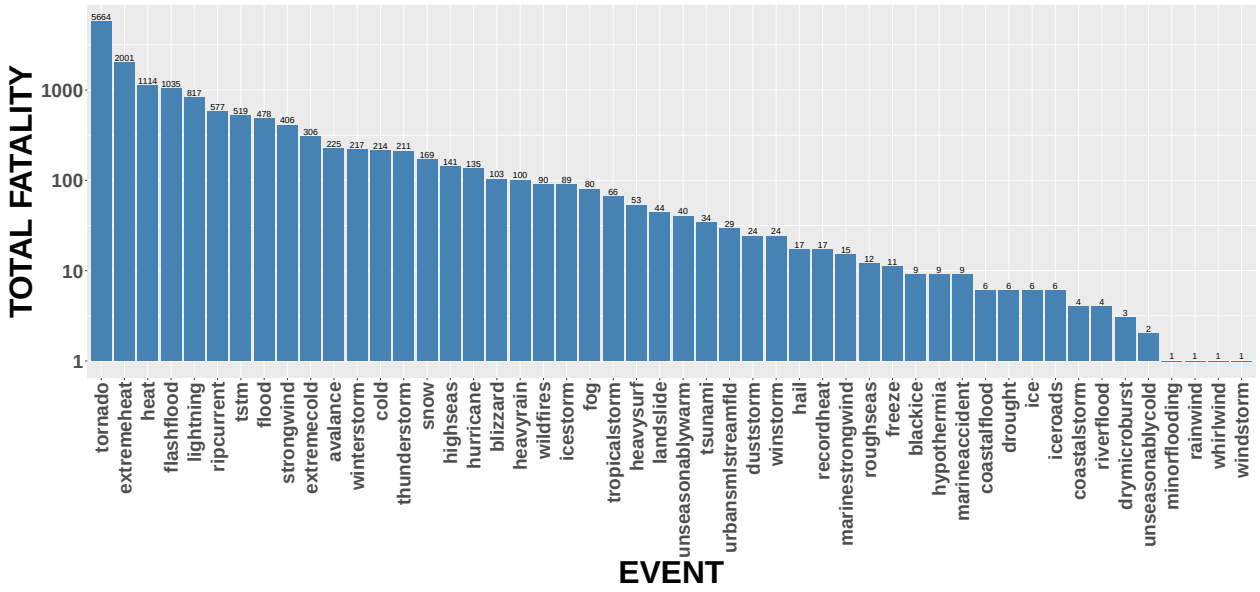
One best way to present our data is to plot both data corresponding to TOTALFATALITY and TOTALINJURY in two different plot. The x-axis is the weather event and y-axis is the total number of fatalities and injuries.

```
library(ggplot2)
library(ggeasy)
g<-ggplot(data=resultF[which(resultF$TOTALFATALITY>0),],
          aes(x=reorder(EVENT2,-TOTALFATALITY),y=TOTALFATALITY)) +
              geom_bar(stat="identity",fill="steelblue") +
  labs(caption="Total fatalities due to each weather event over the years")+
   theme(
        axis.title.x = element_text(face="bold",size=40),
        axis.title.y = element_text(face="bold",size=40),
        axis.text.x = element_text(face="bold",angle = 90,
                                   vjust = 0.5, hjust=1,size=24),
        axis.text.y = element_text(face="bold",angle = 0,
                                   vjust = 0.5, hjust=1,size=24),
        plot.caption=element_text(hjust=0.5,size=40))+
   xlab("EVENT")+ylab("TOTAL FATALITY") +scale_y_continuous(trans='log10')+
    geom_text(aes(label=TOTALFATALITY), vjust=-0.3, size=4.0)




p<-ggplot(data=resultI[which(resultI$TOTALINJURY>0),],
          aes(x=reorder(EVENT2,-TOTALINJURY),y=TOTALINJURY)) +
           geom_bar(stat="identity",fill="steelblue") +
  labs(caption="Total injuries due to each weather event over the years")+
   theme(
      axis.title.x = element_text(face="bold",size=40),
      axis.title.y = element_text(face="bold",size=40),
      axis.text.x = element_text(face="bold",angle = 90,
                                 vjust = 0.5, hjust=1,size=24),
```

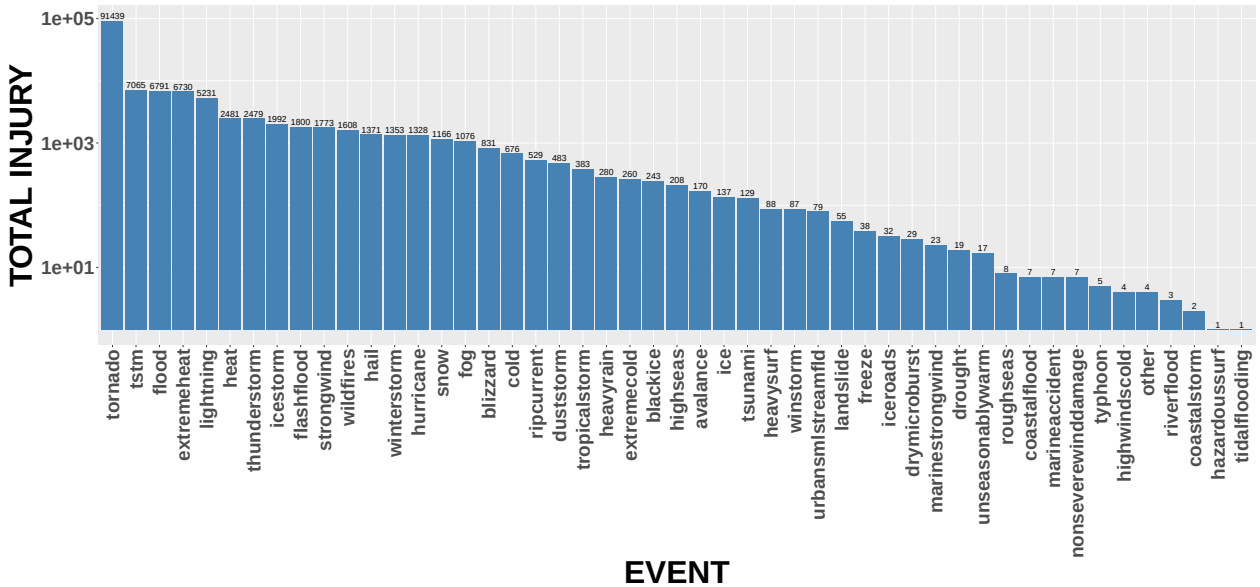
```
      axis.text.y = element_text(face="bold",angle = 0,
                                 vjust = 0.5, hjust=1,size=24),
     plot.caption=element_text(hjust=0.5,size=40))+
   xlab("EVENT")+ylab("TOTAL INJURY")  +scale_y_continuous(trans='log10')+
    geom_text(aes(label=TOTALINJURY), vjust=-0.3, size=3.8)



require("gridExtra")
```

```
## Loading required package: gridExtra
```

```
grid.arrange(arrangeGrob(g, p))
```



Total fatalities due to each weather event over the years



Total injuries due to each weather event over the years

Our plot clearly shows that tornadoes lead to both highest number of injuries and fatalities over the years.

4. Financial damages due to weather events

For this part of analysis we will need five columns for this analysis. In the columns CROPDMGEXP and PROPDMGEXP are the exponent values. These are multiplied by corresponding CROPDMG and PROPDMG to get the total value due to each event. A dictionary valdict maps values in these columns(CROPDMGEXP and PROPDMGEXP) with the corresponding numerical values. These values when multiplied by CROPDMG and PROPDMG gives total value of crop (TOTALCROPDMG column) and property (TOTALPROPDMG column) damage. The total damage(TOTALDMG column) gives the total value obtained after summing both total damage due to crops and due to properties for various event type.

```
# loading only relevant columns of the data
df<-data[,c("EVTYPE","PROPDMG","PROPDMGEXP","CROPDMG","CROPDMGEXP")]

# subset data for finite values of damage
dfdmg<-df[(as.numeric(df$PROPDMG)!=0.0 | as.numeric(df$CROPDMG)!=0.0),]
dfdmg$EVTYPE<-trimws(tolower(dfdmg$EVTYPE))

print(length(unique(dfdmg$EVTYPE)))
```

```
## [1] 394
```

4.1 Data Processing

For this part the preprocessing of data is required for total damage and event name as well. Each data set has DMG and DMGEXP for both PROP(properties) and CROP(crop). The DMGEXP column contains exponential of the cost associated. Each key is assigned to a value (information about this can be found on the course discussion group). This is first part of preprocessing for the second part we want to get unique event names.

```
library(hash)
#library(tidyverse)
# dictionary to store exponent and their numeric values
valdict<-hash()
valdict[["bl"]]<-0;valdict[['+']]<-1;valdict[['-']]<-0;valdict[['?']]<-0
valdict[['0']]<-10;valdict[['1']]<-10;valdict[['2']]<-10;valdict[['3']]<-10
valdict[['4']]<-10;valdict[['5']]<-10;valdict[['6']]<-10;valdict[['7']]<-10
valdict[['8']]<-10
valdict[['h']]<-100;valdict[['k']]<-1000
valdict[['m']]<-1000000;valdict[['b']]<-1000000000
valdict[['H']]<-100;valdict[['K']]<-1000
valdict[['M']]<-1000000;valdict[['B']]<-1000000000


# list of unique elements of CROPDMGEXP columns after removing blank ("") value

cvallist<-unique(dfdmg$CROPDMGEXP)
cvallist<-cvallist[-1]

# list of unique elements of PROPDMGEXP columns after removing blank ("") value

pvallist<-unique(dfdmg$PROPDMGEXP)
pvallist<-pvallist[-5]

print(tail(dfdmg))
```

```
##              EVTYPE PROPDMG PROPDMGEXP CROPDMG CROPDMGEXP
```

```
## 902249 winter storm     2.0          K       0          K
## 902250 winter storm     5.0          K       0          K
## 902255  strong wind     0.6          K       0          K
## 902257  strong wind     1.0          K       0          K
## 902259      drought     2.0          K       0          K
## 902260    high wind     7.5          K       0          K
```

```
# removing "" value in CROPDMGEXP with "bl". This helps in substitution
dfdmg$CROPDMGEXP<-ifelse(dfdmg$CROPDMGEXP %in% cvallist,dfdmg$CROPDMGEXP,"bl")

# replacing string value with appropriate value
dfdmg$CROPDMGEXP<-mapply(function(x) valdict[[x]],dfdmg$CROPDMGEXP)


# removing "" value in PROPDMGEXP column with "bl"
dfdmg$PROPDMGEXP<-ifelse(dfdmg$PROPDMGEXP %in% pvallist,dfdmg$PROPDMGEXP,"bl")

# replacing string value with a numeric value
dfdmg$PROPDMGEXP<-mapply(function(x) valdict[[x]],dfdmg$PROPDMGEXP)

# total property damage
dfdmg$TOTALPROPDMG<-dfdmg$PROPDMG*dfdmg$PROPDMGEXP

# total crop damage
dfdmg$TOTALCROPDMG<-dfdmg$CROPDMG*dfdmg$CROPDMGEXP

## calculating the total damage
dfdmg$TOTALDMG<-(dfdmg$TOTALPROPDMG + dfdmg$TOTALCROPDMG)

#print(tail(dfdmg))
#print(head(dfdmg[order(dfdmg$TOTALDMG,decreasing = TRUE)
#                 ,c("EVTYPE","PROPDMG","CROPDMG","TOTALDMG")],100))


# grouping data according to event type and summing data corresponding to same event.
aggresult<-aggregate(dfdmg[,c("TOTALDMG","TOTALPROPDMG","TOTALCROPDMG")]
                     ,by=list(EVENT=dfdmg$EVTYPE),FUN=sum)
```

We start by removing all the special characters from the name of events. This process helps in comparing different strings that are similar or have typos in them.

```
# removing special characters from the names of EVENTS
temp<-gsub(" ","",gsub("/","",aggresult$EVENT))
temp<-gsub("\\(|\\)","",temp)
temp<-gsub(pattern="\\\\",replacement="",temp)

temp<-gsub("-","",temp)
```

Two string are closer if the distance between them is less. Thus, we find string pairs that are close to each other and substitute them. This process of finding strings and replacing them is performed multiple times with different allowed distance between them. After that manually we remove those entries that are wrong. Once that is done we group similar strings into one. This allows us to combine data from similar strings and represent all those values with one. Also this reduced the unique values into fewer categories thus allowing us to visulaize them much more easily.

```
library(stringdist)

#############################################################################
#############################################################################
### similar words are returns as key-value pair based on distance
stringdst <- function(lstval,tolerance){
    dict<-hash()
    for(i in seq(1,length(lstval))){
        resl<-list()
        for(j in seq(i,length(lstval))){
            di<-stringdist(lstval[i],lstval[j])
            if(di<tolerance){
                resl<-append(resl,j)
                #print(c(lstval[i],lstval[j]))
                }

        }
        if(!(isTRUE(i %in% unlist(values(dict))))){
          dict[i]<-resl

        }}
    dict
}
#############################################################################
#############################################################################
#### replacing similar words from the list
replaceval<-function(lstval,valdict){
    temp2<-lstval
    for (i in keys(valdict)){
       for(j in valdict[[i]]){
        lstval[as.numeric(j)]<-temp2[as.numeric(i)]
       }

        }
    lstval
    }

#############################################################################
#############################################################################

## getting similar words based on distance and replacing them and putting results
## in a new list we do it mulitple times and remove wrong entries.
lst<-stringdst(temp,4)
temp3<-replaceval(temp,lst)

lst<-stringdst(temp3,5)
temp4<-replaceval(temp3,lst)

lst<-stringdst(temp4,6)
temp5<-replaceval(temp4,lst)

## by checcking these entries were completely wrong so replacing these entries
## with the original entries
```

```
wronidx<-c(5,11,16,17,19,21,29,41,62,71,73,74,79,92,93,96,97,109,110,111,113,
           117,119,126,129,138,140,142,143,146,148,149,151,152,153,166,168,178,
           179,182,185,186,188,190,196,199,200,201,202,208,213,227,228,229,236,237,
           239,240,241,247,249,263,272,273,275,276,278,290,292,293,295,296,300,303,
           313,316,321,322,323,324,325,326,327,329,336,350,352,353,361,363,363,
           375,376,377,382,380,381,385,386,388,389,390)
for(i in wronidx){
  temp5[i]<-temp[i]}
```

Once we have calculated the distance and placed similar together we remove the one we got wrong. After that we manually place these events into categories. This analysis is similar to one done in the first part.

```
library(stringr)

#### some entries are similar and they can be grouped together
temp6<-gsub("hail0.75","hail",temp5)

temp6<-gsub("floodflash","flashflood",temp6)
temp6<-gsub("thunderstormwind60mph","thunderstormwinds",temp6)
temp6<-gsub("thuderstormwinds","thunderstormwinds",temp6)
temp6[grep("smallhail",temp6)]<-"hail"
temp6[grep("^tornado",temp6)]<-"tornado"
temp6[grep("^highwind",temp6)]<-"strongwind"
temp6[grep("^coastal",temp6)]<-"coastalflood/surge/storm"

temp6[grep("snow",temp6)]<-"snow/heavysnow"
temp6[grep("fire",temp6)]<-"wildfire/lightningfire"
temp6[grep("cold",temp6)]<-"cold/extremecold"
temp6[grep("heat",temp6)]<-"heat/extremeheat"
temp6[grep("freeze",temp6)]<-"freeze"
temp6[grep("^tstm",temp6)]<-"tstmwind"
temp6[grep("winter",temp6)]<-"winterweather/storms"
temp6[grep("^floo",temp6)]<-"floods"
temp6[grep("^thunder",temp6)]<-"thunderstorm"
temp6[grep("^freezin",temp6)]<-"freezingrain"
temp6[grep("^highs",temp6)]<-"highseas"
temp6[grep("^hight",temp6)]<-"highseas"
temp6[grep("heavyrain",temp6)]<-"heavyrain"
temp6[grep("^heavyp",temp6)]<-"heavyrain"
temp6[grep("^heavysh",temp6)]<-"heavyrain"
temp6[grep("flashflood",temp6)]<-'flashflood'
temp6[grep("flooding",temp6)]<-"floods"
temp6[grep("gusty",temp6)]<-"gustywind/hail/rain"
temp6[grep("blizzard",temp6)]<-"blizzard"
temp6[grep("tropical",temp6)]<-"tropicalstorm"
temp6[grep("hurricane",temp6)]<-"hurricane"
temp6[grep("^ic",temp6)]<-"iceroad/storm/flood"
temp6[grep("smal",temp6)]<-"smallflood/urban"
temp6[grep("^urb",temp6)]<-"smallflood/urban"
temp6[grep("^river",temp6)]<-"smallflood/river"
temp6[grep("^hail",temp6)]<-"hail"
temp6[grep("^wind",temp6)]<-"strongwind"
temp6[grep("^ligh",temp6)]<-"lightning/rain/storm"
temp6[grep("lightning",temp6)]<-"lightning/rain/storm"
```

```
temp6[grep("torndao",temp6)]<-"tornado"
temp6[grep("^strong",temp6)]<-"strongwinds"
temp6[grep("lake",temp6)]<-"lakeflood"
temp6[grep("thund",temp6)]<-"thunderstorm"
temp6[grep("^heavys",temp6)]<-"highseas"
temp6[grep("highwater",temp6)]<-"highseas"
temp6[grep("hvyrain",temp6)]<-"heavyrain"
temp6[grep("^marine",temp6)]<-"marineevents"
temp6[grep("rain",temp6)]<-"heavyrain"
temp6[grep("rainstorm",temp6)]<-"heavyrain"
temp6[grep("recordrainfall",temp6)]<-"heavyrain"
temp6[grep("excessivewetness",temp6)]<-"heavyrain"
temp6[grep("roughsurf",temp6)]<-"highseas"

temp6[1]<-str_replace(temp6[1],"[[:punct:]]","Unknown")
```

Once all the string values are categorized I replace the event name with the category. Group data for each category and data in same category is summed. This is done for TOTALPROPDMG, TOTALCROPDMG and TOTALDMG.

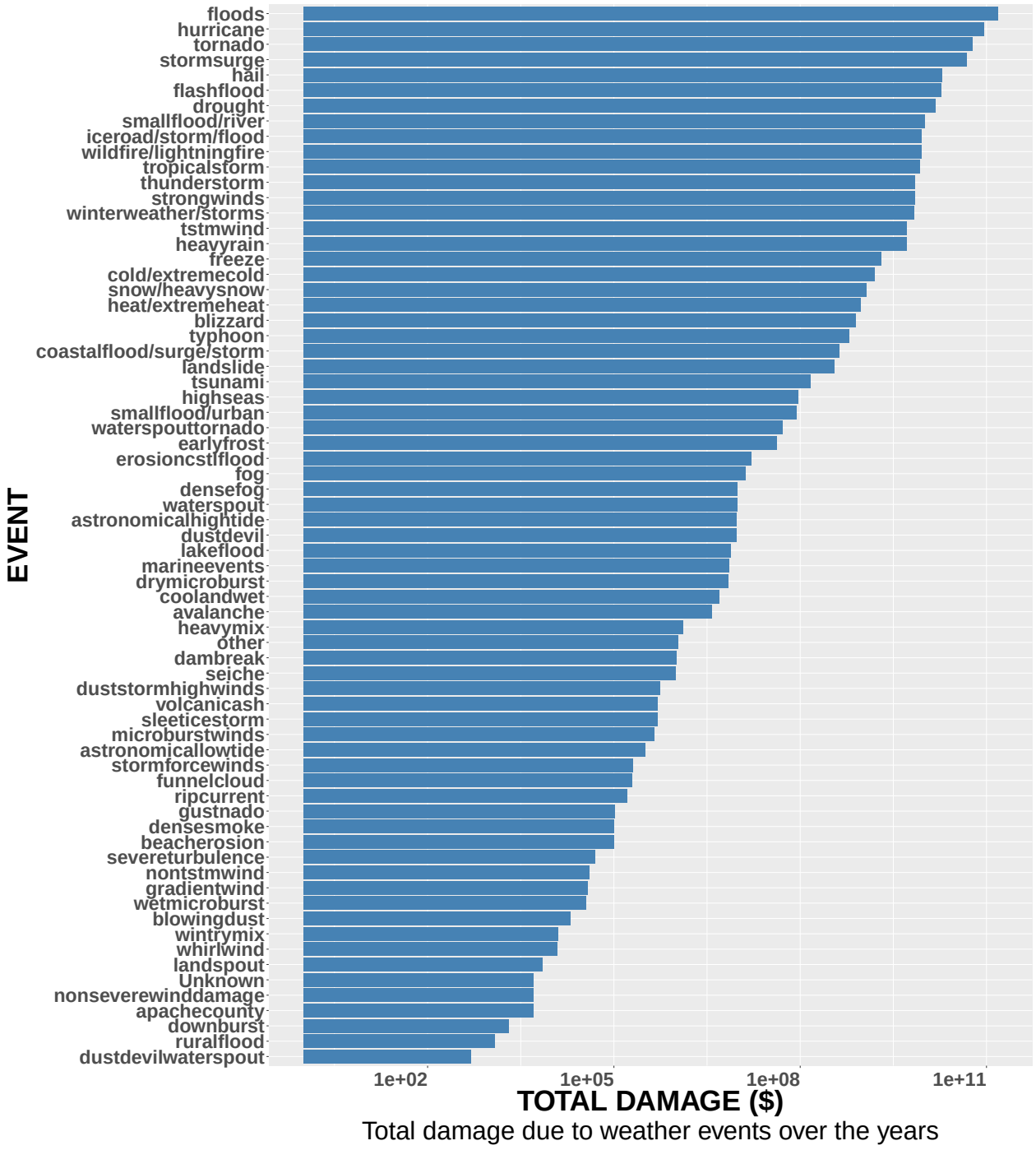
```
aggresult$EVENT2<-as.character(temp6)

result1<-aggregate(aggresult[,c("TOTALDMG","TOTALPROPDMG","TOTALCROPDMG")]
                     ,by=list(EVENT2=aggresult$EVENT2),FUN = sum)

resultT<-result1[order(result1$TOTALDMG,decreasing = TRUE),
                 c("EVENT2","TOTALDMG")]
resultP<-result1[order(result1$TOTALPROPDMG,decreasing = TRUE),
                 c("EVENT2","TOTALPROPDMG")]
resultC<-result1[order(result1$TOTALCROPDMG,decreasing = TRUE),
                 c("EVENT2","TOTALCROPDMG")]
```

4.2. Results

The TOTALDMG for each event is plotted. Since TOTALDMG scale is really large it's a good idea to use a logarithmic scale. This looks better otherwise larges scale overwhelms the data on smaller scales. Here we have large numbe of categories so we plot categories on y axis and TOTALDMG on x axis. From the data we can see flood does most amount of damage, followed by hurricane and tornado. The total damage is sum of both the damage to properties as well as to crops.

```
library(ggplot2)
library(ggeasy)
g<-ggplot(data=resultT,
          aes(y=reorder(EVENT2,TOTALDMG),x=TOTALDMG)) + geom_col(fill="steelblue")+
   labs(caption='Total damage due to weather events over the years')+
    theme(
        axis.title.x = element_text(face="bold",size=36),
        axis.title.y = element_text(face="bold",size=36),
        axis.text.x = element_text(face="bold",angle = 0,
                                   vjust = 0.5, hjust=1,size=24),
        axis.text.y = element_text(face="bold",angle = 0,
                                   vjust = 0.5, hjust=1,size=24),
        plot.caption=element_text(hjust=0.5,size=32))+
       ylab("EVENT")+xlab("TOTAL DAMAGE ($)") +scale_x_continuous(trans='log10')
```

```
require("gridExtra")
grid.arrange(arrangeGrob(g))
```



Results

Thus our results for the first part shows that tornadoes lead to highest number of fatalities as well as injuries on the other hand floods were the highest contributor in terms of total damage (in dollars) over the years.

Useful links.

1. [Data] (https://d396qusza40orc.cloudfront.net/repdata%2Fdata%2FStormData.csv.bz2)
2. [ggplot related] (http://www.sthda.com/english/wiki/ggplot2)