MA415 Final Project: Victims of Fire Incident in the US

Preparation of Data

Sarah Gore- U72145380

This version: May 05, 2017

Contents

Available versions of this document: html–notebook version, pdf version, R script alone, html with full code version. These files can be found on my GitHub.

1 Source of the Data Set

I am using the data from the National Fire Incident Reporting System (NFIRS 5.0) over the 2006-2015 period.

> "NFIRS 5.0 is a modular, all-incident reporting system designed by the U.S. Fire Administration, a part of the Department of Homeland Security, with input from the fire service and other users of the data."

This data, cleaned in the way described below, will be used in my Shiny App to dynamically visualize the results of my analysis.

2 Focus of the Project

With an overwhelming amount of data, I need to select the variables that are in line with my focus.

I have decided to focus on the victims of fire incidents in the US. More specifically, the number as well as their characteristics such as their location, gender, age etc. I aim at finding patterns within these variables.

The structure of the analysis therefore answering three key questions about the victims:

1. Where - where are the victims located
2. Who - what are the characteristics of the victims
3. When - when are there more victims, over time, period of year etc...

3 Importing the Data

The code below was used to import the data. I noticed that the variables and files were the same for each year, so I built a function to speed up the process and make the code cleaner. I noticed that the data files were very large and it took a significant time to load. Therefore, I ran the data import during once preparation. I then saved it into a file that will later be used in the analysis.

3.1 Loading the Required Libraries

```
knitr::opts_chunk$set(echo = TRUE, tidy = FALSE, cache = TRUE)
require(ggplot2)
require(tidyr)
require(dplyr)
require(readr) # to read text
require(foreign) # library used to import the dbfs
require(lubridate) # library used for dates/times
require(zipcode) # to find the coordinates of the zip in case
# I want to match something with coordinates
require(noncensus) # to find the population per state in case
# I want to use this information
require(stringr) # to work on strings
```

3.2 The Import Function

The function below imports the data. Since some of the files are "dbf" and others "txt", I insert a condition in order to use the same function for both type of files.

The main idea of the function below is the following:

- load the files with the required information
  - basicincident: I keep incident and arrival time
  - fireincident: I keep the key in order to link the data to the file civiliancasualty.
  - incidentaddress: I keep the zip codes of the incidents
  - civiliancasualty: I keep data relating to characteristics of the victims such severity of the injury, age of victims etc.
- merge the information of the files using the keys

```
get.data <- function(path, type){
  if (type == "dbf"){

    bi <- read.dbf(paste(path, "basicincident.dbf", sep = "/"), as.is = TRUE) %>%
      tbl_df
    fi <- read.dbf(paste(path, "fireincident.dbf", sep = "/"), as.is = TRUE) %>%
      tbl_df
    ia <- read.dbf(paste(path, "incidentaddress.dbf", sep = "/"), as.is = TRUE) %>%
      tbl_df
    cc <- read.dbf(paste(path, "civiliancasualty.dbf", sep = "/"), as.is = TRUE) %>%
      tbl_df
  }

  if (type == "txt") {
    bi <- read_delim(paste(path, "basicincident.txt", sep = "/"), "^",
                     col_types = cols(.default = "c")) %>%
      tbl_df
    fi <- read_delim(paste(path, "fireincident.txt", sep = "/"), "^",
                     col_types = cols(.default = "c")) %>%
      tbl_df
    ia <- read_delim(paste(path, "incidentaddress.txt", sep = "/"), "^",
                     col_types = cols(.default = "c")) %>%
      tbl_df
    cc <- read_delim(paste(path, "civiliancasualty.txt", sep = "/"), "^",
                     col_types = cols(.default = "c")) %>%
      tbl_df
  }

  id <- c("STATE", "FDID", "INC_DATE", "INC_NO", "EXP_NO")
  # these are the keys for the tables

  cc <-cc %>%
    select(STATE, FDID, INC_DATE, INC_NO, GENDER, AGE, RACE,
           SEV, CAUSE_INJ, SEQ_NUMBER,EXP_NO)


# I select the variables of interest in cc

  aux1 <- left_join(fi, bi, by = id)
  aux2 <- left_join(aux1, ia, by = id)
  aux3 <- aux2 %>%
    select(STATE, FDID, INC_DATE, INC_NO, EXP_NO, INC_TYPE, ALARM, ARRIVAL, ZIP5)

  aux4 <- left_join(cc, aux3, by = id) %>%
    tbl_df
  colnames(aux4) <-tolower(colnames(aux4)) # convert the variables to lower case


# Convert integer variables into characters for allowing binding below

  aux4$inc_date <-as.character(aux4$inc_date)
  aux4$alarm <-as.character(aux4$alarm)
  aux4$arrival <-as.character(aux4$alarm)
```

```
  aux4$exp_no <-as.character(aux4$exp_no)
  aux4$age <-as.character(aux4$age)
  aux4$seq_number <-as.character(aux4$seq_number)

  return(aux4)
}

# Uncomment to load the data

# d06 <- get.data("2006/2006/NFIRS_2006_040108", "dbf")
# d07 <- get.data("2007/NFIRS_2007_042309", "dbf")
# d08 <- get.data("2008/NFIRS_2008_011910", "dbf")
# d09 <- get.data("2009/NFIRS_2009_092710", "dbf")
# d10 <- get.data("2010/NFIRS_2010_100711", "dbf")
# d11 <- get.data("2011/NFIRS_2011_120612", "dbf")
# # For this year I needed to change the name of the file casualties to
# # civiliancasualty for consistency in the function
# d12 <- get.data("2012/NFIRS_2012_052714", "txt")
# d13 <- get.data("2013/NFIRS_2013_121514", "txt")
# d14 <- get.data("2014/NFIRS_2014_030216", "txt")
# d15 <- get.data("2015/NFIRS_FIRES_2015_20170215","txt")

# df_cc <- bind_rows(d06$cc, d07$cc, d08$cc, d09$cc, d10$cc,
# d11$cc, d12$cc, d13$cc, d14$cc, d15$cc)

# save(df_cc, file = "df_cc.Rdata")

# Load the data when needed
load("df_cc.Rdata")
```

3.3 Transforming some variables

I imported the variables as characters. Here I correct for the right class. The information comes from the provided the provided file `codelookup.txt`.

```
df_cc$age <- as.numeric(df_cc$age)
df_cc$age[df_cc$age<1] <- NA
df_cc$age[df_cc$age>100] <- NA

lut <- c("1"= "Male", "2"= "Female")
df_cc$gender <- factor(lut[df_cc$gender])

lut <- c("1"="Minor", "2"="Moderate","3"="Severe", "4"="Life threatening", "5"="Death", "U"="Undetermin
df_cc$sev <- factor(lut[df_cc$sev], levels = c("Minor", "Moderate","Severe", "Life threatening", "Death

#lut <- c("0"="Other","1"="White","2"="Black","3"="American Indian, Eskimo or Aleut","4"="Asian","U"="U
# very few 3 and 4

lut <- c("0"="Other","1"="White","2"="Black","3"="Other","4"="Other","U"="Other")
df_cc$race <- factor(lut[df_cc$race], levels= c("Black", "White", "Other"))
```

4 Adding Location-Based Information to Clean Data

It would be straightforward to add location coordinates to my data using the address provided in the data and a the R command `geocode` that makes queries to a Google API. However, the number of requests for this project exceeds the allowance (it only allows me to run 2500 free requests per day or 100 000 if I pay; if I were to pay this fee, it would still take days to get all the geocodes of all the observations).

After some research mainly through stack overflow (http://stack overflow.com/questions/13316185/r-convert-zipcode-or-lat-long-to-county), I found alternative methods to tackle this problem. Some of them were less accurate than others.

I have settled on using a method based on the zipcodes for location of the incidents and link them to their respective counties.

I want to acknowledge that this method gives approximations and may result in some inconsistencies. These inconsistencies would have been avoided had I been able to use `geocode`. However, I still use this code to demonstrate what I have learnt in R.

```
# source:
# http://mcdc2.missouri.edu
zip <- read_csv("zip.csv", col_types = cols(.default = "c")) %>%
select(zcta5, County) %>%
rename(zip5 = zcta5 , county = County)
zip$county <- tolower(substr(zip$county, 1, nchar(zip$county)-3))

data(zipcode)
zipco <-zipcode %>%
select(zip, latitude, longitude)
zip <- inner_join(zipco, zip, by = c("zip" = "zip5"))

# join the zipcode information to the data

df_cc <- left_join(df_cc, zip, by = c("zip5" = "zip"))
```

5 Adding Population-Based Information to Clean Data

In order to make relevant comparisons I require the population per unit.I use the package `nonconsensus` that provides me with 2010 census data. 2010 falls in the middle of the period being analyzed and I therefore consider the data as a good benchmark.

```
data(counties)
mycounties <- counties %>%
select(county_name, state, population)
mycounties$county_name <- tolower(mycounties$county_name)
mycounties$county_name <- mycounties$county_name %>%
str_replace(" county", "")

# I noticed that the county names include the word "county".
# I need to get rid of this string in order to be able to merge this data.

df_cc <- left_join(df_cc, mycounties, by = c("county" = "county_name", "state" = "state"))
df_cc$population <- as.numeric(df_cc$population)
data(states)
mystates <- states %>%
```

```
select(state, population) %>%
  rename(state_pop = population)

df_cc <- left_join(df_cc, mystates, by = "state")
df_cc$state_pop <- as.numeric(df_cc$state_pop)
lut <- c("AK"="Alaska", "AL"="Alabama", "AR"="Arkansas", "AZ"="Arizona",
"CA"="California", "CO"="Colorado", "CT"="Connecticut",
"DC"="District of Columbia", "DE"="Delaware", "FL"="Florida",
"GA"="Georgia", "HI"="Hawaii", "IA"="Iowa","ID"="Idaho", "IL"="Illinois",
"IN"="Indiana", "KS"="Kansas", "KY"="Kentucky",
"LA"="Louisiana","MA"="Massachusetts", "MD"="Maryland", "ME"="Maine",
"MI"="Michigan","MN"="Minnesota", "MO"="Missouri", "MS"="Mississippi",
"MT"="Montana", "NC"="North Carolina","ND"="North Dakota", "NE"="Nebraska",
"NH"="New Hampshire", "NJ"="New Jersey", "NM"="New Mexico", "NV"="Nevada",
"NY"="New York", "OH"="Ohio", "OK"="Oklahoma", "OR"="Oregon", "PA"="Pennsylvania",
"RI"="Rhode Island", "SC"="South Carolina", "SD"="South Dakota", "TN"="Tennessee",
"TX"="Texas", "UT"="Utah", "VA"="Virginia", "VT"="Vermont",
"WA"="Washington", "WI"="Wisconsin", "WV"="West Virginia", "WY"="Wyoming")
      df_cc$region <- factor(lut[df_cc$state])
 # This is done in order to add a column with the full name of the state which is used for merging
```

6 Adding Date-Based Information to Clean Data

In order to use the information based on the time of the incident, I need to transform the appropriate variables using `lubridate`.

```
df_cc$inc_date <- mdy(as.numeric(df_cc$inc_date))
# as numeric because this variable was previously a character
df_cc$alarm <- mdy_hm(as.numeric(df_cc$alarm))
df_cc$arrival <- mdy_hm(as.numeric(df_cc$arrival))
# I am only interested in the day of the incidient as
# some of the arrival times do not conincide with the day of the incident
```

7 Data for Mapping

In the ShinyApp I will be using the data coming from ggplot2 in order to map the data. The map below, is the basis for my maps. Information is then merged to the map data in order to render the results

See the case below, using Massachusetts as an example:

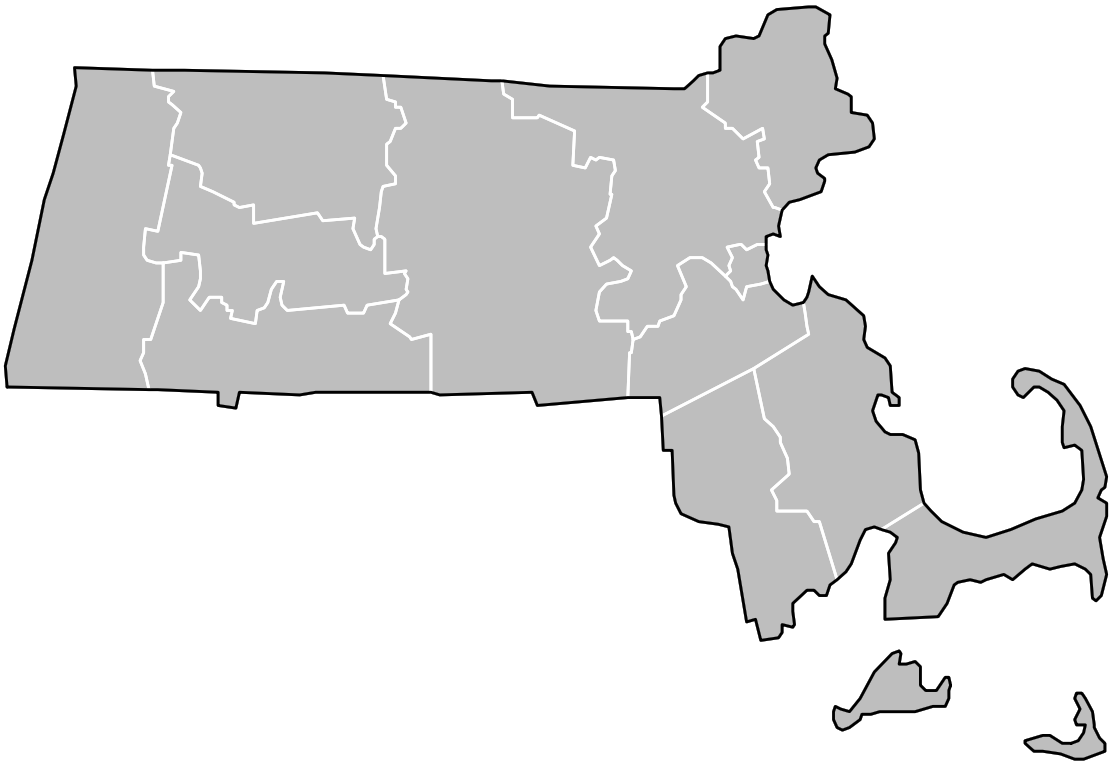
```
states <- map_data("state")
county <- map_data("county")

ma_eg <- states %>%
filter(region == "massachusetts")
ma_county <- county %>%
filter(region == "massachusetts")

# This gets rid of the axes and grid.
# http://eriqande.github.io/rep-res-web/lectures/making-maps-with-R.html
```

```
ditch_the_axes <- theme(axis.text = element_blank(),
                        axis.line = element_blank(),
                        axis.ticks = element_blank(),
                        panel.border = element_blank(),
                        panel.grid = element_blank(),
                        axis.title = element_blank()
                        )
ma_base <- ggplot(data = ma_eg, mapping = aes(x = long, y = lat, group = group)) +
  geom_polygon(color = "black", fill = "grey")

ma_base +
geom_polygon(data = ma_county, fill = NA, color = "white") +
geom_polygon(color = "black", fill = NA) +
  theme_bw() +
  ditch_the_axes
```



8 Saving the Data

Finally, after the cleaning process, I save the data set so that it can then be used by the ShinyApp.

```
df <- df_cc
save(df, file = "MA415_finalproject/shiny_data.Rdata")
```

- I made use of the following resources:
- class notes
- stackoverflow
- cheat sheets

- I acknowledge that I benefited from the help of a tutor with whom I work regularly. He reviewed my code and offered some suggestions of improvement. I remain solely responsible of this work.
- I hope you found this as interesting as I did!