Midterm Report

Haode Qi

March 21, 2017

lets load the library for data cleaning and read relevant data. Since we trying to figure where is the most dangerous places to work in Masschusetts, osha table and accident table will be the most relevant data we need for the purpose.

```
library(foreign)
library(dplyr)
```

```
## Warning: package 'dplyr' was built under R version 3.2.5
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

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

```
library(magrittr)
library(tidyr)
```

```
## Warning: package 'tidyr' was built under R version 3.2.5
```

```
##
## Attaching package: 'tidyr'
```

```
## The following object is masked from 'package:magrittr':
##
##     extract
```

```
osha <- read.dbf("osha.DBF")

accid<- read.dbf("accid.dbf")
```

Next step, we need to examine the data along with the documentation provided.

```
str(accid)
```

```
## 'data.frame':    2147 obs. of  16 variables:
##  $ ACTIVITYNO: int  10096592 10096592 10096592 10096592 10096592 305548745 306811928 306812744 30554
##  $ SITESTATE : Factor w/ 1 level "MA": 1 1 1 1 1 1 1 1 1 1 ...
```

```
##  $ NAME      : Factor w/ 822 levels "000000000000000000",..: NA NA NA NA NA 489 129 617 80 571 ...
##  $ RELINSP   : int  10096592 10096592 10096592 10096592 10096592 305548745 306811928 306812744 305545
##  $ SEX       : Factor w/ 2 levels "F","M": NA NA NA NA NA 2 2 2 2 2 ...
##  $ DEGREE    : Factor w/ 4 levels "0","1","2","3": 4 4 4 4 4 2 2 2 2 2 ...
##  $ NATURE    : Factor w/ 20 levels "00","01","02",..: 20 20 20 20 20 20 20 17 20 7 ...
##  $ BODYPART  : Factor w/ 31 levels "00","01","02",..: 5 5 5 5 5 20 21 6 20 14 ...
##  $ SOURCE    : Factor w/ 39 levels "00","01","02",..: 15 15 15 15 15 14 38 19 24 3 ...
##  $ EVENT     : Factor w/ 15 levels "00","01","02",..: 9 9 9 9 9 6 3 15 3 7 ...
##  $ ENVIRON   : Factor w/ 18 levels "00","01","02",..: 8 8 8 8 8 18 7 6 9 16 ...
##  $ HUMAN     : Factor w/ 21 levels "00","01","02",..: 7 7 7 7 7 21 15 11 2 5 ...
##  $ TASK      : Factor w/ 3 levels "0","1","2": 3 3 3 3 3 2 2 2 2 2 ...
##  $ HAZSUB    : Factor w/ 56 levels "0000","0010",..: NA NA NA NA NA NA NA NA NA NA ...
##  $ OCC_CODE  : Factor w/ 103 levels "000","004","019",..: 1 1 1 1 1 45 47 42 1 102 ...
##  $ AGE       : int  0 0 0 0 0 47 39 22 62 47 ...
##  - attr(*, "data_types")= chr  "N" "C" "C" "N" ...
```

```
# we can see that there are 2000+ observations
length(unique(accid$ACTIVITYNO))
```

```
## [1] 1570
```

```
# but only 1570 unique observations, we need to look at the duplicated data and see what we can do abou

#accid$ACTIVITYNO[duplicated(accid$ACTIVITYNO)]

noneunique<- subset(accid, duplicated(accid$ACTIVITYNO) ==TRUE)

# for the first four record, all the entry is exactly the same

head(noneunique,10)
```

```
##    ACTIVITYNO SITESTATE              NAME   RELINSP  SEX DEGREE NATURE
## 2    10096592        MA              <NA>  10096592 <NA>      3     21
## 3    10096592        MA              <NA>  10096592 <NA>      3     21
## 4    10096592        MA              <NA>  10096592 <NA>      3     21
## 5    10096592        MA              <NA>  10096592 <NA>      3     21
## 15   10109288        MA              <NA>  10109288 <NA>      2     12
## 41  302923149        MA              <NA> 302923149    M      2     10
## 43  116301367        MA              <NA> 116301367    M      2     04
## 49  302919048        MA              <NA> 302919048    M      2     17
## 50  302919048        MA              <NA> 302919048    F      2     17
## 70   10054336        MA 000000000000000000  10054336 <NA>      1     12
##    BODYPART SOURCE EVENT ENVIRON HUMAN TASK HAZSUB OCC_CODE AGE
## 2        04     16    08      07    06    2   <NA>      000   0
## 3        04     16    08      07    06    2   <NA>      000   0
## 4        04     16    08      07    06    2   <NA>      000   0
## 5        04     16    08      07    06    2   <NA>      000   0
## 15       21     29    01      06    09    2   <NA>      000   0
## 41       04     15    13      13    01    1   8870      869  38
## 43       09     09    10      09    03    1   T104      683  30
## 49       04     09    08      09    11    1   <NA>      706  28
## 50       04     09    08      09    11    1   <NA>      706  40
## 70       19     42    05      13    12    1   <NA>      000   0
```

```
# but if we examine the other record, not all the entries with the same activity number are the same
# the fact that this may occur because there may be multiple individuals involved in any accident and t
# we cannot simply removed the duplicated records

# a very important indicator of the seriouness of an accident is the degree. 1 for fatality, 2 for hosp
# it will probably be worthwhile to see if the same activityno also entails the same degree of seriousn
#if they are duplicated in the exact same way, the above statement is true

table(duplicated(accid$ACTIVITYNO) == duplicated(accid$DEGREE))
```

```
##
## FALSE  TRUE
##  1568   579
```

This table tells us that it is not

One way we can eleminate the duplication record is to accumulate the degree of seriousness and number of individuals involved in each incident.

```
accid$DEGREE%<>% as.character() %>% as.numeric()

uniqueaccid <- accid %>% group_by(ACTIVITYNO)


test3 <- accid %>% group_by(ACTIVITYNO) %>% filter(DEGREE == 3)
test2 <- accid %>% group_by(ACTIVITYNO) %>% filter(DEGREE == 2)
test1 <- accid %>% group_by(ACTIVITYNO) %>% filter(DEGREE == 1)

#fatality
test1 %<>% summarise(count = n())
colnames(test1) <- c("ACTIVITYNO", "Fatality")

#hospitality
test2 %<>% summarise(count = n())
colnames(test2) <- c("ACTIVITYNO", "Hospitality")

#non-hospitality
test3 %<>% summarise(count = n())
colnames(test3) <- c("ACTIVITYNO", "Non-hospitality")

uniqueaccid %<>% summarise(cumulativeDegree = sum(DEGREE), count = n())

#join them one by one
uniqueaccid <- uniqueaccid %>% left_join(test1, by = "ACTIVITYNO")
uniqueaccid <- uniqueaccid %>% left_join(test2, by = "ACTIVITYNO")
uniqueaccid <- uniqueaccid %>% left_join(test3, by = "ACTIVITYNO")

head(uniqueaccid,20)
```

```
## # A tibble: 20 × 6
##    ACTIVITYNO cumulativeDegree count Fatality Hospitality
##         <int>            <dbl> <int>    <int>       <int>
## 1      141879                1     1        1          NA
```

```
## 2      142349                1     1        1          NA
## 3      142455                1     1        1          NA
## 4      142737                1     1        1          NA
## 5      159020                1     1        1          NA
## 6      159319                1     1        1          NA
## 7      159475                1     1        1          NA
## 8      159996                1     1        1          NA
## 9      922690                1     1        1          NA
## 10     923946                1     1        1          NA
## 11     924563                0     1       NA          NA
## 12     926097                1     1        1          NA
## 13     926303                1     1        1          NA
## 14     927541                5     3        1           2
## 15     927590                1     1        1          NA
## 16     927897                1     1        1          NA
## 17     927939                2     2        2          NA
## 18     949008                2     1       NA           1
## 19     954289                1     1        1          NA
## 20     954750                1     1        1          NA
## # ... with 1 more variables: `Non-hospitality` <int>
```

```
uniqueaccid %<>% mutate(averageDegree = round(cumulativeDegree/count,3))

# the NAs in each type of injury simply means that is zero number of people involved in it. We should r
uniqueaccid[is.na(uniqueaccid)] <-0


#just to test to see if we do it correctly
uniqueaccid[uniqueaccid$ACTIVITYNO == "10096592",]
```

```
## # A tibble: 1 × 7
##   ACTIVITYNO cumulativeDegree count Fatality Hospitality `Non-hospitality`
##        <int>            <dbl> <int>    <dbl>       <dbl>              <dbl>
## 1   10096592               15     5        0           0                  5
## # ... with 1 more variables: averageDegree <dbl>
```

now we have the list of unique accidents and the cumulative seriouness and the associated number of individuals involved. the address information is stored in osha and we can retrieve the information by matching the activty number

In the osha documentation, the following are the address info. States are all MA, but i prefer to keep state and it become relevant later on in maping. SITE STREET Street Address SITE STATE State Code (alpha postal abbreviation) SITE ZIP United States Postal Zip Code SITE CITY CODE Department of Commerce City Code SITE CNTY CODE

Since the primary purpose of this ste[] is to retrieve the relevant information, NAs and duplications are no less of a concern here. we will worry about it later. Though some of the colnames seems to be overlylong like SITESTATE OR SITEADD, it is necessary to keep them the way it is incase we need to join any information with other table with the same colnames.

```
#glimpse(osha)

#we may want to keep the date information. I actually did examine all the different dates avaliable but
```

```
#str(osha$OSHA1MOD)
#str(osha$OPENDATE)
#str(osha$CLOSEDT)
#str(osha$CLOSEDT2)
#str(osha$CLOSEDATE)
#str(osha$CLOSEDATE2)
#str(osha$OPENDT)
```

Next, we move on to the address information and match it by activity number.

```
oshaaddress <- osha[,c("ACTIVITYNO" , "SITEADD", "SITESTATE", "SITEZIP", "SITECITY", "SITECNTY" ) ]

uniqueaccid %<>% left_join(oshaaddress,by = "ACTIVITYNO")

#just to illustrate some of the things we can do with the cleaned data, the following a frequency table
summarise(group_by(uniqueaccid,SITEZIP),count= n())
```

```
## # A tibble: 383 × 2
##    SITEZIP count
##     <fctr> <int>
## 1    01001     5
## 2    01002     1
## 3    01005     1
## 4    01007     2
## 5    01008     1
## 6    01010     2
## 7    01011     1
## 8    01013     4
## 9    01020     6
## 10   01022     2
## # ... with 373 more rows
```

If we look at the lookup database, we can probably retrieve the city name information instead of encoding.

```
scc <- read.dbf("lookups/scc.dbf")

#by looking at scc and do some googling, the names are indeed city names, since some of the encoding ma
glimpse(scc)
```

```
## Observations: 43,355
## Variables: 5
## $ TYPE   <int> 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2...
## $ STATE  <fctr> AK, AK, AK, AK, AK, AK, AK, AK, AK, AK, AK, AK, AK, AK...
## $ COUNTY <fctr> 000, 010, 013, 016, 020, 050, 060, 068, 070, 090, 100,...
## $ CITY   <fctr> 0000, 0000, 0000, 0000, 0000, 0000, 0000, 0000, 0000, ...
## $ NAME   <fctr> ALASKA, ALEUTIAN ISLANDS, ALEUTIANS EAST, ALEUTIANS WE...
```

```
head(scc)
```

```
##   TYPE STATE COUNTY CITY            NAME
## 1    1    AK    000 0000          ALASKA
```

```
## 2    2    AK    010 0000 ALEUTIAN ISLANDS
## 3    2    AK    013 0000   ALEUTIANS EAST
## 4    2    AK    016 0000   ALEUTIANS WEST
## 5    2    AK    020 0000        ANCHORAGE
## 6    2    AK    050 0000           BETHEL
```

```
colnames(scc) <- c("TYPE", "SITESTATE", "SITECNTY", "SITECITY", "NAME")

testname <- scc[,c("SITESTATE","SITECITY","NAME")]


uniqueaccid  %<>%  left_join(testname,by = c("SITESTATE","SITECITY"))
```

```
## Warning in left_join_impl(x, y, by$x, by$y, suffix$x, suffix$y): joining
## factors with different levels, coercing to character vector

## Warning in left_join_impl(x, y, by$x, by$y, suffix$x, suffix$y): joining
## factors with different levels, coercing to character vector
```

```
head(uniqueaccid)
```

```
## # A tibble: 6 × 13
##   ACTIVITYNO cumulativeDegree count Fatality Hospitality `Non-hospitality`
##        <int>            <dbl> <int>    <dbl>       <dbl>              <dbl>
## 1     141879                1     1        1           0                  0
## 2     142349                1     1        1           0                  0
## 3     142455                1     1        1           0                  0
## 4     142737                1     1        1           0                  0
## 5     159020                1     1        1           0                  0
## 6     159319                1     1        1           0                  0
## # ... with 7 more variables: averageDegree <dbl>, SITEADD <fctr>,
## #   SITESTATE <chr>, SITEZIP <fctr>, SITECITY <chr>, SITECNTY <fctr>,
## #   NAME <fctr>
```

```
dangerouscity<-summarise(group_by(uniqueaccid,NAME),incidence = n(), total= sum(count))

arrange(dangerouscity,desc(total))
```

```
## # A tibble: 257 × 3
##           NAME incidence total
##         <fctr>     <int> <int>
## 1       BOSTON       162   198
## 2      DANVERS        12    86
## 3    CAMBRIDGE        61    66
## 4    HAVERHILL        42    52
## 5     LAWRENCE        22    52
## 6    WORCESTER        44    52
## 7  SPRINGFIELD        29    48
## 8      BEVERLY        20    41
## 9     WINTHROP        13    39
## 10     WALTHAM        31    38
## # ... with 247 more rows
```

Not surprisingly, Boston has the highest number of accidents. Next, we may want to graph the findings in our data. And more intuitively, spatial visualization may be our best option. The following code will plot all the incidents and only the fatalities on two seperate graph.

```
library(ggmap)
```

```
## Warning: package 'ggmap' was built under R version 3.2.5
```

```
## Loading required package: ggplot2
```

```
## Warning: package 'ggplot2' was built under R version 3.2.5
```

```
##
## Attaching package: 'ggmap'
```

```
## The following object is masked from 'package:magrittr':
##
##     inset
```

```
library(ggplot2)

mapdata <- uniqueaccid[,c("ACTIVITYNO", "count","Fatality","SITESTATE","NAME")]

head(mapdata)
```

```
## # A tibble: 6 × 5
##   ACTIVITYNO count Fatality SITESTATE          NAME
##        <int> <int>    <dbl>     <chr>        <fctr>
## 1     141879     1        1        MA     BRIMFIELD
## 2     142349     1        1        MA     WORCESTER
## 3     142455     1        1        MA TURNERS FALLS
## 4     142737     1        1        MA     TEMPLETON
## 5     159020     1        1        MA     HOPKINTON
## 6     159319     1        1        MA      BROCKTON
```

```
mapdata$EXACT <- paste(mapdata$SITESTATE,mapdata$NAME)

test <- summarise(group_by(mapdata,EXACT), total = sum(count))

### this piece of code allows us to retrieve the longtitude and latitude of the cities. It takes around

#ma.location<- geocode(test$EXACT)
#write.table(ma.location, file = "MA_Location")

###

# in case we lost the data

x <- read.table("MA_Location")
head(x)
```

```
##         lon      lat
## 1 -70.94532 42.10482
## 2 -71.43284 42.48509
## 3 -70.89523 41.72237
## 4 -73.11743 42.62423
## 5 -72.61481 42.06954
## 6 -70.93004 42.85839
```

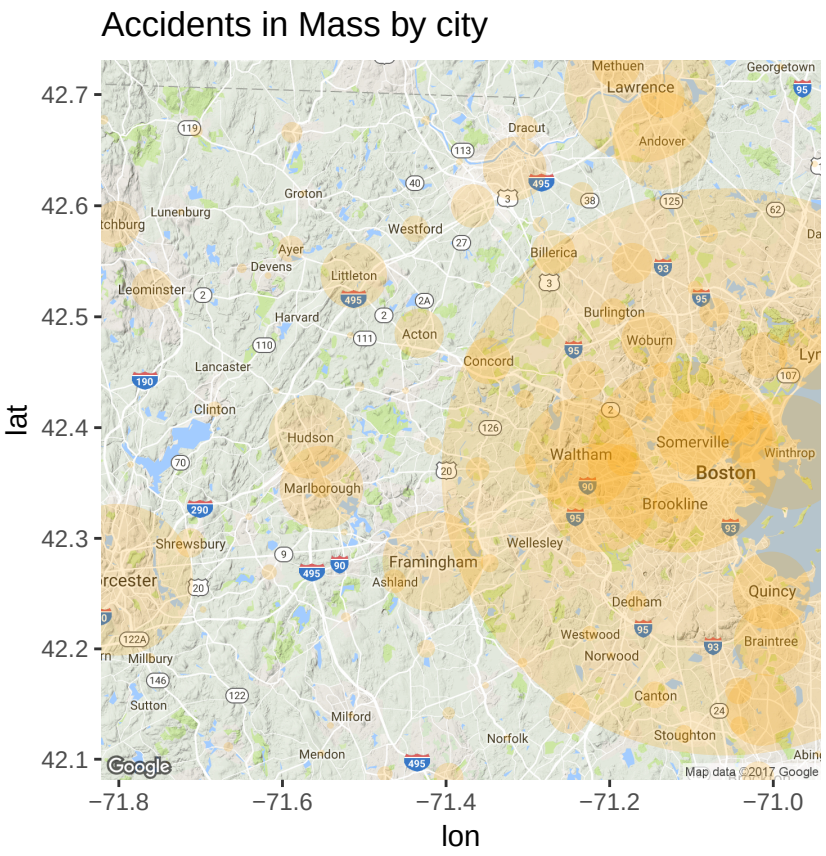
```
test$lon <- x$lon
test$lat <- x$lat


mass<- get_map("Masschusetts")
```

```
## Map from URL : http://maps.googleapis.com/maps/api/staticmap?center=Masschusetts&zoom=10&size=640x64
```

```
## Information from URL : http://maps.googleapis.com/maps/api/geocode/json?address=Masschusetts&sensor=
```

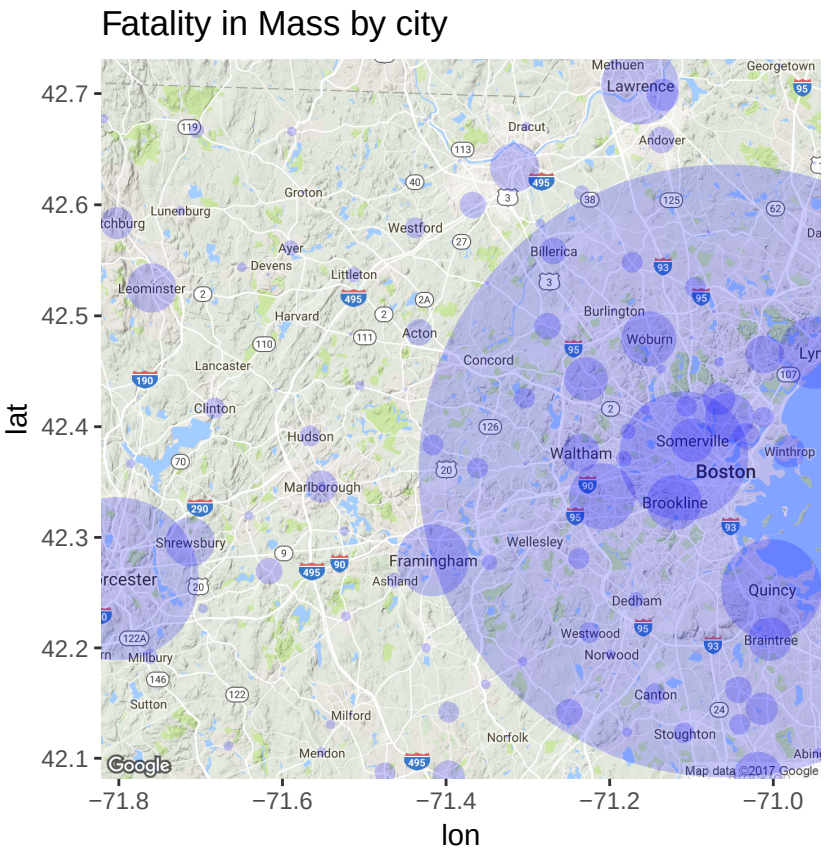
```
# rmarkdown has an inherent issue with graphs, and the size of the dot is much larger than it should be
#the graph more visually appealing
ggmap(mass) +geom_point(aes(x=lon, y=lat), data=test, col="orange", alpha=0.2, size=(test$total)/2) + g
```

```
## Warning: Removed 149 rows containing missing values (geom_point).
```



```
#lets only concern about fatality.

map_fatality <- summarise(group_by(mapdata,EXACT), total = sum(Fatality))

map_fatality$lon <- x$lon
map_fatality$lat <- x$lat

# fatality rate is much lower and ,therefore, the size can be keep as the same.
ggmap(mass) +geom_point(aes(x=lon, y=lat), data=map_fatality, col="blue", alpha=0.2, size=map_fatality$
```

```
## Warning: Removed 149 rows containing missing values (geom_point).
```



Now we know how exactly does the mapping work. Let's have some fun and graph the accidents in Boston at a street level.

```
#testing street level

testinfor <- uniqueaccid[,c( "ACTIVITYNO", "SITEADD", "SITESTATE", "NAME" )]

head(testinfor)
```

```
## # A tibble: 6 × 4
##   ACTIVITYNO        SITEADD SITESTATE          NAME
##        <int>         <fctr>     <chr>        <fctr>
## 1     141879  OFF MONSON RD        MA     BRIMFIELD
```

```
## 2     142349 25 SAGAMORE RD        MA     WORCESTER
## 3     142455   CANAL STREET        MA TURNERS FALLS
## 4     142737        MAIN ST        MA     TEMPLETON
## 5     159020     MESERVE ST        MA     HOPKINTON
## 6     159319   OAK HILL WAY        MA      BROCKTON
```

```
testinfor <- subset(testinfor, testinfor$NAME == "BOSTON")

head(testinfor)
```

```
## # A tibble: 6 × 4
##   ACTIVITYNO                        SITEADD SITESTATE   NAME
##        <int>                         <fctr>     <chr> <fctr>
## 1     922690        CAUSEWAY & LOMASNEY WAY        MA BOSTON
## 2     954883        CAUSEWAY & LOMASNEY WAY        MA BOSTON
## 3     954891 GALLIVAN, MORRISSEY, &NEPONSET        MA BOSTON
## 4    1803592  ST MARYS SCHOOL WARREN & PARK        MA BOSTON
## 5    1804889                  31 BRIMMER ST        MA BOSTON
## 6    1808666            260 FRANKLIN STREET        MA BOSTON
```

```
testinfor$EXACT <- paste(testinfor$SITEADD,testinfor$NAME)

head(testinfor)
```

```
## # A tibble: 6 × 5
##   ACTIVITYNO                        SITEADD SITESTATE   NAME
##        <int>                         <fctr>     <chr> <fctr>
## 1     922690        CAUSEWAY & LOMASNEY WAY        MA BOSTON
## 2     954883        CAUSEWAY & LOMASNEY WAY        MA BOSTON
## 3     954891 GALLIVAN, MORRISSEY, &NEPONSET        MA BOSTON
## 4    1803592  ST MARYS SCHOOL WARREN & PARK        MA BOSTON
## 5    1804889                  31 BRIMMER ST        MA BOSTON
## 6    1808666            260 FRANKLIN STREET        MA BOSTON
## # ... with 1 more variables: EXACT <chr>
```

```
testinfo <- summarise(group_by(testinfor,EXACT), count = n())

###  again this piece of code is for loading data from online and I save a table just to save the time.
#boston_location <- geocode(testinfo$EXACT)

#write.table(boston_location, file = "Boston_location")
###

bos <- read.table("Boston_location")

testinfo$lon <- bos$lon
testinfo$lat<- bos$lat

boston <- get_map("Boston", zoom =13)
```

```
## Map from URL : http://maps.googleapis.com/maps/api/staticmap?center=Boston&zoom=13&size=640x640&scal
```

```
## Information from URL : http://maps.googleapis.com/maps/api/geocode/json?address=Boston&sensor=false
```

```
ggmap(boston) + geom_point(aes(x = lon, y= lat), data = testinfo, col = "red" , alpha = .5 , size = tes
```

```
## Warning: Removed 50 rows containing missing values (geom_point).
```