# Midterm Mapping Report

Zhiwei Liang, Chi Zhang

11/10/2020

## 1.Brief Description

The Public Assistance Funded Projects Details dataset contains obligated (financial obligation to grantee) Public Assistance projects, lists public assistance recipients designated as applicants in the data, and a list of every funded, individual project, called project worksheets. Open projects still under pre-obligation processing are not represented.

The dataset is from OpenFEMA. In this project, we try to produce report, presentation, and web page with R. The fowllowing are all related results resource:

- Our Github Link
- Our Shiny App

## 2.Data Organization

In this project, we will focus on the hurricane,coastal storm during 2009-2018.

The following are the variables we need:

- `disasterNumber`: Sequentially assigned number used to designate an event or incident declared as a disaster.
- `declarationDate`: Date the disaster was declared.
- `incidentType`: Type of incident
- `projectAmount`: The estimated total cost of the Public Assistance grant project in dollars, without administrative costs.
- `totalObligated`: The federal share of the Public Assistance grant eligible project amount in dollars, plus grantee (State) and sub-grantee (applicant) administrative costs.
- `obligatedDate`: Date the grant was obligated.
- `dcc`: Damage category code. Category A: Debris removal; Category B: Emergency protective measures; Category C: Roads and bridges; Category D: Water control facilities; Category E: Public buildings and contents; Category F: Public utilities; Category G: Parks, recreational, and other facilities

- `projectSize`: Projects are designated as Large or Small.

### Filter & Select

We filtered Hurricane and storm from disaster types and chose time and variables in this part.

```
data <- read.csv('PublicAssistanceFundedProjectsDetails.csv')

#choose disaster type
df <- data%>%filter(incidentType %in% c("Hurricane","Coastal Storm"))
df <-na.omit(df)

#choose time
df$declarationDate <-  as_datetime(df$declarationDate)
df$declarationYear <-  year(df$declarationDate)
df$obligatedDate <- as_datetime(df$obligatedDate)
df$obligatedYear <- year(df$declarationDate)

#sum(ifelse(df$declarationYear==df$obligatedYear,0,1))==0
df <- df %>% filter(declarationYear > 2009 & declarationYear < 2018)

#choose variables
df <- df %>% select(-c("pwNumber","applicationTitle","applicantId","dam
ageCategoryCode","damageCategory","hash","lastRefresh","id"))

df$county %<>% tolower()
df$state %<>% tolower()

#write.csv(df,"hur_stm.csv",row.names=FALSE)
```

### Add fips,longitude and latitude

To better find the corresponding area more conveniently, we added Fips code, latitude and longitude. Also, we separate our data into statewide and countywide by looking at the countyCode is 0 or not.

```
###### Add Fips
df= read.csv("hur_stm.csv")
df$fips=NA
for (i in 1:dim(df)[1]){
  if(df$countyCode[i]> 99){
  df$fips[i] <- str_c(as.character(df$stateNumberCode[i]),as.character
(df$countyCode[i]))}
  else if(df$countyCode[i]>=10){
   df$fips[i] <- str_c(as.character(df$stateNumberCode[i]),"0",as.chara
cter(df$countyCode[i]))
  }
  else if(df$countyCode[i]<10){
   df$fips[i] <- str_c(as.character(df$stateNumberCode[i]),"0","0",as.c
```

```
haracter(df$countyCode[i]))
  }
}

#unique(df$stateNumberCode)
#there are 0 value for statewide disasters
#unique(df$countyCode)

df_statewide = df %>% filter(countyCode == 0)
df_countywide = df %>% filter(countyCode != 0)
#write.csv(df_countywide,"df_countywide.csv",row.names = FALSE)

###### GroupBy & Summarise
#df_countywide=read.csv("df_countywide.csv")
df_map=df_countywide %>%
  group_by(declarationYear,state,county,fips,incidentType,disasterNumbe
r)%>%
  summarise(projectAmount=sum(projectAmount),totalObligated=sum(totalOb
ligated),.groups="drop")%>%
   mutate(fips=as.numeric(fips))

MainStates <- map_data("state")
AllCounty <- map_data("county")
df_map_new = left_join(df_map,AllCounty, by=c("state"="region","county"
="subregion"))
#write.csv(df_map_new,"df_map_new.csv",row.names = FALSE)
# for shiny
```

## 3.Mapping

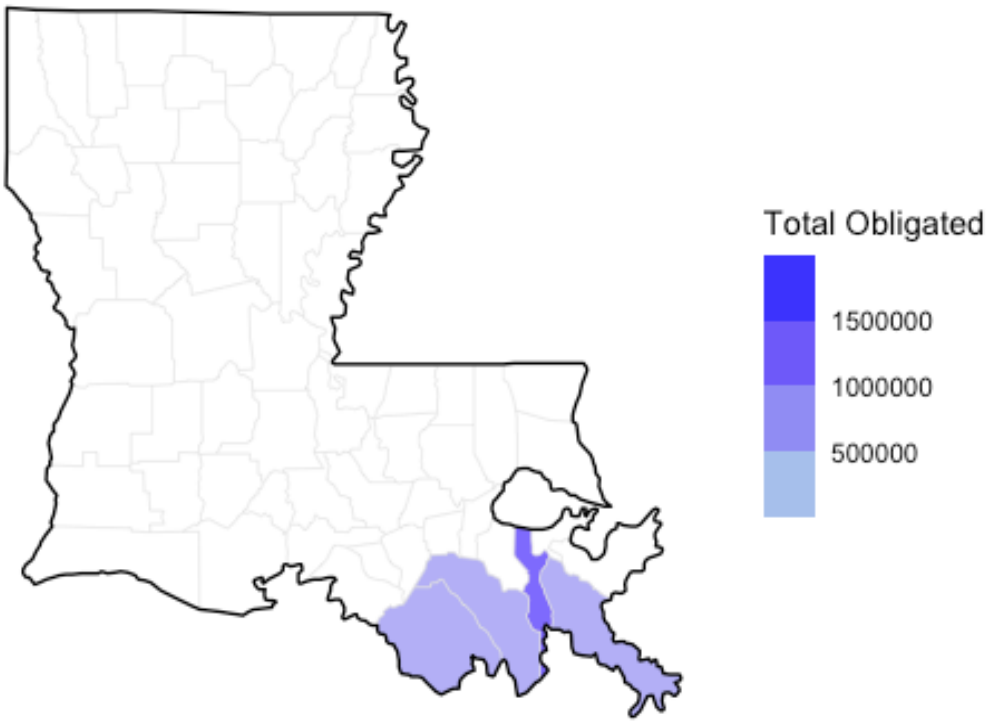
### Total Obligated

For this map, it shows the total obligated amount of whole country and counties by states. User can choose specific year, state and even disaster type to get different maps. There are two examples. The first one displays the obligated amount caused by Coastal Storm, in Louisiana,2011. The second one displays the total obligated amount of U.S caused by Hurricane in 2011.

```
obligate_map=function(myyear,mystate,mytype){

    h=df_map_new%>%filter(declarationYear==myyear,state==mystate,incide
ntType==mytype)
    s=sum(h$totalObligated)
    my_states <- map_data("state", region =mystate)
    my_counties <- map_data("county", region =mystate)


  ggplot() +
        geom_polygon(data=my_counties,
```

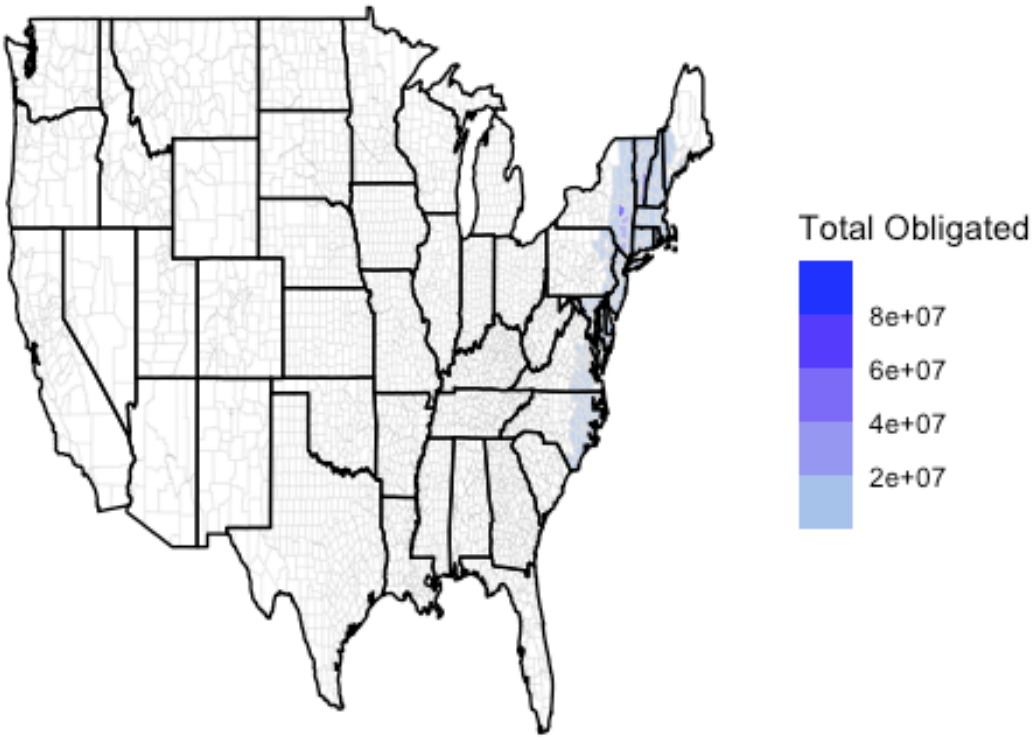
```
                    aes(x=long, y=lat, group=group),
                    color="gray", fill="white", size = .1 ) +
        geom_polygon(data = h,
                    aes(x = long, y = lat, group = group, fill = `tota
lObligated`),
                    color = "lightgrey", size = 0.2, alpha = 1.6,)+
        scale_fill_steps(low="lightblue",high="blue",name='Total Obliga
ted',guide = "coloursteps")+
        geom_polygon(data=my_states,
                    aes(x=long, y=lat, group=group),
                    color="black", fill="white",  size = .5, alpha = .
3)+
        ggtitle(paste(myyear,mystate,mytype,"total obligated: $",s))+
        theme(plot.title=element_text(hjust=0.5),
              panel.background=element_blank(),
              panel.border=element_blank(),
              axis.title=element_blank(),
              axis.text=element_blank(),
              axis.ticks=element_blank())
}

obligate_map(2011,"louisiana","Coastal Storm")
```

uisiana Coastal Storm total obligated: $ 345541951.2



```r
## for whole country
country_obligate_map=function(myyear,mytype){

    h=df_map_new%>%filter(declarationYear==myyear,incidentType==mytype)
    s=sum(h$totalObligated)
    ggplot() +
        geom_polygon(data=AllCounty,
                     aes(x=long, y=lat, group=group),
                     color="gray", fill="white", size = .1 ) +
        geom_polygon(data = h,
                     aes(x = long, y = lat, group = group, fill = `tota
lObligated`),
                     color = "lightgrey", size = 0.2, alpha = 1.6,)+
        scale_fill_steps(low="lightblue",high="blue",name='Total Obliga
ted',guide = "coloursteps")+
        geom_polygon(data=MainStates,
                     aes(x=long, y=lat, group=group),
                     color="black", fill="white",  size = .5, alpha = .
3)+
        ggtitle(paste(myyear,mytype,"total obligated: $",s))+
        theme(plot.title=element_text(hjust=0.5),
              panel.background=element_blank(),
              panel.border=element_blank(),
              axis.title=element_blank(),
              axis.text=element_blank(),
              axis.ticks=element_blank())
}

country_obligate_map(2011,"Hurricane")
```

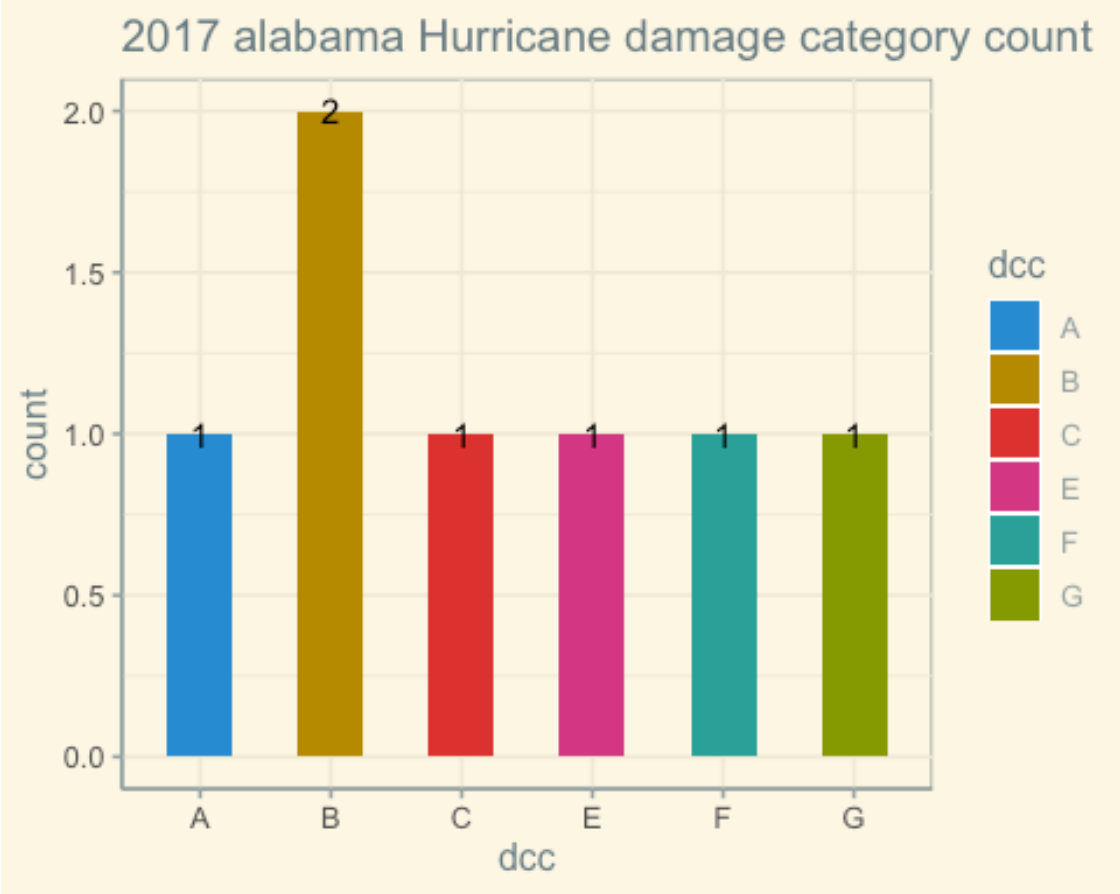
)11 Hurricane total obligated: $ 31681662421.9



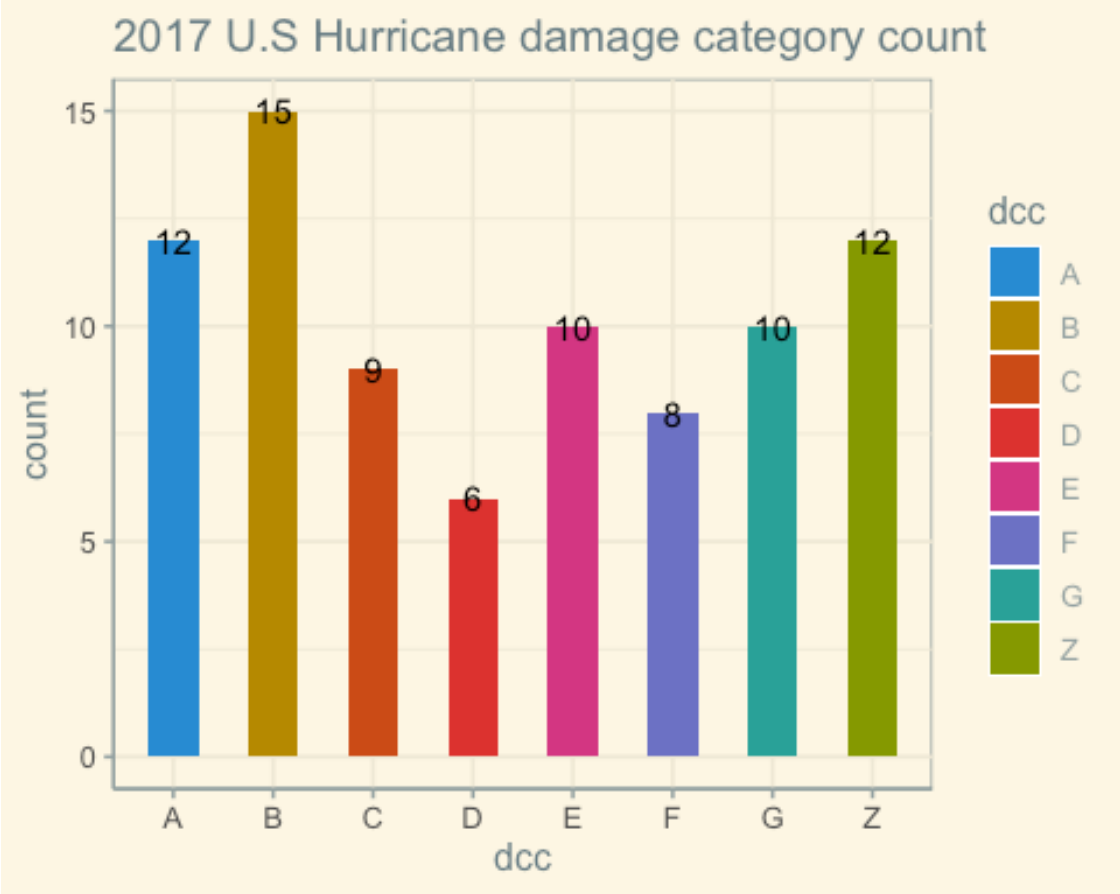

## barplot

For this bar plot, it counts damage categories of one disaster in a specific year. There are two examples. The first bar plot counts the categories of HUrricane damage in Alabama, 2017. The second bar plot counts the categories of Hurricane damage in U.S, 2017.

```
library(ggthemes)
dcc_bar = function(myyear,mystate,mytype){
  hh= df_countywide %>% filter(declarationYear==myyear,state==mystate,i
ncidentType==mytype)%>%
     group_by(disasterNumber)%>%
    summarise(dcc,.groups="drop")%>%
     unique()

   ggplot(hh,aes(x=dcc,fill=dcc))+
    geom_bar(stat = "count",width=0.5)+
    geom_text(aes(label=as.character(..count..)),stat="count")+
    theme_solarized()+
    scale_fill_solarized()+
    ggtitle(paste(myyear,mystate,mytype,"damage category count"))
}
dcc_bar(2017,"alabama","Hurricane")
```

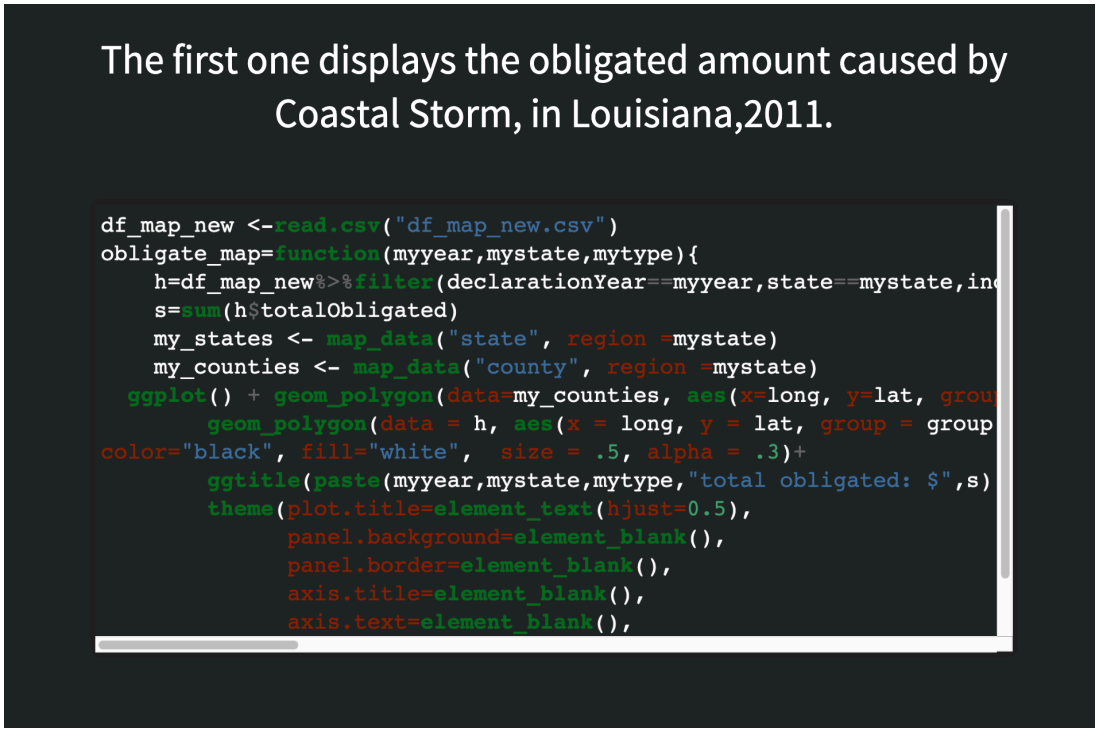



```
country_dcc_bar = function(myyear,mytype){
  hh= df_countywide %>%       filter(declarationYear==myyear,incidentTy
pe==mytype)%>%
     group_by(disasterNumber)%>%
    summarise(dcc,.groups="drop")%>%
     unique()

   ggplot(hh,aes(x=dcc,fill=dcc))+
    geom_bar(stat = "count",width=0.5)+
    geom_text(aes(label=as.character(..count..)),stat="count")+
    theme_solarized()+
    scale_fill_solarized()+
    ggtitle(paste(myyear,"U.S",mytype,"damage category count"))
}
country_dcc_bar(2017,"Hurricane")
```


2017 U.S Hurricane damage category count
count
15
10
5
0
12
15
9
6
10
8
10
12
A
B
C
D
E
F
G
Z
dcc
dcc
A
B
C
D
E
F
G
Z

## 4.Presentation

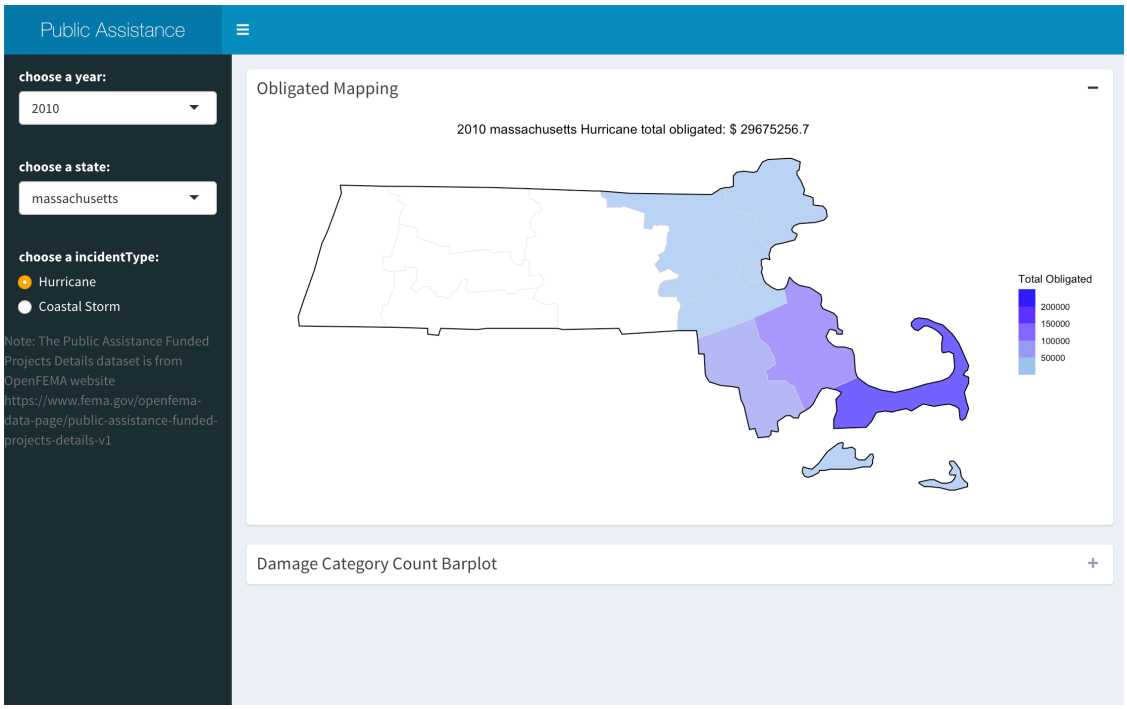
We make presentation with he revealjs package. It provides an output format **revealjs::revealjs_presentation**.



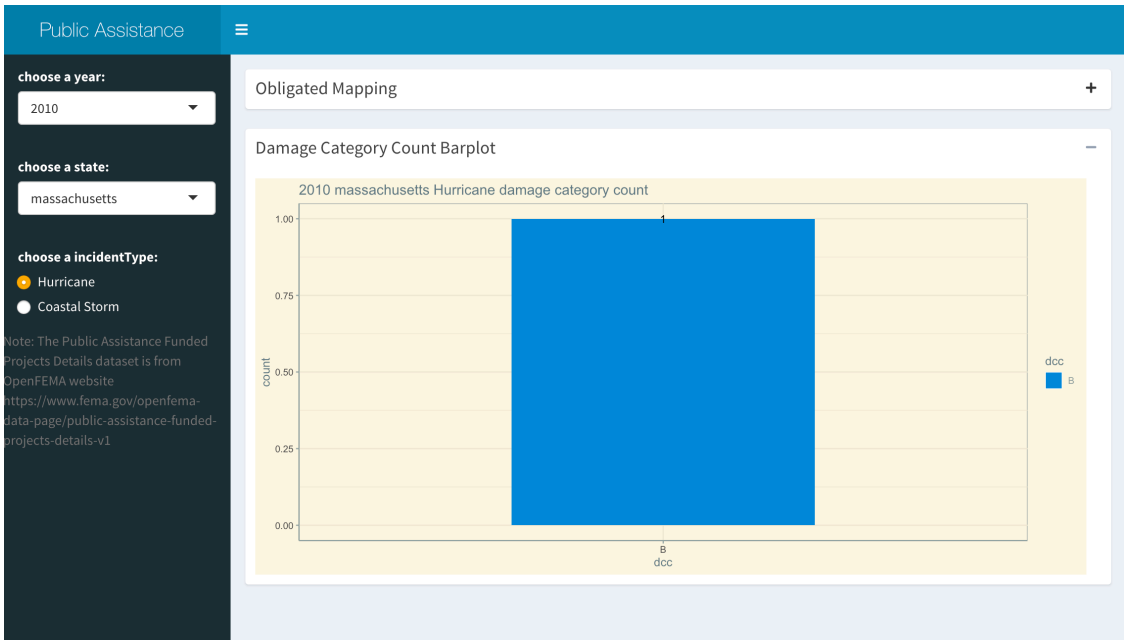

*Slides Interface Example*

## 5.Shiny app

After cleaning and mapping, we build an interactive web app by Shiny to present our data and plots more visually.
There are two view: the first one is *Total Obligated Mapping*, the second oone is *Damage Category Count Barplot*. Users can select the year, state and incident type




*Interface 1*




*Interface 2*

Here is the link to our app: Midterm Mapping Shiny.