Fire Explorer Report – Yu Huang

## Data Cleaning and Processsing

### Install Pakcage

I use function package.require to install and load packages. If the packages do not exist, it will be installed automatically.

#### package.require #####  
#### @param: packages install and load package ####  
package.require <- function(packages){  
 package.local <- as.data.frame(installed.packages()[,c("Package","Version")])  
 package.need.install <- setdiff(packages,  
 as.character(package.local$Package))  
 if(length(package.need.install) != 0){  
 install.packages(package.need.install)  
 }  
 for(package in packages){  
 require(package,character.only = TRUE)  
 }  
}  
  
package.require(c("dplyr", "foreign", "readr", "data.table", "lubridate"))

### Read Data

I build the file.read function to read two extension file dbf, and txt. - Read the codelookup

#### get all the file directory in fire ####  
path <- paste0(getwd(),"/",unlist(lapply(list.dirs("Fire"), function(x) paste0(x,"/",list.files(x)))))  
  
####### fire.read #########  
####### @param: path.dir the file path list or vector #########  
####### return: the list of data.table #########  
fire.read <- function(path.dir){  
 df.list <- list()  
 file.n <- length(path.dir)  
 year <- sapply(path.dir, function(x) unlist(strsplit(x,"/"))[4])  
 for(i in 1:file.n){  
 cat(paste0("File in ", path.dir[i]," is reading"),"\n")  
 # if the file extension is dbf, use the read.dbf to read file  
 if(grepl("dbf", tolower(path.dir[i])))  
 ds <- data.table(read.dbf(path.dir[i], as.is=TRUE))  
 else  
 # else use the readr to read file  
 ds <- read\_delim(path.dir[i], delim = "^", col\_names = TRUE)  
 colnames(ds) <- tolower(colnames(ds))  
 df.list[[i]] <- ds  
 }  
 # name list using year  
 names(df.list) <- year  
 gc()  
 return(df.list)  
}  
  
#### read all the codelookup, and add variable year ####  
codelookup <- fire.read(path[grepl("codelookup", path)])

codelookup <- lapply(2006:2015, function(x)  
 codelookup[[x-2005]] %>% mutate(year = x)) %>%  
 rbindlist  
  
#### get the description of inc\_type in the codelookup ####   
inc\_type.convert.df <- codelookup %>%  
 filter(fieldid == "INC\_TYPE") %>%  
 mutate(inc\_type = as.numeric(code\_value)) %>%  
 select(-fieldid, -code\_value)

#### get the abbreviation of the state  
state\_code <- codelookup %>%  
 filter(fieldid == "STATE" & !is.na(code\_value)) %>%  
 mutate(state = code\_value) %>%  
 select(state, code\_descr) %>%  
 unique

* Read the basicincident When we read the basicincident, we need do some transform, format all the key variable in suitable format and derive the day, weekday, month, year from the vairbale INC\_DATE. There are two format in INC\_DATE, at first, we transform it into character, if the length of INC\_DATE < 7, we should paste 0 0 in front of the INC\_DATE, then we use funciton lubridate::mdy function transorm it into date format.

#### key.transoform #####  
#### THe function to make the primary key unified format ####  
key.transform <- function(df){  
 df %>%  
 mutate(fdid = as.numeric(fdid)) %>%  
 mutate(inc\_date = as.numeric(inc\_date)) %>%  
 mutate(inc\_no = as.numeric(inc\_no)) %>%  
 mutate(exp\_no = as.numeric(exp\_no))  
}  
#### inc\_date.transfrom ####  
#### To derive the year, month, day, weekday from inc\_date ####  
inc\_date.transfrom <- function(df){  
 df %>%  
 mutate(inc\_date = mdy(ifelse(nchar(inc\_date) == 7, paste0(0,inc\_date),as.character(inc\_date)))) %>%  
 mutate(year = as.integer(year(inc\_date))) %>%  
 mutate(month = month(inc\_date,label = TRUE, abbr = TRUE)) %>%  
 mutate(day = day(inc\_date)) %>%  
 mutate(weekday = wday(inc\_date, label = TRUE, abbr = FALSE))  
}  
#### basicincident.stat #####  
#### calculate summarized statistics #####  
basicincident.stat <- function(df) {  
 df %>%  
 mutate(hh = as.integer(as.numeric(alarm)%%10^4/100)) %>%  
 group\_by(state, year, month, weekday, day, hh, inc\_type) %>%  
 summarise(n = n(),  
 ff\_death = sum(ff\_death, na.rm = TRUE),  
 oth\_death = sum(oth\_death, na.rm = TRUE),  
 ff\_inj = sum(ff\_inj, na.rm = TRUE),  
 oth\_inj = sum(oth\_inj, na.rm = TRUE))  
}  
  
  
basicinciden.list <- fire.read(path[grepl("basicincident", path)]) %>%  
 lapply(function(df) df %>%   
 select(state, fdid, inc\_date, inc\_no, exp\_no, inc\_type, alarm, ff\_death, oth\_death, ff\_inj, oth\_inj) %>% key.transform %>% inc\_date.transfrom  
 )

Fire.Stat <- basicinciden.list %>%  
 lapply(function(df) df %>% basicincident.stat) %>%  
 rbindlist %>%  
 mutate(inc\_type = as.numeric(inc\_type)) %>%  
 inner\_join(inc\_type.convert.df, by = c("inc\_type", "year"))

* Read teh hazmat information

#### read teh hazmat information ####  
hazmat <- fire.read(path[grepl("hazmat", path) & !grepl("hazmatequipinvolved", path)]) %>%  
 lapply(function(df) df %>% key.transform %>% inc\_date.transfrom)

for(i in 1:10){  
 hazmat[[i]] <- hazmat[[i]] %>%  
 inner\_join(basicinciden.list[[i]] %>%   
 select(state, fdid, inc\_date, inc\_no, exp\_no, inc\_type) %>%  
 mutate(inc\_type = as.numeric(inc\_type))) %>%   
 inner\_join(inc\_type.convert.df, by = c("inc\_type", "year"))  
}

* load the fifty state map

load("fifty\_states.RData")  
   
state.position <- fifty\_states %>%  
 group\_by(id) %>%  
 summarise(latitude = mean(lat), longitude = median(long))

## Statistic Analysis Fire

### Calcuate the Number of Fire in state by year.

state.df <- Fire.Stat %>%  
 group\_by(state, year) %>%  
 summarise(sum = sum(n)) %>%  
 inner\_join(state\_code) %>%  
 ungroup %>%  
 mutate(id = tolower(code\_descr),   
 state = code\_descr) %>%  
 inner\_join(state.position)

## Joining, by = "state"

## Joining, by = "id"

save(state.df, file = "state.df")

### Calculate the number of different fires with year

#### Calculate the number of occurrences of different fires per year #####  
Fire.year.stat <- Fire.Stat %>%  
 group\_by(year, inc\_type, code\_descr) %>%  
 summarise(n = n())  
#### save the result into .RData ####  
save(Fire.year.stat, file = "Fire.year.stat.RData")  
  
#### Calculate the number of occurrences of different fires per hour #####  
Fire.time.stat <- Fire.Stat %>%  
 group\_by(year, hh, inc\_type, code\_descr) %>%  
 summarise(n = n())  
#### save the result into .RData ####  
save(Fire.time.stat, file = "Fire.time.stat.RData")

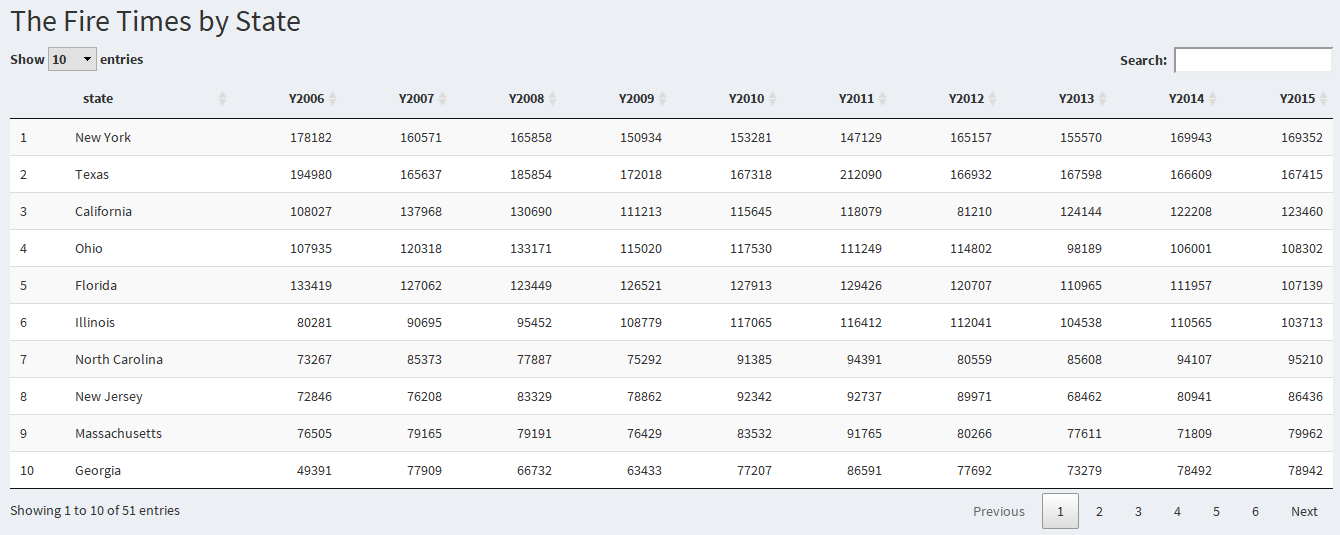
### Calculate the number of occurrences of person death in different types of fires per hour

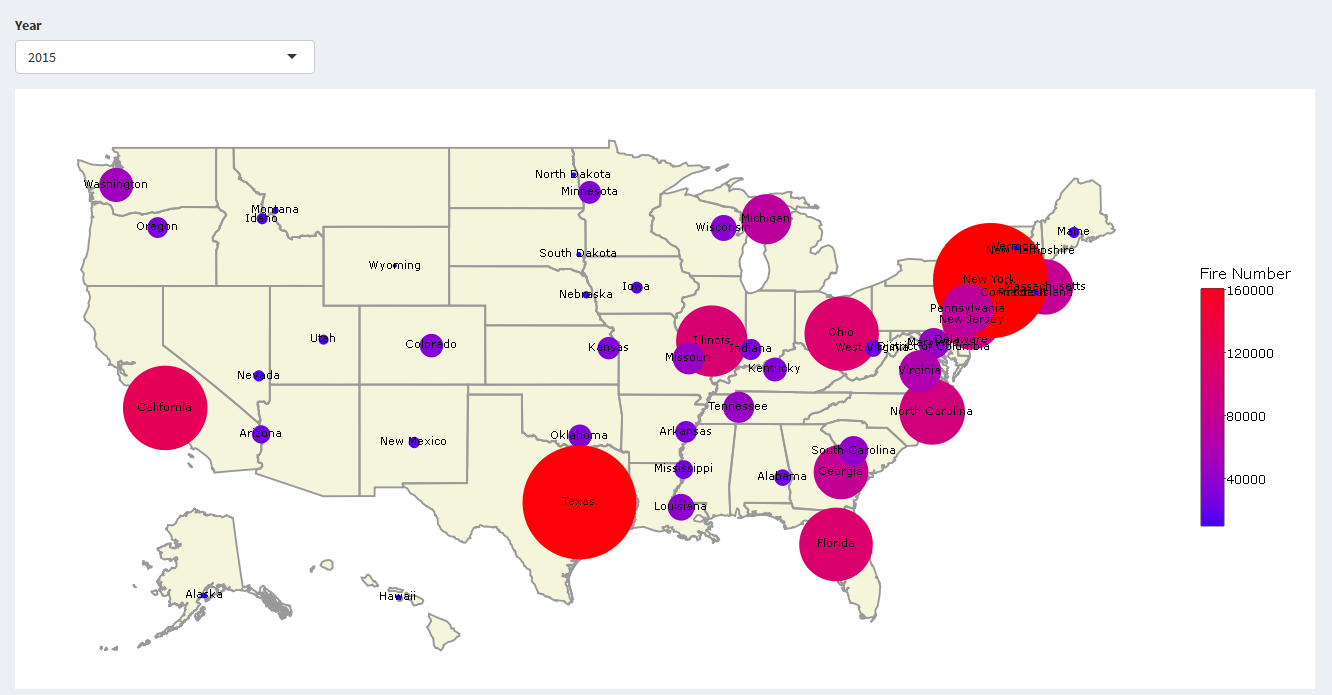
death.month.week.stat <- hazmat %>%  
 rbindlist %>%  
 group\_by(month, weekday, inc\_type, code\_descr) %>%  
 summarise(death.num = sum(haz\_death, na.rm = TRUE))

death.month.week.stat <- within(  
 death.month.week.stat,{   
 month <- factor(as.character(month), c("Jan", "Feb", "Mar", "Apr", "May", "Jun", "Jul", "Aug", "Sep", "Oct", "Nov", "Dec"))  
 weekday <- factor(as.character(weekday), c("Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday", "Sunday"))  
 }  
)  
  
save(death.month.week.stat, file = "death.month.week.stat.RData")  
  
death.fire <- hazmat %>%  
 rbindlist %>%  
 ungroup %>%  
 group\_by(year, inc\_type, code\_descr) %>%  
 summarise(death.num = sum(haz\_death, na.rm = TRUE))  
  
save(death.fire, file = "death.fire.RData")

## Analysis and Visualization Fire

### Accident State Analysis

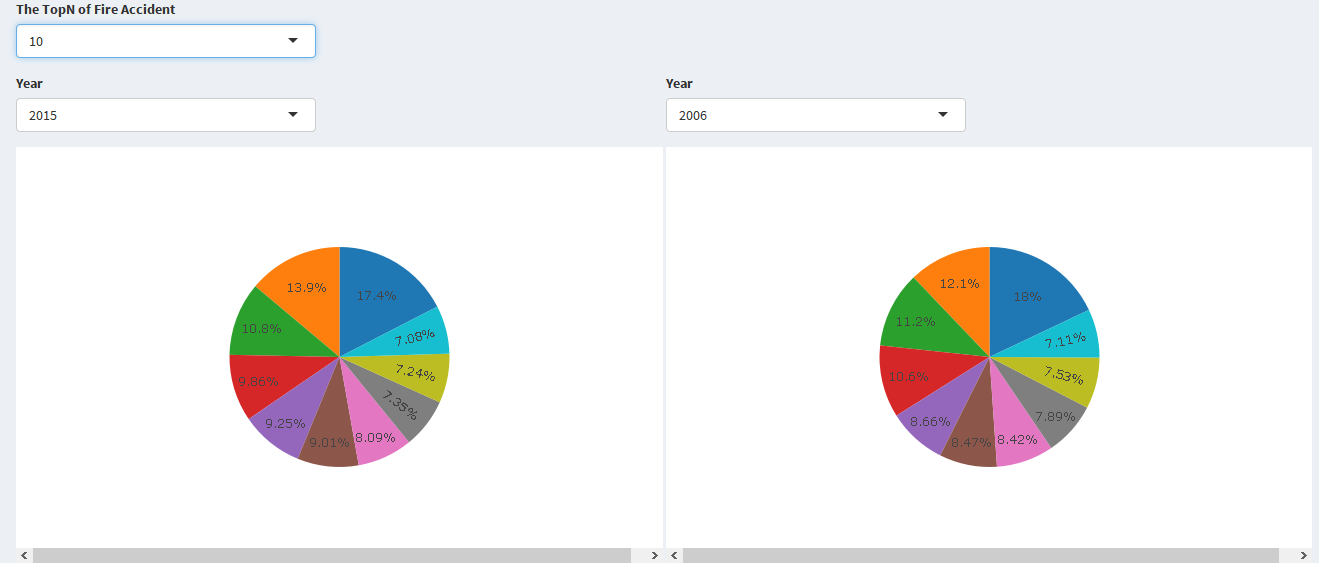




From the table and graphcics above, the five states where the most frequent fire occurred are Texas, New York, Florida, California, and Illionis. The rank for 2006 to 2015 is almost the same

### Fire Occurrence with Year

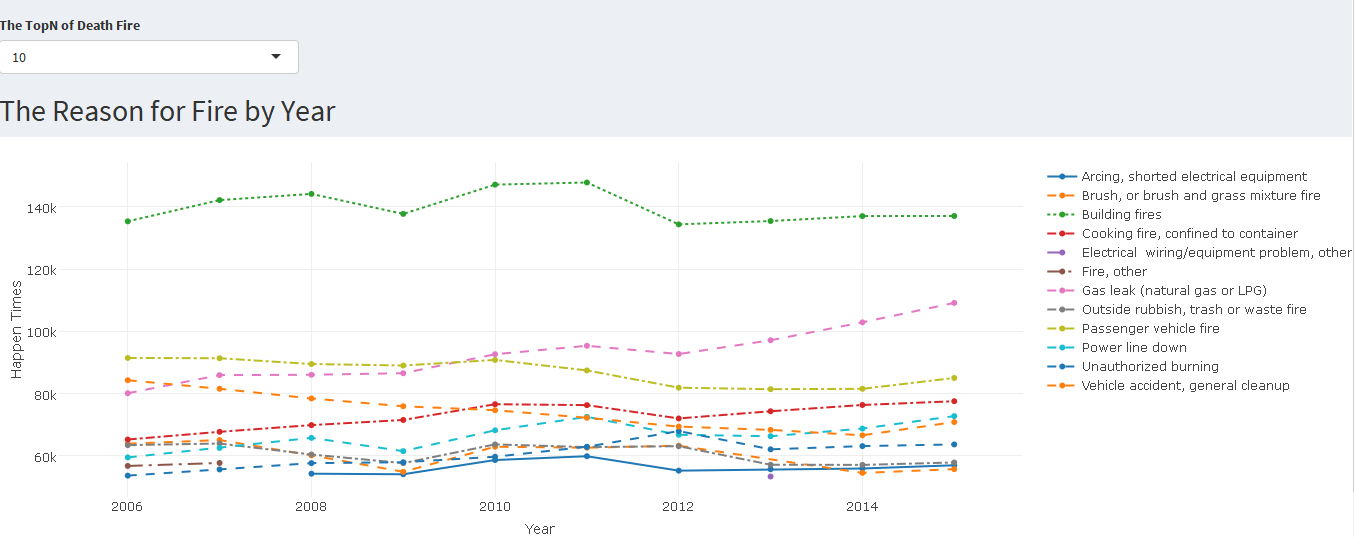
#### Top10 For All Year





As can be seen from the figure, the maximum number of fire occurred the top 10 reasons for the fire, compared to 2015 and 2014. The first reason is the Building fires, reduced by 0.35% points, the Gas (natural gas or LPG) increase very much, from 10.63% to 13.88. Similarly, the power line down also increase very much, from 7.89% to 9.25%.

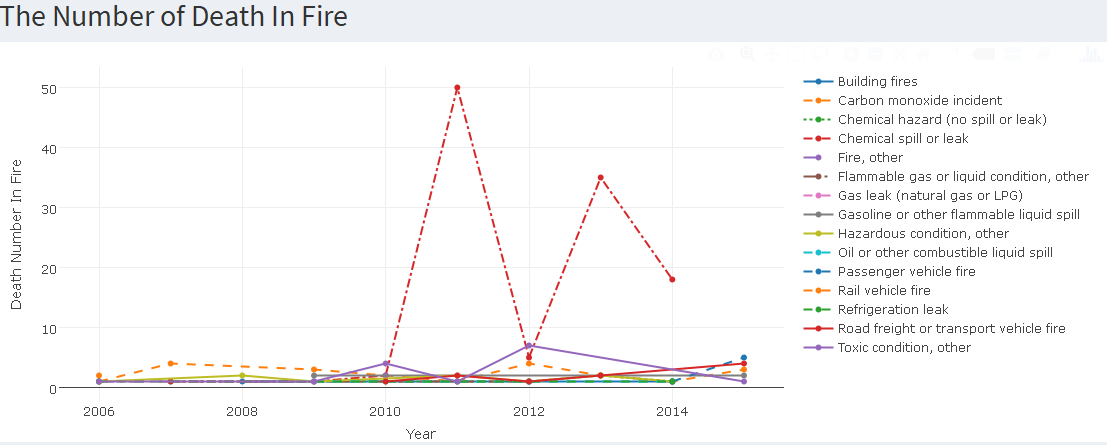
#### Top10 For Every Year

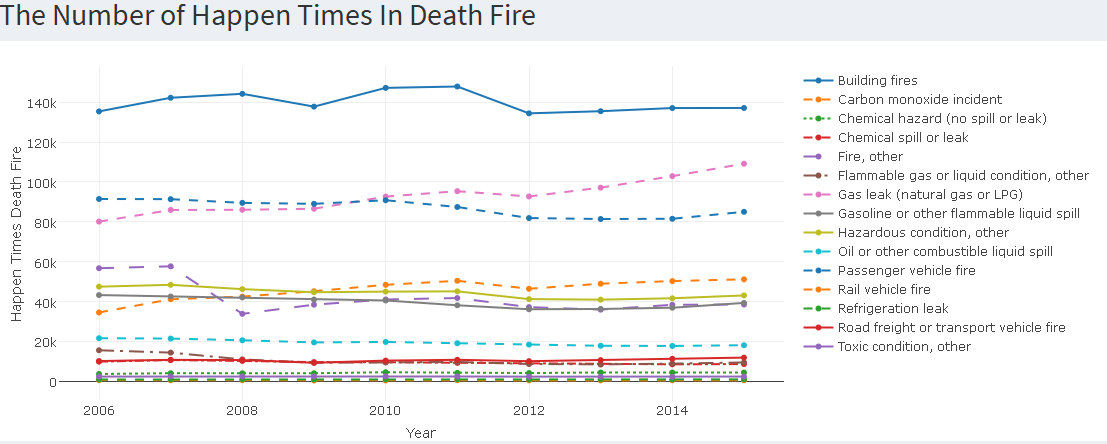




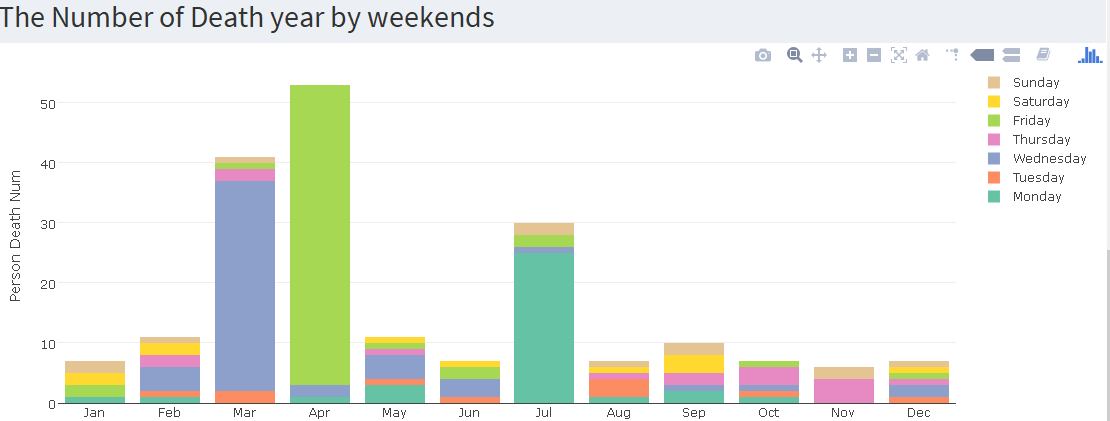
For every year, we choose top10 fire reason, the result show, there are 12 fire reasons, [Arcing, shorted electrical equipment], [Brush, or brush and grass mixture fire], [Building fires], [Cooking fire, conofined to container], [Electrical wiring/equipment problem, other], [Fire, other], [Gas leak(natural gas or LPG)], [Outside rubbish], [trash or waste fire], [Passenger vehicle fire], and [Power line down]. The fire times of building fire is the most than the other.

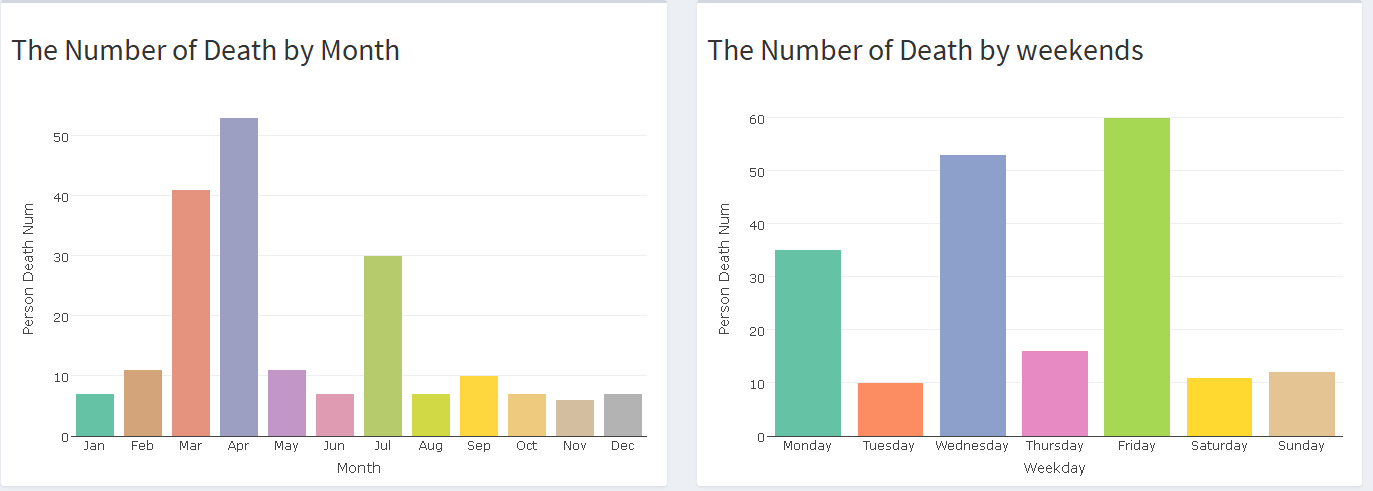
### Fire Death Analysis





From the number of death in fire, we can know that the chemical spill or leak is the high probability of death, with less times occupancy. It is worthy that the number of deaths in 2011 was 50.





From the number of death in month by week, it’s known that high mortality rates were on Friday in April.

## Analysis Conclusion

The five states where the most frequent fire occurred are Texas, New York, Florida, California, and Illinois. The rank for 2006 to 2015 is almost the same.

From the number of death in fire, we can know that the chemical spill or leak is the high probability of death, with less times occupancy. It is worthy that the number of deaths in 2011 was 50.

## References

Pedersen, Thomas Lin. 2016. Ggraph: An Implementation of Grammar of Graphics for Graphs and Networks.

Allaire, JJ. 2016. Flexdashboard: R Markdown Format for Flexible Dashboards. [https://CRAN.R-project.org/package=flexdashboard](https://cran.r-project.org/package=flexdashboard).

Attali, Dean. 2016. Colourpicker: A Colour Picker Widget for Shiny Apps, Rstudio, R-Markdown, and ’Htmlwidgets’. [https://CRAN.R-project.org/package=colourpicker](https://cran.r-project.org/package=colourpicker).

Cheng, Joe. 2016. Crosstalk: Inter-Widget Interactivity for Html Widgets.

Contributors, Brian Reavis &. 2016. “Selectize Is an Extensible jQuery-Based Custom <select> Ui Control.” <https://github.com/selectize/selectize.js>.

Bache, Stefan Milton, and Hadley Wickham. 2014. Magrittr: A Forward-Pipe Operator for R. [https://CRAN.R-project.org/package=magrittr](https://cran.r-project.org/package=magrittr).

## ShinyApp Code

*# Simple header -----------------------------------------------------------*  
**library**(shiny)  
**library**(shinydashboard)

**library**(plotly)

**library**(tidyr)  
**library**(ggplot2)  
**library**(dplyr)

**library**(plyr)

**library**(data.table)

**library**(DT)

*# Sidebar --------------------------------------------------------------*  
  
sidebar <- ## Sidebar content  
 **dashboardSidebar**(  
 **sidebarMenu**(  
 **menuItem**(  
 "Accident State Analysis",  
 tabName = "State",  
 icon = **icon**("dashboard")  
 ),  
 **menuItem**(  
 "Fire Occurrence Analysis",  
 tabName = "HappenTimes",  
 icon = **icon**("th")  
 ),  
 **menuItem**(  
 "Fire Reason Analysis",  
 tabName = "FireType",  
 icon = **icon**("dashboard")  
 ),  
 **menuItem**(  
 "Fire Death Analysis",  
 tabName = "Death",  
 icon = **icon**("dashboard")  
 )  
 )  
 )  
  
*# Compose dashboard body --------------------------------------------------*  
  
body <- **dashboardBody**(  
 **tabItems**(  
   
 *# First tab content*  
 *# Boxes need to be put in a row (or column)*  
 **tabItem**(  
 tabName = "HappenTimes",  
 **fluidPage**(  
 **selectInput**("TopN", "The TopN of Fire Accident", choices = **c**(10, 20, 30, 40)),  
 **splitLayout**(  
 **verticalLayout**(**selectInput**("Year1", "Year", choices = **c**(2015:2006)),  
 **plotlyOutput**("p1")),  
 **verticalLayout**(**selectInput**("Year2", "Year", choices = **c**(2006:2015)),  
 **plotlyOutput**("p2"))  
 )  
 ),  
 **fluidRow**(  
 **splitLayout**(  
 **verticalLayout**(  
 **tableOutput**('Year1'))  
 ,  
 **verticalLayout**(  
 **tableOutput**('Year2')  
 )  
 )) *# End of fluidRow*  
 ), *# End of tabItem*  
   
 *# Second tab content*  
 *# Boxes need to be put in a row (or column)*  
 **tabItem**(  
 tabName = "FireType",  
 **fluidRow**(  
 **selectInput**("FireTopN", "The TopN of Death Fire", choices = **c**(10, 20, 30, 40)),  
 **h2**("The Reason for Fire by Year"),  
 **plotlyOutput**("p3"),  
 *# tableOutput('dt1')*  
 DT::**dataTableOutput**('dt1')  
 *# End of tabBox*  
 ) *# End of fluidRow,*  
 ), *# End of tabItem*  
   
 *# Third tab content*  
 *# Boxes need to be put in a row (or column)*  
 **tabItem**(  
 tabName = "Death",  
 **fluidRow**(**h2**("The Number of Death In Fire"), status = "info",  
 **plotlyOutput**("p5\_1")),  
 **fluidRow**(**h2**("The Number of Happen Times In Death Fire"), status = "info",  
 **plotlyOutput**("p5\_2")),  
 **fluidRow**(**h2**("The Number of Death year by weekends"), status = "info",  
 **plotlyOutput**("p6")),  
 **fluidRow**(  
 **box**(  
 **h2**("The Number of Death by Month"),  
 **plotlyOutput**("p7")  
 ),  
 **box**(  
 **h2**("The Number of Death by weekends"),  
 **plotlyOutput**("p8")  
 )  
 ) *# End of fluidRow*  
 ), *# End of tabItem*  
   
 *# Fourth tab content*  
 *# Boxes need to be put in a row (or column)*  
 **tabItem**(  
 tabName = "State",  
 **h2**("The Fire Times by State"),  
 DT::**dataTableOutput**('dt2'),  
 **selectInput**("Year\_map\_choose", "Year", choices = **c**(2015:2006)),  
 **plotlyOutput**("Year\_map", height = 600, width = 1300)  
 )  
 ) ## End tabItems  
) ## End dashboardBody  
  
  
  
  
*# Setup Shiny app UI components -------------------------------------------*  
  
ui <- **dashboardPage**(header, sidebar, body, skin = "purple")  
  
*# Setup Shiny app back-end components -------------------------------------*  
  
server <- function(input, output) {  
 **source**("load.R")  
 formulaText1 <- **reactive**({  
 **paste0**("The Reason for Fire in ", input$Year1)  
 })  
   
 formulaText2 <- **reactive**({  
 **paste0**("The Reason for Fire in ", input$Year2)  
 })  
  
 *# Return the formula text for printing as a caption*  
 output$caption1 <- **renderText**({  
 **formulaText1**()  
 })  
 output$caption2 <- **renderText**({  
 **formulaText2**()  
 })  
   
 output$Year1 <- **renderTable**({  
 Fire.year.stat %>%  
 **filter**(year == input$Year1) %>%  
 **arrange**(**desc**(n)) %>%  
 **head**(**as.numeric**(input$TopN)) %>%  
 ungroup %>%  
 **select**(year, code\_descr, n) %>%  
 **mutate**(Percent = **paste0**(**round**(n / **sum**(n)\*100,2),"%")) %>%  
 **rename**(**c**("code\_descr" = "Fire Accident", "n" = "times"))  
 })  
 output$Year2 <- **renderTable**({  
 Fire.year.stat %>%  
 **filter**(year == input$Year2) %>%  
 **arrange**(**desc**(n)) %>%  
 **head**(**as.numeric**(input$TopN)) %>%  
 ungroup %>%  
 **select**(year, code\_descr, n) %>%  
 **mutate**(Percent = **paste0**(**round**(n / **sum**(n)\*100,2),"%")) %>%  
 **rename**(**c**("code\_descr" = "Fire Accident", "n" = "times"))  
 })  
 output$Year\_map <- **renderPlotly**({  
 year.state.df <- state.df %>%  
 **filter**(year == input$Year\_map\_choose) %>%  
 **mutate**(sum = **ifelse**(**is.na**(sum), 0, sum)) %>%  
 ungroup  
 p <- year.state.df %>%  
 **ggplot**() +   
 **geom\_polygon**(**aes**(x=long, y=lat, group=group), data=fifty\_states, fill="beige", colour="grey60")  
 p <- p + **geom\_point**(data=year.state.df,**aes**(colour = sum, x = longitude, y= latitude),  
 size=30\*year.state.df$sum/**max**(year.state.df$sum)) +  
 **geom\_text**(**aes**(label=state,x=longitude,y=latitude),size=3) +  
 **scale\_colour\_gradient**("Fire Number",high="red",low="blue") +  
 **theme**(  
 panel.grid = **element\_blank**(),  
 panel.background = **element\_blank**(),  
 axis.text = **element\_blank**(),  
 axis.ticks = **element\_blank**(),  
 axis.title = **element\_blank**()  
 )  
 **ggplotly**(p)  
 })  
 output$dt2 <- DT::**renderDataTable**({  
 state.df %>%  
 **select**(-code\_descr, -id, -latitude, -longitude) %>%  
 **mutate**(year = **paste0**("Y", year)) %>%  
 **spread**(year, sum) %>%  
 data.frame %>%  
 **arrange**(**desc**(Y2015))  
 })  
 output$p1 <- **renderPlotly**({  
 Fire.year.stat %>%  
 **filter**(year == input$Year1) %>%  
 **arrange**(**desc**(n)) %>%  
 **head**(input$TopN) %>%  
 **plot\_ly**(type="pie", labels = ~code\_descr,  
 values = ~n, textinfo="percent",  
 showlegend=F)  
 })  
 output$p2 <- **renderPlotly**({  
 Fire.year.stat %>%  
 **filter**(year == input$Year2) %>%  
 **arrange**(**desc**(n)) %>%  
 **head**(input$TopN) %>%  
 **plot\_ly**(type="pie", labels = ~code\_descr,  
 values = ~n, textinfo="percent",  
 showlegend=F)  
 })  
   
 output$p3 <- **renderPlotly**({  
 Fire.year.stat %>%  
 **group\_by**(year) %>%  
 **top\_n**(**as.numeric**(input$FireTopN)) %>%  
 **plot\_ly**(x= ~year, y = ~n,linetype = ~code\_descr) %>%  
 **layout**(mode = "markers", xaxis = **list**(title = "Year"), yaxis = **list**(title = "Happen Times"))  
 })  
 output$dt1 <- DT::**renderDataTable**({  
 Fire.year.stat %>%  
 **group\_by**(year) %>%  
 **top\_n**(**as.numeric**(input$FireTopN)) %>%  
 **select**(-inc\_type) %>%  
 **rename**(**c**("code\_descr" = "Fire Accident")) %>%  
 **spread**(year, n)   
 })  
 output$p5\_1 <- **renderPlotly**({  
 Fire.year.stat %>%  
 **inner\_join**(death.fire) %>%  
 **plot\_ly**(x= ~year, y = ~death.num,linetype = ~code\_descr) %>%  
 **layout**(mode = "markers", xaxis = **list**(title = "Year"), yaxis = **list**(title = "Death Number In Fire"))  
 })  
 output$p5\_2 <- **renderPlotly**({  
 Fire.year.stat %>%  
 **inner\_join**(death.fire %>% ungroup %>%  
 **select**(inc\_type) %>% unique) %>%  
 **plot\_ly**(x= ~year, y = ~n,linetype = ~code\_descr) %>%  
 **layout**(mode = "markers", xaxis = **list**(title = "Year"), yaxis = **list**(title = "Happen Times Death Fire"))  
 })  
   
 output$p6 <- **renderPlotly**({  
 death.month.week.stat %>%  
 **plot\_ly**(x= ~month, y = ~death.num, color = ~weekday) %>%  
 **add\_bars**() %>%  
 **layout**(barmode = "stack", xaxis = **list**(title = "Month"), yaxis = **list**(title = "Person Death Num"))  
 })  
 output$p7 <- **renderPlotly**({  
 death.month.week.stat %>%  
 **plot\_ly**(x= ~month, y = ~death.num, color = ~month) %>%  
 **add\_bars**() %>%  
 **layout**(barmode = "stack",showlegend = FALSE, xaxis = **list**(title = "Month"), yaxis = **list**(title = "Person Death Num"))  
 })  
 output$p8 <- **renderPlotly**({  
 death.month.week.stat %>%  
 **plot\_ly**(x= ~weekday, y = ~death.num, color = ~weekday) %>%  
 **add\_bars**() %>%  
 **layout**(barmode = "stack",showlegend = FALSE, xaxis = **list**(title = "Weekday"), yaxis = **list**(title = "Person Death Num"))  
 })  
}