Appendix 1 - Code used to collect data

Group 5

Table of Contents

# Collection of Property Data

Data has been collection from the website Domain ( <https://www.domain.com.au/> ).

### The Process

##### Introduction

We will get the sold property data from the website Domain ( <https://www.domain.com.au/> ). We target to get all the sold property data in NSW started from 2018

#### Download the property list html file for each subsurb

Import the libraries

library(tidyverse)  
library(parallel)  
library(rvest)  
library(plyr)  
library(ggmap)  
library(jsonlite)

Initalize the variables

PROPERTY\_BASE\_URL\_FORMAT <- "https://www.domain.com.au/sold-listings/%s-nsw-%s/?excludepricewithheld=1&ssubs=0&page=%s"  
no\_of\_cores <- detectCores() - 1  
# create directory  
dir.create(file.path("property\_web\_scraping", "property\_listing\_html"))

#### Read the suburb information

postcode\_list <- read\_csv(file.path("..", "property\_web\_scraping", "postcodes\_geo\_NSW.csv"))  
filtered\_postcode\_list <- postcode\_list %>% filter(postcode\_list$postcode >= 2000)  
str(filtered\_postcode\_list)

## tibble [4,714 × 6] (S3: spec\_tbl\_df/tbl\_df/tbl/data.frame)  
## $ X1 : num [1:4714] 262 263 264 265 266 267 268 269 270 271 ...  
## $ postcode : num [1:4714] 2000 2000 2000 2000 2000 ...  
## $ suburb : chr [1:4714] "Barangaroo" "Dawes Point" "Haymarket" "Millers Point" ...  
## $ state : chr [1:4714] "NSW" "NSW" "NSW" "NSW" ...  
## $ latitude : num [1:4714] -33.9 -33.9 -33.9 -33.9 -33.9 ...  
## $ longitude: num [1:4714] 151 151 151 151 151 ...  
## - attr(\*, "spec")=  
## .. cols(  
## .. X1 = col\_double(),  
## .. postcode = col\_double(),  
## .. suburb = col\_character(),  
## .. state = col\_character(),  
## .. latitude = col\_double(),  
## .. longitude = col\_double()  
## .. )

Function download\_property\_listing\_html\_files will download the property list html file to local machine by constructing url from suburb name and postcode

################## download list html files ##################  
download\_property\_listing\_html\_files <- function(row\_index) {  
 out <- tryCatch({  
   
 page = 1  
   
 while (TRUE) {  
   
 postcode\_data <- filtered\_postcode\_list[row\_index, ]  
 suburb <- str\_replace\_all(tolower(postcode\_data$suburb), " ", "-")  
 property\_postcode\_website\_url <- sprintf(PROPERTY\_BASE\_URL\_FORMAT, suburb, postcode\_data$postcode, page)  
   
 # construct the file path and name  
 filename <- sprintf("property\_link\_%s\_%s\_%s.html", suburb, postcode\_data$postcode, page)  
 downloaded\_file\_path <- file.path("property\_web\_scraping", "property\_listing\_html", filename)  
   
 download.file(url = property\_postcode\_website\_url, destfile = downloaded\_file\_path)   
   
 # read the downloaded file and grep the last sold year in the page  
 html\_data <- read\_html(downloaded\_file\_path)  
 sold\_date <- html\_data %>% html\_nodes(xpath = '//span[@class="css-1nj9ymt"]') %>% html\_text()  
 sold\_year <- as.integer(substrRight(sold\_date[length(sold\_date)], 4))  
   
 # if last item is older than 2018, stop looping  
 if (sold\_year>=2018) {  
 page = page + 1  
 } else {  
 break;  
 }  
   
 }  
   
 },  
 error = function(cond) {  
 writeLines(sprintf("Error at index : %d, error : %s", row\_index, cond))  
 return(NA)  
 })  
}

Run the function download\_property\_listing\_html\_files in parallel to speed up the process

no\_of\_data <-nrow(filtered\_postcode\_list)  
remain\_property\_data <- c(1:no\_of\_data)  
# download html files in parallel  
results <- mclapply(remain\_property\_data, download\_property\_listing\_html\_files, mc.cores = no\_of\_cores)

#### Scraping the property data

Function scrape\_data\_from\_list\_file will scrap the proeprty data for each suburb form the HTML file

scrape\_data\_from\_list\_file <- function(row\_index) {  
   
 property\_data <- data.frame("date" = as.Date(character()),  
 "price" = integer(),  
 "address" = character(),  
 "suburb" = character(),  
 "postcode" = integer(),  
 "url" = character(),  
 "no\_of\_bed" = integer(),   
 "no\_of\_bath" = integer(),  
 "no\_of\_parking" = integer(),  
 "house\_size" = integer(),  
 "type" = character()  
 )  
   
 out <- tryCatch({  
   
 page = 1  
   
 while (TRUE) {  
   
 postcode\_data <- filtered\_postcode\_list[row\_index, ]  
   
 # construct the filename and read the html file  
 suburb <- str\_replace\_all(tolower(postcode\_data$suburb), " ", "-")  
 filename <- sprintf("property\_link\_%s\_%s\_%s.html", suburb, postcode\_data$postcode, page)  
 list\_file\_path <- file.path("property\_web\_scraping", "property\_listing\_html", filename)  
 html\_data <- read\_html(list\_file\_path)  
   
 writeLines(sprintf("Process file : %s", list\_file\_path))  
   
 # get the last property sold year  
 all\_sold\_dates <- html\_data %>% html\_nodes(xpath = '//span[@class="css-1nj9ymt"]') %>% html\_text()  
 last\_property\_sold\_year <- as.integer(substrRight(all\_sold\_dates[length(all\_sold\_dates)], 4))  
   
 property\_card\_nodes <- html\_data %>% html\_nodes(xpath = '//div[@class="css-1kk6519"]')  
   
 for (card\_index in 1:length(property\_card\_nodes)) {  
   
 # initial variables  
 property\_sold\_date <- NA  
 property\_price <- NA  
 property\_address <- NA  
 property\_link <- NA  
 no\_of\_bed <- NA  
 no\_of\_bath <- NA  
 no\_of\_parking <- NA  
 property\_type <- NA  
 house\_size <- NA  
   
 property\_card\_node <- property\_card\_nodes[card\_index]  
   
 # get property sold date  
 property\_sold\_date <- property\_card\_node %>% html\_nodes(xpath = './/span[@class="css-1nj9ymt"]') %>% html\_text()  
 property\_sold\_date <- substrRight(property\_sold\_date, 11)  
 property\_sold\_date <- as.Date(property\_sold\_date, format="%d %b %Y")  
 property\_sold\_date <- as.character(property\_sold\_date)  
   
 # get property sold price  
 property\_price <- property\_card\_node %>% html\_nodes(xpath = './/p[@data-testid="listing-card-price"]') %>% html\_text()  
 property\_price <- str\_split\_fixed(property\_price, " ", n = Inf)[1]  
 property\_price <- str\_remove\_all(property\_price, ",")  
 property\_price <- sub(".", "", property\_price)  
   
 # get property address  
 property\_address <- property\_card\_node %>% html\_nodes(xpath = './/meta') %>% html\_attr("content")  
   
 # get property link  
 property\_link <- property\_card\_node %>% html\_nodes(xpath = './/link') %>% html\_attr("href")  
   
 # get property bed, bath, parking info  
 property\_features <- property\_card\_node %>% html\_nodes(xpath = './/span[@data-testid="property-features-text-container"]') %>% html\_text()  
 property\_features  
 for (property\_feature in property\_features) {  
 if (grepl("Bed", property\_feature, fixed=TRUE)) {  
 no\_of\_bed <- as.integer(str\_replace(property\_feature, " .\*", ""))  
 } else if (grepl("Bath", property\_feature, fixed=TRUE)) {  
 no\_of\_bath <- as.integer(str\_replace(property\_feature, " .\*", ""))  
 } else if (grepl("Parking", property\_feature, fixed=TRUE)) {  
 no\_of\_parking <- as.integer(str\_replace(property\_feature, " .\*", ""))  
 } else if (grepl("m²", property\_feature, fixed=TRUE)) {  
 house\_size <- as.integer(str\_replace(property\_feature, "m² .\*", ""))  
 }  
 }  
   
 # get property type  
 property\_type <- property\_card\_node %>% html\_nodes(xpath = './/span[@class="css-693528"]') %>% html\_text()  
   
   
 property <- data.frame("date" = property\_sold\_date,  
 "price" = property\_price,  
 "address" = property\_address,  
 "suburb" = suburb,  
 "postcode" = postcode\_data$postcode,  
 "url" = property\_link,  
 "no\_of\_bed" = no\_of\_bed,  
 "no\_of\_bath" = no\_of\_bath,  
 "no\_of\_parking" = no\_of\_parking,  
 "house\_size" = house\_size,  
 "type" = property\_type  
   
 )  
 print(property)  
 property\_data <- rbind(property\_data, property)  
   
   
   
 }  
   
 # if last item is older than 2018, stop looping  
 if (last\_property\_sold\_year>=2018) {  
 page = page + 1  
 } else {  
 break;  
 }  
   
 }  
   
   
   
   
 },  
 error = function(cond) {  
 writeLines(sprintf("Error at index : %d, error : %s", row\_index, cond))  
 # return(property\_data)  
 },  
 finally = {  
   
   
 return (property\_data)  
   
 }  
 )  
   
}

Call function to get the property data in parallel

remain\_property\_data <- c(1:no\_of\_data)  
results <- mclapply(remain\_property\_data, scrape\_data\_from\_list\_file, mc.cores = no\_of\_cores)  
all\_property\_data <- ldply(results)

Process the property data in save into a csv for future use

# keep the prooperty which sold data is later than 2018  
filtered\_all\_property\_data <- all\_property\_data %>% filter(date >= "2018-01-01")  
filtered\_all\_property\_data$suburb <- str\_to\_title(filtered\_all\_property\_data$suburb)  
write.csv(filtered\_all\_property\_data, file.path("property\_web\_scraping", "new\_property\_data.csv"), row.names=TRUE)

Read the csv file and check the content

new\_property\_data <- read\_csv(file.path("..", "property\_web\_scraping", "new\_property\_data.csv"))  
str(new\_property\_data)

## tibble [198,982 × 12] (S3: spec\_tbl\_df/tbl\_df/tbl/data.frame)  
## $ X1 : num [1:198982] 1 2 3 4 5 6 7 8 9 10 ...  
## $ date : Date[1:198982], format: "2019-07-20" "2019-06-01" ...  
## $ price : num [1:198982] 3150000 1975000 5700000 1490000 3500000 ...  
## $ address : chr [1:198982] "502/17 Barangaroo Avenue, BARANGAROO NSW 2000" "503/31 Barangaroo Avenue, BARANGAROO NSW 2000" "702/17 Barangaroo Avenue, BARANGAROO NSW 2000" "203/19 Barangaroo Avenue, BARANGAROO NSW 2000" ...  
## $ suburb : chr [1:198982] "Barangaroo" "Barangaroo" "Barangaroo" "Barangaroo" ...  
## $ postcode : num [1:198982] 2000 2000 2000 2000 2000 2000 2000 2000 2000 2000 ...  
## $ url : chr [1:198982] "https://www.domain.com.au/502-17-barangaroo-avenue-barangaroo-nsw-2000-2015388329" "https://www.domain.com.au/503-31-barangaroo-avenue-barangaroo-nsw-2000-2016121789" "https://www.domain.com.au/702-17-barangaroo-avenue-barangaroo-nsw-2000-2014965904" "https://www.domain.com.au/203-19-barangaroo-avenue-barangaroo-nsw-2000-2014416067" ...  
## $ no\_of\_bed : num [1:198982] 2 1 3 1 2 2 3 2 2 1 ...  
## $ no\_of\_bath : num [1:198982] 2 1 3 1 2 2 2 2 2 1 ...  
## $ no\_of\_parking: num [1:198982] 1 1 2 NA 1 1 2 1 1 NA ...  
## $ house\_size : num [1:198982] NA NA NA NA NA NA NA 99 83 65 ...  
## $ type : chr [1:198982] "Apartment / Unit / Flat" "Apartment / Unit / Flat" "Apartment / Unit / Flat" "Apartment / Unit / Flat" ...  
## - attr(\*, "spec")=  
## .. cols(  
## .. X1 = col\_double(),  
## .. date = col\_date(format = ""),  
## .. price = col\_double(),  
## .. address = col\_character(),  
## .. suburb = col\_character(),  
## .. postcode = col\_double(),  
## .. url = col\_character(),  
## .. no\_of\_bed = col\_double(),  
## .. no\_of\_bath = col\_double(),  
## .. no\_of\_parking = col\_double(),  
## .. house\_size = col\_double(),  
## .. type = col\_character()  
## .. )

head(new\_property\_data)

## # A tibble: 6 x 12  
## X1 date price address suburb postcode url no\_of\_bed no\_of\_bath  
## <dbl> <date> <dbl> <chr> <chr> <dbl> <chr> <dbl> <dbl>  
## 1 1 2019-07-20 3.15e6 502/17… Baran… 2000 http… 2 2  
## 2 2 2019-06-01 1.98e6 503/31… Baran… 2000 http… 1 1  
## 3 3 2019-03-08 5.70e6 702/17… Baran… 2000 http… 3 3  
## 4 4 2018-07-02 1.49e6 203/19… Baran… 2000 http… 1 1  
## 5 5 2018-02-14 3.50e6 202/31… Baran… 2000 http… 2 2  
## 6 6 2020-03-12 1.35e6 109/33… Hayma… 2000 http… 2 2  
## # … with 3 more variables: no\_of\_parking <dbl>, house\_size <dbl>, type <chr>

# Crime data preparation

library(tidyverse)  
library(dplyr)

Input raw data and crime type mapping

crime\_data <- read.csv("[Data] Crime\_Data\_by\_postcode.csv")  
crime\_type <- read.csv("[Mapping] Crime\_Type.csv")

### The Process

First, unselect columns (from column Jan.2010 to column Dec.2017) to extract dataset from 2018 to 2019.

Then gather the non-variable columns (from column Jan.2018 to column Dec.2019) into a two-column key-value pair (month\_year, number\_of\_case).

Next, join with crime\_type data with common variables (Offence.category, subcategory)

Finally, group data and summaries by Postcode and Crime.Type to show the total number of crime cases over 2018 to 2019.

total\_crime\_18\_19 <- crime\_data %>%  
 select(-(Jan.10:Dec.17)) %>%  
 gather(month\_year, number\_of\_case, Jan.18:Dec.19, na.rm = TRUE) %>%  
 left\_join(crime\_type, c("Offence.category", "Subcategory")) %>%  
 select(Postcode, Crime.Type, Offence.category, Subcategory, month\_year, number\_of\_case) %>%  
 group\_by(Postcode, Crime.Type) %>%  
 summarise(Total.Crime = sum(number\_of\_case)) %>%  
 spread(Crime.Type, Total.Crime) %>%  
 arrange(Postcode)

# School and Transport Data preparation

### The Process

First, input the raw data

property\_data <- read.csv("property\_data\_v3.csv")%>%  
 select(id,longtitude,latitude)  
station\_data <- read.csv("./Distance From Train Station/[Data] NSW\_Station\_Entrances\_Locations\_2020.csv")  
public\_school\_data <- read.csv("./Distance From Public School/[Data] NSW\_public\_schools\_2016.csv")  
private\_school\_data <- read.csv("./Distance From Non-Government School/[Data] NSW\_non\_government\_school\_2017.csv")

# Distances from Schooling

Data has been gathered from both the public and private schooling systems

## Distance From the closest Public School

### The Process

Using the longtitude and latitude data of properties and NSW public schools, calculate the distance between each property to each public school. Then choose the shortest distance for each property to form the “distance\_from\_closest\_public\_school” variable.

temp <- round(distm(cbind(property\_data$longtitude, property\_data$latitude), cbind(public\_school\_data$Longitude, public\_school\_data$Latitude), fun = distGeo))/1000  
for (i in 1:nrow(property\_data)){  
 public\_school\_number <- which(temp[i, ] == min(temp[i, ]))  
 property\_data$distance\_from\_closest\_public\_school[i] <- temp[i, public\_school\_number]  
}

## Distance From the closest Non-Government School

### The Process

Using the longtitude and latitude data of properties and NSW non-government schools, calculate the distance between each property to each non-government school. Then choose the shortest distance for each property to form the “distance\_from\_closest\_private\_school” variable.

temp <- round(distm(cbind(property\_data$longtitude, property\_data$latitude), cbind(private\_school\_data$longtitude, private\_school\_data$latitude), fun = distGeo))/1000  
for (i in 1:nrow(property\_data)){  
 private\_school\_number <- which(temp[i, ] == min(temp[i, ]))  
 property\_data$distance\_from\_closest\_private\_school[i] <- temp[i, private\_school\_number]  
}

# Distance from nearest Train Station

### The process

Using the longtitude and latitude data of properties and train station entrances, calculate the distance between each property to each train station entrance. Then choose the shortest distance for each property to form the “distance\_from\_closest\_station” variable.

temp <- round(distm(cbind(property\_data$longtitude, property\_data$latitude), cbind(station\_data$LONG, station\_data$LAT), fun = distGeo))/1000  
for (i in 1:nrow(property\_data)){  
 station\_entrance\_number <- which(temp[i, ] == min(temp[i, ]))  
 property\_data$distance\_from\_closest\_station[i] <- temp[i, station\_entrance\_number]  
}

# Weather Information

Daily weather data has been gathered and an average taken across each month for the two year period

### The Process

weather=read.csv("MonthlyAverageTemp.csv")  
is.data.frame(weather)  
sum(is.na(weather))  
head(weather)  
summary(weather)

agg=aggregate(temp\_avg~date, data=weather, FUN=mean)  
agg

ggplot(agg, aes(x=date, y=temp\_avg, group=1))+  
 coord\_fixed(ratio=0.8)+  
 geom\_line(color="steel blue", size=1)

# Gathering Geocode data

Geocode data has been gathered to use as a reference point for further analysis

### The Process

##Get geocode data  
  
library(jsonlite)  
library(parallel)  
library(stringr)  
library(tidyverse)  
library(plyr)  
no\_of\_cores <- detectCores() - 1  
  
data <- read.csv("property\_data\_v2.csv", stringsAsFactors = FALSE)  
  
#create address for iteration  
address <- paste(data[,3], data[,4], "NSW+Australia", sep="+")  
address <- str\_replace\_all(address, " ", "+")  
#replace space with plus sign  
  
head(address)  
geocode\_data <- data.frame(  
 property\_id = integer(),  
 subpremise = character(),  
 street\_number = character(),  
 street\_name = character(),  
 suburb = character(),  
 council = character(),  
 state = character(),  
 country = character(),  
 postal\_code = character(),  
 formatted\_address = character(),  
 lat = numeric(),  
 lon = numeric()  
)   
   
  
#api key for google API access  
api\_key = "ENTER USE API KEY"  
base\_url <- "https://maps.googleapis.com/maps/api/geocode/json?key=%s&address=%s"  
  
for (i in 1:length(address)){  
 tryCatch({  
 #request address and geolocation in JSON format using Google Geocode API  
 temp1 <- data.frame(fromJSON(sprintf(base\_url, api\_key, address[i])))  
 #select subpremise, stree\_number, street\_name, suburb, council, state, country, postal\_code  
 temp2 <- t(data.frame(temp1$results.address\_components[[1]]$long\_name))  
 colnames(temp2) <- unlist(lapply(temp1$results.address\_components[[1]]$types, function(x) x[1]))  
 #select formatted\_address, lat and lon  
 temp3 <- temp1[,-grep("results.address\_components", names(temp1))] %>%  
 unlist() %>%  
 t() %>%  
 data.frame() %>%  
 select(results.formatted\_address, results.geometry.location.lat, results.geometry.location.lng)  
 colnames(temp3) <- c("formatted\_address", "lat", "lon")  
 #join data  
 geocode\_data <- rbind.fill(geocode\_data, data.frame(property\_id = i, cbind(temp2, temp3)))  
 },  
 error = function(e) {  
 writeLines(sprintf("Error at index : %d, error : %s", i, e))  
 }  
 )  
}  
  
write.csv(geocode\_data, "geocode\_data.csv", row.names = FALSE)

# Liveability Data

Liveability Data has bee gathered across Sydney’s 569 suburbs

### The Process

##Get lieability  
  
library(tidyverse)  
library(rvest)  
  
#Sydney’s 569 suburbs ranked for liveability  
url = "https://www.domain.com.au/liveable-sydney/sydneys-most-liveable-suburbs-2019/sydneys-569-suburbs-ranked-for-liveability-2019-903130/"  
temp <- read\_html(url)  
  
suburbs <- temp %>%  
 html\_nodes(xpath = '//h3') %>%  
 html\_text()  
  
info <- temp %>%   
 html\_nodes(xpath = '//\*[@id="post-903130"]/section/p') %>%  
 html\_text()   
  
data <- data.frame(suburbs) %>%  
 separate(info, c("ranking", "suburb"), "\\.") %>%  
 na.omit() %>%  
 mutate(ranking = as.numeric(ranking))  
  
data$info = info[-1]  
  
write.csv(data, "liveability.csv", row.names = FALSE)

# The distance from the CBD as a reference point

The distance from the CBD has been gathered for each property sale

### The Process

##Get distance from CBD  
library(geosphere)  
library(tidyverse)  
#latitude & longitude of 2000 Sydney NSW is 33.8688° S, 151.2093° E   
distance\_km <- function(lon1, lat1){  
 distm(c(lon1, lat1), c(151.2093, -33.8688), fun = distGeo)/1000  
}  
  
data <- read.csv("geocode\_data.csv")  
#select row\_index, latitude and longitude  
data <- data[, c(1, (ncol(data)-1):ncol(data))]  
names(data)  
names(data) <- c("no", "lat1", "lon1")  
  
#calculate distance for across all properties  
distance\_from\_CBD\_km <- apply(data, 1, function(x) distance\_km(x["lon1"], x["lat1"]))  
data <- data[,c(1,4)]  
data <- cbind(data, distance\_from\_CBD\_km = distance\_from\_CBD\_km)  
  
write.csv(data, "distance\_from\_CBD.csv", row.names=FALSE)

# Collection of BushFire Data in the Sydney Region

Bushfire data has been collected and the distances between the property sales and recent fires has been analysed.

### The Process

Reading Data : NASA FIRMS

First, import the necessary libraries

Import raw data having filter conditions applied

Second, have a look on the structure of data and number of rows data

nrow(DF\_2019)  
str(DF\_2019)

Next, initialise with an empty tibble

res\_nf <- as\_tibble()

Merging with Property Geocode:

Then every fire pixel data compared against geocode data of every NSW property and finding out its the distance from the property using distHarvesine method. This will return all the hotspots/fire pixels at the radius upto 15km distance.

for(i in 1: nrow(Df\_2019)){  
 Df2019\_ind <- Df\_2019[i,]  
 p\_gc <- as\_tibble(fread("DataSet/geocode\_data.csv", header = TRUE, stringsAsFactors = FALSE) %>%  
 filter(country=='Australia' & state=='New South Wales') %>%  
 mutate(  
 #Apply distHarvesine method  
 dist\_between=distHaversine(cbind(Df2019\_ind$longitude,Df2019\_ind$latitude),cbind(lng,lat))/1000,  
 FIRMS\_19\_lat=Df2019\_ind$latitude,  
 FIRMS\_19\_lng=Df2019\_ind$longitude,  
 FIRMS\_19\_cf=Df2019\_ind$confidence,  
 FIRMS\_19\_aq=Df2019\_ind$acq\_date  
 ) %>%  
 filter(dist\_between<=15) %>%  
 dplyr::select(FIRMS\_19\_lat,FIRMS\_19\_lng,FIRMS\_19\_cf,FIRMS\_19\_aq,dist\_between,no,postal\_code))  
   
 res\_nf <- rbind(p\_gc,res\_nf)  
}

Then export the result set into csv file like:

write\_csv(res\_nf,"Results/NASA\_Firms\_Impact\_15km\_2019.csv")

The above process will be repeated for 2018-09 to 2019-03 as well.

# ATO Deferral Data

ATO deferral data has been collected across the Sydney Region

### The Process

First, import the necessary libraries

library(xml2)  
library(rvest)  
library(stringr)  
library(XML)  
library(data.table)  
library(tidyverse)

Next,connect to the ATO webpage URL and retrieve LGA’s and Postcode related to New South Wales

ato\_webpg <- read\_html("https://www.ato.gov.au/Individuals/Dealing-with-disasters/In-detail/Specific-disasters/Bushfires-2019-20/?anchor=NotinanimpactedpostcodeorLGA#NewSouthWales")  
df <- ato\_webpg %>%  
 html\_nodes("div:nth-child(1) > p:nth-child(n+14):nth-child(-n+49)") %>%  
 xml\_text() %>%  
 str\_split\_fixed(":", 2)

Parse the data to pivot the postcode values into tabular format

LGA\_Names <- df[,1]  
postal\_code <- strsplit(df[,2],",")  
parse <-""  
for (i in 1:nrow(df)) {  
 parse<-rbind(as.data.frame(list("LGA"=rep(LGA\_Names[i],length(postal\_code[[i]])),"postal\_code"=as.numeric(postal\_code[[i]]))),parse)  
}  
df\_parse <- as.data.frame(parse)

Then export the result set into csv file like:

write.csv(df\_parse,"Bushfire/Results/ATO\_Deferred.csv")

Copmbineing the ATO Deferred Flag with Property data

##### Add ATO Deferred Flag with Property data :  
library(data.table)  
library(tidyverse)  
#Link it with NASA Firms data  
Firms\_19 <- fread("../Bushfire/Results/NASA\_Firms\_Impact\_15km\_2019.csv", header = TRUE, stringsAsFactors = FALSE)  
Firms\_18 <- fread("../Bushfire/Results/NASA\_Firms\_Impact\_15km\_2018.csv", header = TRUE, stringsAsFactors = FALSE)  
ATO\_Def <- fread("../Bushfire/Results/ATO\_Deferred.csv", header = TRUE, stringsAsFactors = FALSE) %>%  
 select(LGA,postal\_code)  
Firms\_19\_ato <- left\_join(Firms\_19, ATO\_Def, by="postal\_code") %>%  
 dplyr::select(FIRMS\_19\_lat,FIRMS\_19\_lng,FIRMS\_19\_cf,FIRMS\_19\_aq,dist\_between,no,postal\_code,LGA)  
Firms\_19\_ato$ato\_flag <- if\_else(is.na(Firms\_19\_ato$LGA),"N","Y")   
Firms\_18\_ato <- left\_join(Firms\_18, ATO\_Def, by="postal\_code") %>%  
 dplyr::select(FIRMS\_18\_lat,FIRMS\_18\_lng,FIRMS\_18\_cf,FIRMS\_18\_aq,dist\_between,no,postal\_code,LGA)  
Firms\_18\_ato$ato\_flag <- if\_else(is.na(Firms\_18\_ato$LGA),"N","Y")   
write.csv(Firms\_19\_ato,"../Bushfire/Results/NASA\_Firms\_Impact\_15km\_2019\_atoflag.csv")  
write.csv(Firms\_18\_ato,"../Bushfire/Results/NASA\_Firms\_Impact\_15km\_2018\_atoflag.csv")

Finally, view the results:

as\_tibble(fread("../BushFire/Results/NASA\_Firms\_Impact\_15km\_2018\_atoflag.csv")) %>%   
 glimpse()

## Rows: 28,938  
## Columns: 10  
## $ V1 <int> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 1…  
## $ FIRMS\_18\_lat <dbl> -34.10368, -34.10368, -34.10368, -34.10368, -34.10368, -…  
## $ FIRMS\_18\_lng <dbl> 150.9636, 150.9636, 150.9636, 150.9636, 150.9636, 150.96…  
## $ FIRMS\_18\_cf <chr> "h", "h", "h", "h", "h", "h", "h", "h", "h", "h", "h", "…  
## $ FIRMS\_18\_aq <chr> "10/03/2019", "10/03/2019", "10/03/2019", "10/03/2019", …  
## $ dist\_between <dbl> 12.876231, 14.677182, 13.705161, 12.669885, 4.919849, 14…  
## $ no <int> 46, 63, 147, 148, 151, 163, 180, 359, 360, 361, 402, 463…  
## $ postal\_code <int> 2226, 2229, 2227, 2227, 2233, 2232, 2210, 2233, 2228, 22…  
## $ LGA <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, …  
## $ ato\_flag <chr> "N", "N", "N", "N", "N", "N", "N", "N", "N", "N", "N", "…

# Migration Data for the Sydney Region from ABS

Based on the data collected from ABS via (3218.0 - Regional Population Growth, Australia, 2018-19, 2017-18), we extracted the net migration numbers for every suburb/council recorded by ABS.

### The Process

library(readxl)  
library(tidyverse)  
#2019 Dataset  
setwd("./../NOM/DataSet")  
xl\_data\_2019 <- "NOM\_32180ds0002\_2018-19.xls"  
xl\_data\_2018 <- "NOM\_32180ds0002\_2017-18.xls"  
df\_nsw\_2019 <- read\_excel(path = xl\_data\_2019, sheet = "Table 1")  
df\_nsw\_2018 <- read\_excel(path = xl\_data\_2018, sheet = "Table 1")  
cnt\_2019 <- nrow(df\_nsw\_2019)  
cnt\_2018 <- nrow(df\_nsw\_2018)  
#Remove headers and read only the data rows corresponding to NOM no's  
Ext\_tib\_2019 <- data.frame(rep(2019,cnt\_2019-8),  
 df\_nsw\_2019[9:cnt\_2019,2],  
 df\_nsw\_2019[9:cnt\_2019,11]) %>%  
 na.omit()   
colnames(Ext\_tib\_2019) <- c("Year", "Suburb", "No\_ofNOM")  
Ext\_tib\_2018 <- data.frame(rep(2018,cnt\_2018-8),  
 df\_nsw\_2018[9:cnt\_2018,2],  
 df\_nsw\_2018[9:cnt\_2018,11]) %>%  
 na.omit()   
colnames(Ext\_tib\_2018) <- c("Year", "Suburb", "No\_ofNOM")  
colnames(Ext\_tib\_2019) <- c("Year", "LGA", "Nom")  
colnames(Ext\_tib\_2018) <- c("Year", "LGA", "Nom")  
write.csv(rbind(Ext\_tib\_2019,Ext\_tib\_2018),"./Results/Nom\_LGA.csv")

# SocioEconomic Indicator Indexes (SEIFA) Data

SocioEconomic Data based on the data collected from ABS via (2071.0 - Census of Population and Housing: Reflecting Australia - Stories from the Census, 2016), we extracted Index of Relative Socio-economic Advantage and Disadvantage,Index of Economic Resources ,Index of Education and Occupation Scores and Decile points

### The Process

library(readxl)  
library(tidyverse)  
#2016 Dataset  
xl\_data\_2016 <- "./SEIFA/2033055001 - ssc indexes.xls"  
df\_nsw\_2016 <- read\_excel(path = xl\_data\_2016, sheet = "Table 1")  
cnt\_2016 <- nrow(df\_nsw\_2016)  
Ext\_tib\_2016 <- data.frame(Year = rep(2016,cnt\_2016-5),  
 Suburb = df\_nsw\_2016[6:cnt\_2016,2],  
 SAdvDisav\_score = df\_nsw\_2016[6:cnt\_2016,5], #Socio-Economic Score   
 SAdvDisav\_decile = df\_nsw\_2016[6:cnt\_2016,6], #RSocio-Economic Decile  
 ER\_score= df\_nsw\_2016[6:cnt\_2016,7], #Economic Resources Score  
 ER\_decile= df\_nsw\_2016[6:cnt\_2016,8], #Economic Resources Decile  
 EDUOCC\_score= df\_nsw\_2016[6:cnt\_2016,9], #Education and Occupation Score  
 EDUOCC\_decile= df\_nsw\_2016[6:cnt\_2016,10] #Education and Occupation Decile  
 )  
write.csv(Ext\_tib\_2016,"SEIFA/Results/SEIFA\_2016.csv")