Appendix 1 - Code used to collect data

Group 5

Contents


```
## [1] FALSE
```

1 Collection of Property Data

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

1.0.1 The Process

1.0.1.0.1 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

1.0.1.1 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&s
no_of_cores <- detectCores() - 1
# create directory
dir.create(file.path("property_web_scraping", "property_listing_html"))
```

1.0.1.2 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 x 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

      # 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)
```

1.0.1.3 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) {
```

```r
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"]')
  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-pri
  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-f
  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"]') %>% h


      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.
```

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 x 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 Aven
 $ 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
 $ 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 /
 - 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>
```

2 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")
```

2.0.1 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)
```

3 School and Transport Data preparation

3.0.1 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
```

4 Distances from Schooling

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

4.1 Distance From the closest Public School

4.1.1 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$L
for (i in 1:nrow(property_data)){
  public_school_number <- which(temp[i, ] == min(temp[i, ]))
  property_data$distance_from_closest_public_school[i] <- temp[i, public_school_number]
}
```

4.2 Distance From the closest Non-Government School

4.2.1 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$
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]
}
```

5 Distance from nearest Train Station

5.0.1 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$LONG, s
for (i in 1:nrow(property_data)){
  station_entrance_number <- which(temp[i, ] == min(temp[i, ]))
  property_data$distance_from_closest_station[i] <- temp[i, station_entrance_number]
}
```

6 Weather Information

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

6.0.1 The Process

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

```r
agg=aggregate(temp_avg~date, data=weather, FUN=mean)
agg
```

```r
ggplot(agg, aes(x=date, y=temp_avg, group=1))+
  coord_fixed(ratio=0.8)+
  geom_line(color="steel blue", size=1)
```

7 Gathering Geocode data

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

7.0.1 The Process

```r
##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)
```

8 Liveability Data

Liveability Data has bee gathered across Sydney's 569 suburbs

8.0.1 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
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)
```

9 The distance from the CBD as a reference point

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

9.0.1 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)
```

10 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.

10.0.1 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))
              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.

11 ATO Deferral Data

ATO deferral data has been collected across the Sydney Region

11.0.1 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-dis
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.num
}
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
Firms_18 <- fread("../Bushfire/Results/NASA_Firms_Impact_15km_2018.csv", header = TRUE, stringsAsFactors
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,...
$ 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....
$ 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, ...
$ no           <int> 46, 63, 147, 148, 151, 163, 180, 359, 360, 361, 402, 4...
$ postal_code  <int> 2226, 2229, 2227, 2227, 2233, 2232, 2210, 2233, 2228, ...
$ 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",...
```

12 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.

12.0.1 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")
```

13 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

13.0.1 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")
```