Merge Code Sample

Wenshu Yang (Monica)

1. Introduction

This coding sample is merging three different data sets with policing and crime data by precincts. The goal is to get a final data set with information on each Stop, Question and Frisk case from 2010 to 2014, the census population of difference races of the precincts where the Stop, Question and Frisk happened, and the crime statistics of the precincts.

The first set of data are the Stop, Question and Frisk (SQF) Data, which are available at the NYPD website https://www1.nyc.gov/site/nypd/stats/reports-analysis/stopfrisk.page. There are five separate data sets used here, for each year from 2010 to 2014. Each of them contains details about each case of stop/question/frisk by police, such as the demographics of the person encountering the police enforcement, the precints and geographic information of the stop/question/frisk spots.

The second data set is the 2010 Census Redistricting Data (Public Law 94-171), which contains the population for different races of each block in each precinct. The data set is available at https://fusiontables.google.com/data?docid=1rxy8ycg9ZsRZG7z9sNQdottvAwIaNVCNCRKr9Q#rows:id=1.

The third data set contains historical crime statistics, which are the numbers of the citywide seven major felony offenses by precinct from 2000 to 2018. In this data set, each precinct has multiple observations, with each observation representing a kind of felony.

2. Merge the 2010 Census Redistricting Data and the Crime Statistics Data

This step is merging the 2010 Census Redistricting Data and the major felony offenses crime data by percincts to get each percinct's population and crime statistics.

Read the data

```
# 2010 Census Redistricting Data
census <- read.csv("Data/Precinct-Census/NYC_Blocks_2010CensusData_Plus_Precincts.csv")

# Crime Data
crime <- read_excel("Data/Precinct-Crime/seven-major-felony-offenses-by-precinct-2000-2018.xls", skip =
```

Pre-process of data

Pre-process the Census Data

```
# Get the useful variables
census <- census %>%
  ## Drop the obs with no data on percinct
  filter(!is.na(precinct)) %>%
  ## Keep only the needed variables: the population of different races (one race alone)
  select(precinct, pct_ttl_ppl = P0010001, pct_onerace_ttl = P0010002,
         pct_white = P0010003, pct_black = P0010004, pct_indian_alaska = P0010005,
         pct_asian = P0010006, pct_hawaiian = P0010007,
         pct_other_onerace = P0010008) %>%
```

```
  ## Add a variable of the population for people with two or more races
  mutate(pct_twomore_race_ttl = pct_ttl_ppl - pct_onerace_ttl)


# Aggregate the population of all the blocks in each precinct
census <- census %>%
  group_by(precinct) %>%
  summarise_all(sum, na.rm = TRUE)
```

Preprocess of the Crime Data

```
# Remove the last 12 rows which are just some notes
crime <- crime[-(nrow(crime):(nrow(crime)-11)), ]


# Rename the "PCT" variable as "precinct" to match the variable name in the census data set
crime <- rename(crime, precinct = PCT)


# The precinct numbers are stored in merged cells in the excel file,
# so some became NAs when read into R
## Fill the NAs with the neareast last precinct numbers
for (i in 1:nrow(crime)) {
  if (is.na(crime$precinct[i])) {
    crime$precinct[i] <- crime$precinct[i-1]
  }
}


# Remove the "DOC" precinct:
## these are complaints occurring within the jurisdiction of the Department of Correction
## and are disaggregated from the precinct crime totals
crime <- filter(crime, precinct != "DOC")


# Tranform the precinct variable from character to numeric
## in order to match the data type of the precinct variable in the census data set
crime$precinct <- as.numeric(crime$precinct)


# Keep only year 2010 to year 2014
crime <- select(crime, precinct, CRIME, "2010", "2011", "2012", "2013", "2014")


# Reshape the data
## From mixed to long format (year as a new variable)
crime <- melt(crime, id.vars = c("precinct", "CRIME"),
              measure.vars = c("2010", "2011", "2012", "2013", "2014"),
              variable.name = "year", value.name = "n_felony")

## From long to mixed format (crime types as new variables)
crime <- reshape(crime, direction = "wide", idvar = c("precinct", "year"),
                 timevar = "CRIME", v.names = "n_felony")
```

```r
### Fix the variable names of the crime types
names(crime) <- gsub("n_felony.", "", x = names(crime))
names(crime)[-(1:2)] <- paste0("pct_", tolower(names(crime)[-(1:2)]))


# Convert the year variable from factor to numeric
crime$year <- as.character(crime$year) %>%
  as.numeric()
```

Merge the two data sets

```r
# Merge the data
census_crime <- merge(census, crime, by = "precinct")

# Save the merged data as csv
#write.csv(census_crime, file = "Data/census_crime.csv")
```

3. Merge the above data set into the SQF data

Read the SQF data

```r
# Read the 2010 to 2014 data sets, store all the data sets of different years in a list
sqf <- list()
for (i in 0:4) {
  sqf[[i+1]] <- read.csv(paste0("Data/SQF/201", i, ".csv"))
}

# Add a column indicating the year
for (i in 0:4) {
  sqf[[i+1]]$year <- 2010 + i
}
```

Preprocess of the SQF data

The five data sets have slightly different variables. Therefore, they need to be matchd by variable names before combining the rows.

```r
# Write the function to match the variable names to combine the two data sets
mc.fun <- function(data1, data2) {

  # Match of data1 variable names in data2 variable names
  ## sqf.1 is for the final rbind with the other data set
  sqf.1 <- data1
  ## If not all the variables in data2 are included in data1, then do the match below
  if (!all(colnames(data2) %in% colnames(data1))) {
    ### position index of the matched data1 variable names in data2
    idx <- match(colnames(data1), colnames(data2))
    ### variables that are in data2 but not in data1
    name.added <- colnames(data2)[-(idx[!is.na(idx)])]
    ### Add these variables as NAs to sqf.1
    for (i in 1:length(name.added)) {
      sqf.1[, ncol(data1)+i] <- NA
      colnames(sqf.1)[ncol(data1)+i] <- name.added[i]
```

```
    }
  }

  # Match of data2 variable names in data1 variable names
  ## The steps are the same as above except that data1 and data2 are inversed
  sqf.2 <- data2
  if (!all(colnames(data1) %in% colnames(data2))) {
    idx <- match(colnames(data2), colnames(data1))
    name.added <- colnames(data1)[-(idx[!is.na(idx)])]
    for (i in 1:length(name.added)) {
      sqf.2[, ncol(data2)+i] <- NA
      colnames(sqf.2)[ncol(data2)+i] <- name.added[i]
    }
  }

  # Combind the rows of the two data sets
  ## Since they have exactly the same variable names, the rows can be binded together
  sqf.all <- rbind(sqf.1, sqf.2)
  return(sqf.all)
}


# Apply the function to combind all the data sets of year 2010 to year 2014
sqf.allyear <- sqf[[1]]
for (i in 2:5) {
  sqf.allyear <- mc.fun(sqf.allyear, sqf[[i]])
}


# The variable names of the sqf.allyear data set
sort(colnames(sqf.allyear))
```

```
##   [1] "ac_assoc" "ac_cgdir" "ac_evasv" "ac_incid" "ac_inves" "ac_other"
##   [7] "ac_proxm" "ac_rept"  "ac_stsnd" "ac_time"  "addrnum"  "addrpct"
##  [13] "addrtyp"  "adtlrept" "age"      "aptnum"   "arstmade" "arstoffn"
##  [19] "asltweap" "beat"     "build"    "city"     "comppct"  "compyear"
##  [25] "contrabn" "crimsusp" "crossst"  "cs_bulge" "cs_casng" "cs_cloth"
##  [31] "cs_descr" "cs_drgtr" "cs_furtv" "cs_lkout" "cs_objcs" "cs_other"
##  [37] "cs_vcrim" "datestop" "detailcm" "detailCM" "dettypcm" "dettypCM"
##  [43] "dob"      "explnstp" "eyecolor" "forceuse" "frisked"  "haircolr"
##  [49] "ht_feet"  "ht_inch"  "inout"    "knifcuti" "linecm"   "lineCM"
##  [55] "machgun"  "officrid" "offshld"  "offunif"  "offverb"  "othfeatr"
##  [61] "othpers"  "othrweap" "pct"      "perobs"   "perstop"  "pf_baton"
##  [67] "pf_drwep" "pf_grnd"  "pf_hands" "pf_hcuff" "pf_other" "pf_pepsp"
##  [73] "pf_ptwep" "pf_wall"  "pistol"   "post"     "premname" "premtype"
##  [79] "race"     "radio"    "recstat"  "repcmd"   "rescode"  "revcmd"
##  [85] "rf_attir" "rf_bulg"  "rf_furt"  "rf_knowl" "rf_othsw" "rf_rfcmp"
##  [91] "rf_vcact" "rf_vcrim" "rf_verbl" "riflshot" "sb_admis" "sb_hdobj"
##  [97] "sb_other" "sb_outln" "searched" "sector"   "ser_num"  "sex"
## [103] "state"    "stinter"  "stname"   "sumissue" "sumoffen" "timestop"
## [109] "trhsloc"  "typeofid" "weight"   "xcoord"   "ycoord"   "year"
## [115] "zip"
```

The variable names of the sqf.allyear data set show that there are some variables with the same meanings but

different names (for example, dettypcm and dettypCM). Therefore, the variable names need to be further examined.

```
# Check if there are variables with the same meaning but different names
## Length of the five SQF data sets
sapply(sqf, function(x) length(colnames(x)))
```

```
## [1] 111 112 112 112 112
```

```
## The last four data sets (2011-2014) all have the same length
## So we can first check whether the variable names in these four are exactly the same
all(colnames(sqf[[2]])==colnames(sqf[[3]])) #sqf[[2]] and sqf[[3]] have same variable names
```

```
## [1] TRUE
```

```
all(colnames(sqf[[2]])==colnames(sqf[[4]])) #sqf[[2]] and sqf[[4]] have different variable names
```

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

```
all(colnames(sqf[[2]])==colnames(sqf[[5]])) #sqf[[2]] and sqf[[5]] have different variable names
```

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

```
### Check the difference between sqf[[2]] and sqf[[4]] variable names
idx <- match(colnames(sqf[[2]]), colnames(sqf[[4]]))
only.in.4 <- colnames(sqf[[4]])[-(idx[!is.na(idx)])] #variables only in sqf[[4]]
idx <- match(colnames(sqf[[4]]), colnames(sqf[[2]]))
only.in.2 <- colnames(sqf[[2]])[-(idx[!is.na(idx)])] #variables only in sqf[[2]]
data.frame(only.in.2, only.in.4)
```

```
##   only.in.2 only.in.4
## 1  dettypcm  dettypCM
## 2    linecm    lineCM
## 3  detailcm  detailCM
```

```
  #They are actually the same, but just different in the lower/upper case of the names
### Change the uppercase variable names in sqf[[4]] to lowercase
sqf[[4]] <- rename(sqf[[4]],
                   dettypcm = dettypCM, linecm = lineCM, detailcm = detailCM)
  #Now sqf[[2]] sqf[[3]], sqf[[4]] have the same variable names

### Check the difference between sqf[[2]] and sqf[[5]] variable names
idx <- match(colnames(sqf[[2]]), colnames(sqf[[5]]))
only.in.5 <- colnames(sqf[[5]])[-(idx[!is.na(idx)])] #variables only in sqf[[5]]
idx <- match(colnames(sqf[[5]]), colnames(sqf[[2]]))
only.in.2 <- colnames(sqf[[2]])[-(idx[!is.na(idx)])] #variables only in sqf[[2]]
data.frame(only.in.2, only.in.5)
```

```
##   only.in.2 only.in.5
## 1  dettypcm  dettypCM
## 2    linecm    lineCM
## 3  detailcm  detailCM
```

```
  #It's the same case as sqf[[2]] and sqf[[4]]
sqf[[5]] <- rename(sqf[[5]],
                   dettypcm = dettypCM, linecm = lineCM, detailcm = detailCM)
  #Now the last four data sets have the same variable names
```

```
## Compare sqf[[1]] and sqf[[2]] variable names
all(colnames(sqf[[1]]) %in% colnames(sqf[[2]]))
```

```
## [1] TRUE
```

```
  #the variable names in sqf[[1]] are all included in sqf[[2]]
### Identify the one not in [[1]]
idx <- match(colnames(sqf[[1]]), colnames(sqf[[2]]))
not.in.1 <- colnames(sqf[[2]])[-idx] #this is the "forceuse" variable
### Add this variable to sqf[[1]] as NAs
sqf[[1]]$forceuse <- NA
```

Now all the variables in the five data sets are the same, each with unique meanings. This time their rows can be combined together correctly.

```
# Combind the rows of the five data sets
sqf_allyear <- sqf[[1]]
for (i in 2:5) {
  sqf_allyear <- rbind(sqf_allyear, sqf[[i]])
}

# Rename the variable "pct" as "precinct" to match the variable name is the census_crime data set
sqf_allyear <- rename(sqf_allyear, precinct = pct)

# Save the 5-year SQF data set as csv
#write.csv(sqf_allyear, file = "Data/sqf_allyear.csv")
```

Merge the Census_Crime Data Set into the All-year SQF Data Set

```
# Merge the data
final_data <- left_join(sqf_allyear, census_crime, by = c("precinct", "year"))

# Save the final data set as csv
#write.csv(final_data, file = "Data/final_data.csv")
```