Merging Gentzkow Replication Data with Census Places

nwfried

2024-03-18

Data Wrangling

Initialising libraries.

```
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.4     v readr     2.1.5
## v forcats   1.0.0     v stringr   1.5.1
## v ggplot2   3.5.1     v tibble    3.2.1
## v lubridate 1.9.3     v tidyr     1.3.1
## v purrr     1.0.2
## -- Conflicts ---------------------------------------- tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```
library(fedmatch)
```

Loading in data.

```
cities <- read_tsv("30261-0006-Data.tsv")
```

```
## Rows: 2159 Columns: 4
## -- Column specification --------------------------------------------------------
## Delimiter: "\t"
## chr (2): cityname_constant, state
## dbl (2): citypermid, cnty90
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
census_places <- read_csv("places2msa1970.csv")
```

```
## Rows: 6584 Columns: 37
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (24): STATE, NHGISST, PLACE, NAME, NHGISPLACE, GISJOIN, NHGISNAM, NHGISS...
## dbl (12): YEAR, DECADE, ICPSRST, ICPSRCTY, ICPSRSTI, ICPSRCTYI, ICPSRFIP, PI...
## lgl  (1): entityfips
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

First, we need to deal with the fact that Gentzkow's cities dataframe contains city names in all caps and the Census dataframe doesn't. This won't work with fuzzy match, so let's format both of these in all lowercase.

```
cities$cityname_constant <- tolower(cities$cityname_constant)
census_places$NAME <- tolower(census_places$NAME)
census_places$PLACE <- tolower(census_places$PLACE)
```

Next, we can fuzzy match based on the "NAME" column.

```
fuzzy_merge <- merge_plus(data1 = census_places,
          data2 = cities,
          by.x = "NAME",
          by.y = "cityname_constant", match_type = "fuzzy",
          unique_key_1 = "NHGISPLACE", unique_key_2 = "citypermid")

name_result <- fuzzy_merge$matches
```

Let's look at this result.

```
head(name_result)
```

```
## Key: <citypermid>
##    citypermid NHGISPLACE    STATE NHGISST             PLACE  YEAR  GISJOIN
##         <num>     <char>   <char>  <char>            <char> <num>   <char>
## 1:          1 G170407670 Illinois     170 la grange village  1970 G1703065
## 2:          1 G390412300     Ohio     390  lagrange village  1970 G3902085
## 3:          2 G390247660     Ohio     390  eldorado village  1970 G3901275
## 4:          4 G200200750   Kansas     200    el dorado city  1970 G2000780
## 5:          6 G020661400   Alaska     020   st. mary's city  1970 G0202860
## 6:          9 G100508000 Delaware     100   new castle city  1970 G1000200
##    DECADE   NHGISNAM NHGISST_2 NHGISCTY ICPSRST ICPSRCTY   ICPSRNAM STATENAM
##     <num>     <char>    <char>   <char>   <num>    <num>     <char>   <char>
## 1:   1970       Cook       170     0310      21      310       COOK Illinois
## 2:   1970     Lorain       390     0930      24      930     LORAIN     Ohio
## 3:   1970     Greene       390     0570      24      570     GREENE     Ohio
## 4:   1970     Butler       200     0150      32      150     BUTLER   Kansas
## 5:   1970     Juneau       020     1100      81     1100     JUNEAU   Alaska
## 6:   1970 New Castle       100     0030      11       30 NEW CASTLE Delaware
##    ICPSRSTI ICPSRCTYI ICPSRFIP STATE_2 COUNTY   PID X_CENTROID Y_CENTROID
##       <num>     <num>    <num>  <char> <char> <num>      <num>      <num>
## 1:       21       310        0     170   0310   596   673976.8  515200.60
## 2:       24       930        0     390   0930  1228  1146699.6  508982.44
## 3:       24       570        0     390   0570  1213  1026419.6  311105.56
## 4:       32       150        0     200   0150  1007   -73203.1   31813.45
## 5:       81      1100        0    <NA>   <NA>     0        0.0       0.00
## 6:       11        30        0     100   0030   277  1718758.8  417391.14
##    GISJOIN_2 GISJOIN2 SHAPE_AREA SHAPE_LEN statefips countyfips   fips smsacode
##       <char>   <char>      <num>     <num>    <char>     <char> <char>   <char>
## 1:  G1700310  1700310 2481420600  303706.3        17        031  17031     1600
## 2:  G3900930  3900930 1281184984  172594.8        39        093  39093     4440
## 3:  G3900570  3900570 1077986175  137720.1        39        057  39057     2000
## 4:  G2000150  2000150 3746230669  246761.8        20        015  20015     9040
## 5:  G0201100  0201100 5028498768 1235933.0      <NA>       <NA>   <NA>     <NA>
## 6:  G1000030  1000030 1121606309  220727.1        10        003  10003     9160
##    statefips_ countyfi_1 entityfips               name_2
##        <char>     <char>     <lgcl>               <char>
## 1:         17        031         NA          Cook County
## 2:         39        093         NA        Lorain County
```

```
## 3:         39        057         NA         Greene County
## 4:         20        015         NA         Butler County
## 5:       <NA>       <NA>         NA                  <NA>
## 6:         10        003         NA New Castle County, DE
##                            namemsa fips_2       NAME cityname_constant cnty90
##                             <char> <char>     <char>            <char>  <num>
## 1:                Chicago, IL SMSA  17031  la grange          lagrange  13285
## 2:           Lorain-Elyria, OH SMSA  39093   lagrange          lagrange  13285
## 3:                 Dayton, OH SMSA  39057   eldorado          eldorado  17165
## 4:                Wichita, KS SMSA  20015  el dorado         el dorado  20015
## 5: Wilkes-Barre--Hazleton, PA SMSA   <NA> st. mary's        st. mary's  39011
## 6:       Wilmington, DE-NJ-MD SMSA  10003 new castle        new castle  42073
##           state   tier
##          <char> <char>
## 1:      GEORGIA    all
## 2:      GEORGIA    all
## 3:     ILLINOIS    all
## 4:       KANSAS    all
## 5:         OHIO    all
## 6: PENNSYLVANIA    all
```

The problem here is that many place names are often reused, so the fuzzy match will match cities with the same name in different states e.g. Springfield, Illinois is matched will all other Springfields. We can use the county numbers of each city (found in the cnty90 and fips_2 columns, respectively) to make sure that the cities matched by the fuzzy match are actually the same. (we could also check this with state names, but I thought more specificty would be better at avoiding false positives).

First, we have to account for the fact that the four-digit county numbers in the Census dataframe are prefixed with a "0," while the ones in the city databse are not. Let's add this 0 to the city database.

```
cnty_number <- cities$cnty90
fixed_cnty_number <- sprintf("%05d", cnty_number) #fix to 5 character width
cities <- cbind(fixed_cnty_number, cities) #add to dataset
```

Now we can do another fuzzy match and then filter out only the entries with matching county numbers.

```
merge2 <- merge_plus(data1 = census_places,
          data2 = cities,
          by.x = "NAME",
          by.y = "cityname_constant", match_type = "fuzzy",
          unique_key_1 = "NHGISPLACE", unique_key_2 = "citypermid")
result2 <- merge2$matches
unmatched_census <- merge2$data1_nomatch
unmatched_cities <- merge2$data2_nomatch
```

```
filtered_result2 <- filter(result2, fixed_cnty_number == fips_2)
head(filtered_result2)
```

```
## Key: <citypermid>
##    citypermid NHGISPLACE         STATE NHGISST                PLACE  YEAR
##         <num>     <char>        <char>  <char>               <char> <num>
## 1:          4 G200200750        Kansas     200       el dorado city  1970
## 2:         39 G060165320    California     060      costa mesa city  1970
## 3:         40 G060690000    California     060       santa ana city  1970
## 4:         57 G120240000       Florida     120 fort lauderdale city  1970
## 5:        108 G250633050 Massachusetts     250      southbridge cdp  1970
## 6:        109 G250760300 Massachusetts     250       westfield city  1970
```

```
##     GISJOIN DECADE  NHGISNAM NHGISST_2 NHGISCTY ICPSRST ICPSRCTY  ICPSRNAM
##      <char>  <num>    <char>    <char>   <char>   <num>    <num>    <char>
## 1: G2000780   1970    Butler       200     0150      32      150    BUTLER
## 2: G0600625   1970    Orange       060     0590      71      590    ORANGE
## 3: G0602570   1970    Orange       060     0590      71      590    ORANGE
## 4: G1200645   1970   Broward       120     0110      43      110   BROWARD
## 5: G2503980   1970 Worcester       250     0270       3      270 WORCESTER
## 6: G2504690   1970   Hampden       250     0130       3      130   HAMPDEN
##         STATENAM ICPSRSTI ICPSRCTYI ICPSRFIP STATE_2 COUNTY   PID X_CENTROID
##           <char>    <num>     <num>    <num>  <char> <char> <num>      <num>
## 1:        Kansas       32       150        0     200   0150  1007   -73203.1
## 2:    California       71       590        0     060   0590  2907 -1985465.2
## 3:    California       71       590        0     060   0590  2907 -1985465.2
## 4:       Florida       43       110        0     120   0110   285  1557532.5
## 5: Massachusetts        3       270        0     250   0270  2769  1950999.8
## 6: Massachusetts        3       130        0     250   0130  2762  1899443.2
##     Y_CENTROID GISJOIN_2 GISJOIN2 SHAPE_AREA SHAPE_LEN statefips countyfips
##          <num>    <char>   <char>      <num>     <num>    <char>     <char>
## 1:    31813.45  G2000150  2000150 3746230669  246761.8        20        015
## 2:  -196678.07  G0600590  0600590 2067710610  234799.8        06        059
## 3:  -196678.07  G0600590  0600590 2067710610  234799.8        06        059
## 4: -1134960.10  G1200110  1200110 3162680337  247973.3        12        011
## 5:   792018.33  G2500270  2500270 4089604891  356649.8        25        027
## 6:   753974.26  G2500130  2500130 1642345438  274455.1        25        013
##      fips smsacode statefips_ countyfi_1 entityfips                  name_2
##    <char>   <char>     <char>     <char>     <lgcl>                  <char>
## 1:  20015     9040         20        015         NA            Butler County
## 2:  06059     0360         06        059         NA            Orange County
## 3:  06059     0360         06        059         NA            Orange County
## 4:  12011     2680         12        011         NA           Broward County
## 5:  25027     2600         25        027         NA   Worcester County (pt.)
## 6:  25013     8000         25        013         NA Hampden County, MA (pt.)
##                                    namemsa fips_2           NAME
##                                     <char> <char>          <char>
## 1:                          Wichita, KS SMSA  20015       el dorado
## 2:  Anaheim-Santa Ana-Garden Grove, CA SMSA  06059      costa mesa
## 3:  Anaheim-Santa Ana-Garden Grove, CA SMSA  06059       santa ana
## 4:       Fort Lauderdale-Hollywood, FL SMSA  12011 fort lauderdale
## 5:            Fitchburg-Leominster, MA SMSA  25027     southbridge
## 6: Springfield-Chicopee-Holyoke, MA-CT SMSA  25013       westfield
##    cityname_constant fixed_cnty_number cnty90         state   tier
##                <char>            <char>  <num>        <char> <char>
## 1:         el dorado             20015  20015        KANSAS    all
## 2:        costa mesa             06059   6059    CALIFORNIA    all
## 3:         santa ana             06059   6059    CALIFORNIA    all
## 4:   fort lauderdale             12011  12011       FLORIDA    all
## 5:       southbridge             25027  25027 MASSACHUSETTS    all
## 6:         westfield             25013  25013 MASSACHUSETTS    all
```

This gives us a list of 289 matched cities and census places. This is around a tenth of size of the original Gentzkow dataset and a fifth of the size of the initial fuzzy match merge. I'm wondering if there's some issue with the county filtering in that it's a bit too strict, given that ~1000 matches were dropped because the county numbers didn't match.

We can try filtering by state to see if this relaxes restrictions a bit. Again, we must set both datasets's state

column to lowercase to deal with case sensitivity.

```
cities$state <- tolower(cities$state)
census_places$STATE <- tolower(census_places$STATE)

state_merge <- merge_plus(data1 = census_places,
           data2 = cities,
           by.x = "NAME",
           by.y = "cityname_constant", match_type = "fuzzy",
           unique_key_1 = "NHGISPLACE", unique_key_2 = "citypermid") # merge again with changes to data
state_result <- state_merge$matches

filtered_state_result <- filter(state_result, state == STATE) # create dataset with matching states
head(filtered_state_result)
```

```
## Key: <citypermid>
##    citypermid NHGISPLACE         STATE NHGISST                PLACE  YEAR
##         <num>     <char>        <char>  <char>               <char> <num>
## 1:          4 G200200750        kansas     200       el dorado city  1970
## 2:         11 G420624160  pennsylvania     420    pottstown borough  1970
## 3:         39 G060165320    california     060      costa mesa city  1970
## 4:         40 G060690000    california     060       santa ana city  1970
## 5:         57 G120240000       florida     120 fort lauderdale city  1970
## 6:        108 G250633050 massachusetts     250      southbridge cdp  1970
##     GISJOIN DECADE  NHGISNAM NHGISST_2 NHGISCTY ICPSRST ICPSRCTY  ICPSRNAM
##      <char>  <num>    <char>    <char>   <char>   <num>    <num>    <char>
## 1: G2000780   1970    Butler       200     0150      32      150    BUTLER
## 2: G4207342   1970   Chester       420     0290      14      290   CHESTER
## 3: G0600625   1970    Orange       060     0590      71      590    ORANGE
## 4: G0602570   1970    Orange       060     0590      71      590    ORANGE
## 5: G1200645   1970   Broward       120     0110      43      110   BROWARD
## 6: G2503980   1970 Worcester       250     0270       3      270 WORCESTER
##         STATENAM ICPSRSTI ICPSRCTYI ICPSRFIP STATE_2 COUNTY   PID X_CENTROID
##           <char>    <num>     <num>    <num>  <char> <char> <num>      <num>
## 1:        Kansas       32       150        0     200   0150  1007   -73203.1
## 2:  Pennsylvania       14       290        0     420   0290  1477  1701418.1
## 3:    California       71       590        0     060   0590  2907 -1985465.2
## 4:    California       71       590        0     060   0590  2907 -1985465.2
## 5:       Florida       43       110        0     120   0110   285  1557532.5
## 6: Massachusetts        3       270        0     250   0270  2769  1950999.8
##     Y_CENTROID GISJOIN_2 GISJOIN2 SHAPE_AREA SHAPE_LEN statefips countyfips
##          <num>    <char>   <char>      <num>     <num>    <char>     <char>
## 1:    31813.45  G2000150  2000150 3746230669  246761.8        20        015
## 2:   459142.37  G4200290  4200290 1967886858  250279.4        42        029
## 3:  -196678.07  G0600590  0600590 2067710610  234799.8        06        059
## 4:  -196678.07  G0600590  0600590 2067710610  234799.8        06        059
## 5: -1134960.10  G1200110  1200110 3162680337  247973.3        12        011
## 6:   792018.33  G2500270  2500270 4089604891  356649.8        25        027
##      fips smsacode statefips_ countyfi_1 entityfips                name_2
##    <char>   <char>     <char>     <char>     <lgcl>                <char>
## 1:  20015     9040         20        015         NA          Butler County
## 2:  42029     6160         42        029         NA    Chester  County, PA
## 3:  06059     0360         06        059         NA          Orange County
## 4:  06059     0360         06        059         NA          Orange County
## 5:  12011     2680         12        011         NA         Broward County
```

```
## 6:  25027     2600         25        027         NA Worcester County (pt.)
##                                    namemsa fips_2            NAME
##                                     <char> <char>          <char>
## 1:                        Wichita, KS SMSA  20015       el dorado
## 2:                Philadelphia, PA-NJ SMSA  42029       pottstown
## 3: Anaheim-Santa Ana-Garden Grove, CA SMSA  06059      costa mesa
## 4: Anaheim-Santa Ana-Garden Grove, CA SMSA  06059       santa ana
## 5:      Fort Lauderdale-Hollywood, FL SMSA  12011 fort lauderdale
## 6:           Fitchburg-Leominster, MA SMSA  25027     southbridge
##    cityname_constant fixed_cnty_number cnty90         state   tier
##                <char>            <char>  <num>        <char> <char>
## 1:         el dorado             20015  20015        kansas    all
## 2:         pottstown             42091  42091  pennsylvania    all
## 3:        costa mesa             06059   6059    california    all
## 4:         santa ana             06059   6059    california    all
## 5:   fort lauderdale             12011  12011       florida    all
## 6:       southbridge             25027  25027 massachusetts    all
```

```
state_merge_residue <- filter(state_result, state != STATE) # preserve data without matching states for
```

This returns around 540 matched cities and census places, filtering by state name.

Analysis

False matches

We can make it a little easier to look at this dataset by extracting only the most relevant columns.

```
extracted_state_result <- filtered_state_result[,c("STATE", "PLACE", "namemsa", "NAME", "cityname_consta
```

```
filter(extracted_state_result, cityname_constant != NAME)
```

```
##         STATE             PLACE                  namemsa         NAME
##        <char>            <char>                   <char>       <char>
## 1:    indiana    edinburgh town   Indianapolis, IN SMSA    edinburgh
## 2: california    los altos city       San Jose, CA SMSA    los altos
## 3:   missouri lee's summit city Kansas City, MO-KS SMSA lee's summit
##    cityname_constant      state
##                <char>     <char>
## 1:          edinburg    indiana
## 2:         los gatos california
## 3:       lees summit   missouri
```

Filtering this dataset for place names that do not match yields only these three results. This should mean that the only entries with different city names are these three, of which the Los Altos/Los Gatos match seems to be the only false association.

Thus the only possible remaining false matches will be cities with the same name in the same state, which is hopefully only a handful. (I tried doing some research on this but I could only find various Reddit/Quora threads; there wasn't a definitive answer on how many of these places exist.)

Unmatched cities

This is where it gets a bit more tricky. First, we can look at the places that weren't matched at all in the fuzzy match: these seem to roughly fall into two categories.

- Places like North Attleboro, MA – there are a few cities near this in the unmatched Census data (same county) but no exact match. If we want to match these places, we could probably do this by looking at county number, but this might be dubious because they are ultimately different cities.
- Places like Dubois, PA – there's nothing in this county in the Census data: presumably just not in the dataset.

There also seem to be some larger cities that are not getting matched because of differences in which they are labelled: for instance, "south chicago" exists in the unmatched_cities dataset and "south chicago heights" exists in the unmatched_census dataset. It might be prudent to go through some of the major cities to see if there are any other examples like this.

In general, major cities seem to be the biggest issue with this system of merging. I tried to look for New York to see how it was merged in this process, but I couldn't find the original cities entry for "manhattan" in either the unmatched cities dataset or the matched dataset. This led me to believe that it had been incorrectly matched (say, with Manhattan, Kansas) and subsequently filtered out, so I created a dataset (state_merge_residue) to keep track of the cities that were matched and then filtered out because the state columns were different. But I couldn't find the "manhattan" entry here either. I'm not sure where exactly this particular entry got removed, but it's definitely possible that something similar has happened to other major cities.

Manual Matching

We can try to bring the major cities that have been dropped back into this merge manually.

Creating a dataframe to store our manual matches:

```
manual_matches <- head(cities, 0)
manual_matches <- cbind(manual_matches, head(census_places, 0))
# colnames(manual_matches) <- c(paste0("cities_", colnames(cities)), paste0("census_places_", colnames(
colnames(manual_matches) <- c(paste0(colnames(cities)), paste0(colnames(census_places)))
```

Let's first try matching Manhattan-New York manually:

```
# Manhattan
match_nyc_city <- cities[cities$citypermid == 148, ]
match_nyc_census <- census_places[census_places$PLACE == "new york city", ]
manual_matches <- rbind(manual_matches, c(match_nyc_city, match_nyc_census))
```

This is kind of tedious, we can write a function to speed it up a bit:

```
new_manual_match <- function(manual_matches, id_cities, id_census_places) {
  match_cities <- cities[cities$citypermid == id_cities, ]
  match_census <- census_places[census_places$PLACE == id_census_places, ]
  new_match <- c(match_cities, match_census)
  manual_matches <- rbind(manual_matches, new_match)
  return(manual_matches)
}
```

Let's test the function:

```
# Test with Bronx-New York match
manual_matches <- new_manual_match(manual_matches, 1944, "new york city")
```

Now we can use a hashmap to store all of our manual matches and just pass that to the function. That way if we want to add a new match we can just add it to the hashmap.

```
manual_matches_map <- list(
  "queens" = list(permid = 2042, place= "new york city"),
  "brooklyn" = list(permid = 394, place= "new york city"),
```

```
  "staten island" = list(permid = 1647, place = "new york city"),
  "south chicago heights" = list(permid = 766, place = "south chicago heights village"),
  "austin texas" = list(permid = 549, place = "austin city"),
  "columbus ohio" = list(permid = 386, place = "columbus city"),
  "indianapolis" = list(permid = 319, place = "indianapolis city (remainder)"), # what does remainder m
  "nashville" = list(permid = 47037, place = "nashville-davidson metropolitan government (balance)"),
  "las vegas" = list(permid = 1839, place = "las vegas city"),
  "mesa" = list(permid = 20, place = "mesa city"),
  "miami" = list(permid = 62, place = "miami city"), #census data contains both miami and miami beach
  "virginia beach" = list(permid = 215, place = "virginia beach city") #include norfolk as well?

)
```

Iterate over this hashmap and add each entry to our manual_matches dataframe.

```
for (i in manual_matches_map) {
  id_cities <- i$permid
  id_census_places <- i$place
  match_cities <- cities[cities$citypermid == id_cities, ]
  match_census <- census_places[census_places$PLACE == id_census_places, ]
  new_match <- c(match_cities, match_census)
  manual_matches <- rbind(manual_matches, new_match)
}
```

Some places can't be identified by their census place name because there are multiple entries for one place name. Let's fix these manually:

```
manual_matches <- rbind(manual_matches, c(cities[cities$citypermid == 533, ], census_places[census_place
manual_matches <- rbind(manual_matches, c(cities[cities$citypermid == 2023, ], census_places[census_pla
```

Now we have a dataframe complete with our manual matches that can easily be added to and appended to our main dataframe.