Querying Census Data

Enoch Shin, Felix Stetsenko

December 07, 2019

Contents

Purpose

This file's purpose is to query the census data, particularly the 2017 American Community Survey data using the `censusapi` package in order to get income and race-related variables on a census tract level.

Once the data has been queried, we will join it with the shapefiles for Cook County.

Ingest (Query)

We looked at the Census Bureau's website (https://www.census.gov/data/developers/data-sets/acs-5year.html) for example API calls for the ACS data.

To reference the variable names that we want, we referenced the codebook from the Bureau: (https://api.census.gov/data/2017/acs/acs5/variables.html). More examples of data provenance to follow.

Set API Key for Census Bureau

You'll have to request a key here: https://api.census.gov/data/key_signup.html.

```
Sys.setenv(CENSUS_KEY="XXXXX")
readRenviron("~/.Renviron")
Sys.getenv("CENSUS_KEY")
```

Query available variables and datasets

```
apis <- listCensusApis()
# list of all possible API soruces
apis <- listCensusApis()
head(apis)
```

```
##                           title     name vintage
## 76  ACS 1-Year Detailed Tables acs/acs1    2010
## 91  ACS 1-Year Detailed Tables acs/acs1    2011
## 102 ACS 1-Year Detailed Tables acs/acs1    2012
## 185 ACS 1-Year Detailed Tables acs/acs1    2013
## 208 ACS 1-Year Detailed Tables acs/acs1    2014
## 249 ACS 1-Year Detailed Tables acs/acs1    2015
##                                          url isTimeseries     temporal
## 76  https://api.census.gov/data/2010/acs/acs1           NA unidentified
## 91  https://api.census.gov/data/2011/acs/acs1           NA unidentified
## 102 https://api.census.gov/data/2012/acs/acs1           NA unidentified
## 185 https://api.census.gov/data/2013/acs/acs1           NA unidentified
## 208 https://api.census.gov/data/2014/acs/acs1           NA unidentified
## 249 https://api.census.gov/data/2015/acs/acs1           NA unidentified
##
## 76  The American Community Survey (ACS) is a nationwide survey designed to provide communities a fres
```

```
## 91  The American Community Survey (ACS) is a nationwide survey designed to provide communities a fres
## 102 The American Community Survey (ACS) is a nationwide survey designed to provide communities a fres
## 185 The American Community Survey (ACS) is a nationwide survey designed to provide communities a fres
## 208
## 249
##                   modified
## 76  2018-07-04 00:00:00.0
## 91  2018-07-04 00:00:00.0
## 102 2018-07-04 00:00:00.0
## 185 2018-07-04 00:00:00.0
## 208 2018-07-05 00:00:00.0
## 249 2018-07-05 00:00:00.0
```

```
# https://api.census.gov/data/2017/acs/acs5/variables.html

# pull the codebook
varii <- getCensus(name = "2017/acs/acs5/variables", vars="*")
str(varii)
```

```
## 'data.frame':    25110 obs. of  3 variables:
##  $ name   : chr  "for" "in" "ucgid" "B24022_060E" ...
##  $ label  : chr  "Census API FIPS 'for' clause" "Census API FIPS 'in' clause" "Uniform Census Geograp
##  $ concept: chr  "Census API Geography Specification" "Census API Geography Specification" "Census AP
```

Find correct variables for race data

Let's narrow down the available variables in the the ACS 2017 5-year census. The structure of the codebook shows us that we have a variable name, label, and a concept describing that variable. First, we'll search for race-related variables.

```
# find the possible race-related variables to query
racedata <- varii[grep("RACE", varii$label, ignore.case = TRUE), ]
```

Sift through the list of matches above to get the desired variable. We wish to get the population of *one* race in each tract. Thus, we know we'll need an estimate of a total population. We'll do another search for this metric:

```
# racedata[which(racedata$name=="C02003_016E"),] #test case
onerace <- grep("Estimate!!Total!!Population of one race!!", racedata$label)
onerace_data <- racedata[onerace, ]
head(onerace_data)
```

```
##              name
## 11630 C02003_008E
## 11794 C02003_004E
## 11798 C02003_003E
## 11799 C02003_007E
## 11803 C02003_006E
## 11807 C02003_005E
##                                                                                label
## 11630                          Estimate!!Total!!Population of one race!!Some other race
```

```
## 11794                  Estimate!!Total!!Population of one race!!Black or African American
## 11798                                        Estimate!!Total!!Population of one race!!White
## 11799 Estimate!!Total!!Population of one race!!Native Hawaiian and Other Pacific Islander
## 11803                                    Estimate!!Total!!Population of one race!!Asian alone
## 11807          Estimate!!Total!!Population of one race!!American Indian and Alaska Native
##             concept
## 11630 DETAILED RACE
## 11794 DETAILED RACE
## 11798 DETAILED RACE
## 11799 DETAILED RACE
## 11803 DETAILED RACE
## 11807 DETAILED RACE
```

```
# View(onerace_data)
```

We can see that the census describes its population measures like so: "Estimate!!Total!!Population of one race!!Black or African American". Now, let's grab the variable name in the census that corresponds to these values, and also extract the race name from the long ""Estimate!!Total!!Population of one race!!Black or African American" label:

```
#get census variable name for value
codes_onerace <- racedata[onerace, 1]
codes_onerace
```

```
## [1] "C02003_008E" "C02003_004E" "C02003_003E" "C02003_007E" "C02003_006E"
## [6] "C02003_005E"
```

```
# we want to grab just the race name from the
## "Estimate!!Total!!Population of one race!!Black or African American"
## codes
everyfourth <- seq(from=4, to=24, by=4)

# split the label into respective parts, delineated by !!
## grab the fourth element of every split label,
### which is the race name
race_decode <- unlist(strsplit(racedata[onerace, 2], "\\!!"))[everyfourth]
race_decode
```

```
## [1] "Some other race"
## [2] "Black or African American"
## [3] "White"
## [4] "Native Hawaiian and Other Pacific Islander"
## [5] "Asian alone"
## [6] "American Indian and Alaska Native"
```

```
#make a key for the census codes and the race names
racekey <- data.frame(code=codes_onerace, race=as.character(race_decode))
```

Query race data

Once we've gotten the census variable names for the race population estimates, pull the data from Cook County on a tract level:

```
racedf <- getCensus(name = "acs/acs5",
                    vintage=2017,
                    vars=c(codes_onerace),
                    region = "tract:*",
                    regionin = "state:17+county:031")

#wrangling/renaming:
colnames(racedf) <- c("state", "county", "tract",as.character(racekey$race))
# write_feather(racedf, "ChicagoCommute/feather/raceByTract.feather")

# we want the proportions of race, not counts
rowsums <- racedf %>%
          mutate(totalTractPop = rowSums(.[, 4:9], na.rm=TRUE))

racedf_prop <- rowsums %>%
                mutate_at(4:9, .funs = function(x){x/.$totalTractPop}) %>%
                mutate_at(vars(4:9), ~replace(., is.na(.), 0))

#write into the ShinyApp directory and the project directory
write_feather(racedf_prop, "ChicagoCommute/feather/raceByTract_proportion.feather")
write_feather(racedf_prop, "ChicagoCommute/app/feather/raceByTract_proportion.feather")
```

Income on Tract Basis

Now, we're going to do the same thing except for income metrics. To find the right variables, our path was as follows:

(1) Go to the developer documentation for the ACS 5-year data: https://www.census.gov/data/developers/data-sets/acs-5year.html

(2) Under 2017, there are several tables available to us. A good overview of the types of tables available are found in the "Examples and Supported Geography" link used under "Detailed Tables" (https://api.census.gov/data/2017/acs/acs5.html).

(3) For race, we used regular expressions to find the variables. For income, things are a little more complicated. We got somewhat lucky and picked the right table under the "Examples and Supported Geography" list: the table "ACS 5-Year Data Profiles" is what we want. The variable list "https://api.census.gov/data/2017/acs/acs5/profile/variables/" was then manually searched for the right income variables.

```
income_codes <- c("DP03_0062E", "DP03_0088E", "DP03_0092E", "DP03_0128PE")

colnames <- c("state", "county", "tract", "Median household income", "Per capita income",
              "Median earnings for workers", "Pct under poverty level")

income_df <- getCensus(name = "acs/acs5/profile",
                    vintage=2017,
                    vars= income_codes,
                    region = "tract:*",
                    regionin = "state:17+county:031")
```

```r
colnames(income_df) <- colnames

# saveRDS(income_df, file="ChicagoCommute/RDA/incomeByTract.Rda")
```

Final Wrangling and add Shapefiles

Here, we'll do some wrangling of the census data, and then we'll join the census data with shapefiles and save those so that the app can access them right away.

```r
#recode the income, which had indicator values like -6666666 that
##should be changed to 0
income <-  income_df %>%
              mutate_at(c("Median household income",
                          "Per capita income",
                          "Median earnings for workers",
                          "Pct under poverty level"),
                        function(x){ifelse(x < 0, 0, x)})
# write into the app's folder
write_feather(income, path="ChicagoCommute/app/feather/incomeByTract.feather")
# write into the feather folder
write_feather(income, path="ChicagoCommute/feather/incomeByTract.feather")
```

Make shapefiles with census data

```r
shapefile <- tigris::tracts(state = "17", county = "031")

# shapefile@data$NAME

shapefile_race <- geo_join(shapefile, racedf_prop,"TRACTCE", "tract", how="left")
shapefile_income <- geo_join(shapefile, income,"TRACTCE", "tract", how="left")

#can't use feather since it can't handle shapefile
saveRDS(shapefile_race, "ChicagoCommute/app/RDA/shapefile_race.Rda")
saveRDS(shapefile_income, "ChicagoCommute/app/RDA/shapefile_income.Rda")
saveRDS(shapefile_race, "ChicagoCommute/RDA/shapefile_race.Rda")
saveRDS(shapefile_income, "ChicagoCommute/RDA/shapefile_income.Rda")
```