

get_census_data

The Census Bureau has built an API infrastructure for developers that allow for semi-automatic access to their data products. Each product is organized by a certain geographic area and a reference year. Furthermore, each product can have a sub-product with multiple data tables. In this exercise we are interested in building a function that will pull data from the Census Bureau API's in a semi-automatic manner following the Census Data API User Guide.

Example

Data from Puerto Rico is identified by the code 72 in the geography level. In the following example we go through a process of pulling data for a specific product. In particular, we pull data from the 2021 ACS 1-year subject table for a set of population estimates variables.

```
#### Pulling Puerto Rico Census Data from APIs ####

## We need a Census Key that allows for access. This key is associated to the
## author's information.
source("census_key.R")

## 1. Identify the dataset we want to use
## This link has all of the datasets available https://api.census.gov/data.html

## For example, let's look at the latest 1-year ACS subject tables
### Here we have a product (acs), a sub-product (acs1), and a data table (subject).

url_base <- "http://api.census.gov/data/2021/acs/acs1/subject"

### 2. Identify which variables we want, and the geography granularity

# https://api.census.gov/data/2021/acs/acs1/subject/geography.html
geography_level <- "060" ### state and counties

# https://api.census.gov/data/2021/acs/acs1/subject/variables.html
variables <- paste0(c("NAME", "COUNTY", sprintf("S0101_C01_%03dE", 1:20)), collapse=",")

## In this case the year is 2021.

url <- paste0("https://api.census.gov/data/2021/acs/acs1/subject?get=",
              variables, "&for=county:*&in=state:72&key=", census_key)

data_census <- GET(url)

data_census <- fromJSON(rawToChar(data_census$content)) %>% row_to_names(1)

head(data_census)
```

```
##      NAME                                COUNTY S0101_C01_001E S0101_C01_002E
## [1,] "Carolina Municipio, Puerto Rico" "031" "152993" "4708"
## [2,] "Arecibo Municipio, Puerto Rico" "013" "87053" "2861"
## [3,] "Bayamón Municipio, Puerto Rico" "021" "182673" "6013"
## [4,] "Caguas Municipio, Puerto Rico" "025" "126756" "3967"
## [5,] "Guaynabo Municipio, Puerto Rico" "061" "89195" "2645"
## [6,] "Mayagüez Municipio, Puerto Rico" "097" "71939" "2166"
##      S0101_C01_003E S0101_C01_004E S0101_C01_005E S0101_C01_006E S0101_C01_007E
## [1,] "8052" "6388" "8917" "10320" "10974"
## [2,] "3933" "4473" "4990" "5964" "6042"
## [3,] "8007" "8757" "10366" "12585" "13712"
## [4,] "6082" "6262" "8069" "8115" "8849"
## [5,] "3428" "3967" "5037" "4973" "6300"
## [6,] "3683" "2872" "5599" "8105" "4416"
##      S0101_C01_008E S0101_C01_009E S0101_C01_010E S0101_C01_011E S0101_C01_012E
## [1,] "9657" "9361" "8425" "9688" "10160"
## [2,] "3908" "7083" "4184" "5550" "6004"
## [3,] "12360" "11066" "10536" "10357" "10940"
## [4,] "8254" "7065" "8600" "8346" "8445"
## [5,] "5834" "4947" "5732" "5318" "5879"
## [6,] "3827" "3326" "3444" "3488" "3709"
##      S0101_C01_013E S0101_C01_014E S0101_C01_015E S0101_C01_016E S0101_C01_017E
## [1,] "11114" "8308" "8582" "8729" "9076"
## [2,] "6521" "5159" "5102" "4967" "4411"
## [3,] "11781" "12753" "11243" "9486" "9678"
## [4,] "7198" "9619" "8606" "5512" "5339"
## [5,] "5523" "7414" "5771" "5534" "3049"
## [6,] "3666" "4915" "5057" "3754" "3637"
##      S0101_C01_018E S0101_C01_019E S0101_C01_020E state county
## [1,] "5499" "5035" "14440" "72" "031"
## [2,] "3022" "2879" "8406" "72" "013"
## [3,] "6398" "6635" "16764" "72" "021"
## [4,] "3740" "4688" "12344" "72" "025"
## [5,] "3602" "4242" "7395" "72" "061"
## [6,] "3111" "3164" "6555" "72" "097"
```

As can be seen, we obtain a dataset of estimates, but it is difficult to identify which variables were pulled. The Census website organizes the variables available for each data set. Furthermore, observe that the variables begin with S0101. It is also an option to call a group of variables; however, these groups are already built by the Census Bureau. To specify the variables, we can call the description for each of the variables and do some wrangling to obtain a well-documented dataset.

```
variables_url <- "https://api.census.gov/data/2021/acs/acs1/subject/variables"
variables_information <- GET(variables_url)
variables_information <- fromJSON(rawToChar(variables_information$content)) %>%
  row_to_names(1) %>%
  as.data.frame() %>%
  filter(name %in% colnames(data_census))

variables_information
```

```
##      name
## 1 S0101_C01_004E
```

```

## 2 S0101_C01_005E
## 3 S0101_C01_002E
## 4 S0101_C01_003E
## 5 S0101_C01_001E
## 6 S0101_C01_008E
## 7 S0101_C01_009E
## 8 S0101_C01_006E
## 9 S0101_C01_007E
## 10 S0101_C01_020E
## 11 S0101_C01_016E
## 12 S0101_C01_017E
## 13 S0101_C01_014E
## 14 S0101_C01_015E
## 15 S0101_C01_012E
## 16 S0101_C01_013E
## 17 S0101_C01_010E
## 18 S0101_C01_011E
## 19 S0101_C01_018E
## 20 S0101_C01_019E
## 21 COUNTY
##
## label
## 1 Estimate!!Total!!Total population!!AGE!!10 to 14 years
## 2 Estimate!!Total!!Total population!!AGE!!15 to 19 years
## 3 Estimate!!Total!!Total population!!AGE!!Under 5 years
## 4 Estimate!!Total!!Total population!!AGE!!5 to 9 years
## 5 Estimate!!Total!!Total population
## 6 Estimate!!Total!!Total population!!AGE!!30 to 34 years
## 7 Estimate!!Total!!Total population!!AGE!!35 to 39 years
## 8 Estimate!!Total!!Total population!!AGE!!20 to 24 years
## 9 Estimate!!Total!!Total population!!AGE!!25 to 29 years
## 10 Estimate!!Total!!Total population!!SELECTED AGE CATEGORIES!!5 to 14 years
## 11 Estimate!!Total!!Total population!!AGE!!70 to 74 years
## 12 Estimate!!Total!!Total population!!AGE!!75 to 79 years
## 13 Estimate!!Total!!Total population!!AGE!!60 to 64 years
## 14 Estimate!!Total!!Total population!!AGE!!65 to 69 years
## 15 Estimate!!Total!!Total population!!AGE!!50 to 54 years
## 16 Estimate!!Total!!Total population!!AGE!!55 to 59 years
## 17 Estimate!!Total!!Total population!!AGE!!40 to 44 years
## 18 Estimate!!Total!!Total population!!AGE!!45 to 49 years
## 19 Estimate!!Total!!Total population!!AGE!!80 to 84 years
## 20 Estimate!!Total!!Total population!!AGE!!85 years and over
## 21 Geography
## concept
## 1 AGE AND SEX
## 2 AGE AND SEX
## 3 AGE AND SEX
## 4 AGE AND SEX
## 5 AGE AND SEX
## 6 AGE AND SEX
## 7 AGE AND SEX
## 8 AGE AND SEX
## 9 AGE AND SEX
## 10 AGE AND SEX
## 11 AGE AND SEX

```

```
## 12 AGE AND SEX
## 13 AGE AND SEX
## 14 AGE AND SEX
## 15 AGE AND SEX
## 16 AGE AND SEX
## 17 AGE AND SEX
## 18 AGE AND SEX
## 19 AGE AND SEX
## 20 AGE AND SEX
## 21          <NA>
```

Building a simple function

We now generalize the previous example for the ACS (acs) and the Population Estimates (pep) products.

Now let's work on generalizing

```
get_census_data <- function(product, subproduct, year , variables, group = F,
                             table_type = NULL, census_key) {
  ## product <- acs1 ; acs3, acs5,
  ### product details here: https://api.census.gov/data.html
  ## acs1 only provides estimates for counties where population >= 65,000
  ## acs5 https://www.census.gov/data/developers/data-sets/acs-5year.html
  library(httr)
  library(jsonlite)
  library(janitor)
  library(tidyverse)
  if (group){
    url_base <- ifelse(is.null(table_type), paste0("http://api.census.gov/data/",
                                                    year, "/", product, "/", subproduct),
                       paste0("http://api.census.gov/data/", year, "/", product, "/",
                               subproduct, "/", table_type))
    url <- ifelse(product == "pep", paste0(url_base, "?get=", variables, "&for=state:72&key=",
                                           census_key),
                 paste0(url_base, "?get=group(", variables, "&for=county:*&in=state:72&key=",
                           census_key))
  } else {
    variables <- ifelse(product=="pep", paste0(c("NAME" ,paste(variables)), collapse=","),
                       paste0(c("NAME", "COUNTY" ,paste(variables)), collapse=","))
    url_base <- ifelse(is.null(table_type), paste0("http://api.census.gov/data/",
                                                    year, "/", product, "/", subproduct),
                       paste0("http://api.census.gov/data/", year, "/", product, "/",
                               subproduct, "/", table_type))
    url <- ifelse(product == "pep", paste0(url_base, "?get=", variables, "&for=state:72&key=",
                                           census_key),
                 paste0(url_base, "?get=", variables, "&for=county:*&in=state:72&key=",
                           census_key))
  }

  data_census <- GET(url)
  data_census <- fromJSON(rawToChar(data_census$content), flatten=T) %>%
    as.data.frame() %>%
    row_to_names(1)
```

```

variables_url <- paste0(url_base, "/variables")
variables_information <- GET(variables_url)
variables_information <- fromJSON(rawToChar(variables_information$content)) %>%
  row_to_names(1) %>%
  as.data.frame() %>%
  filter(name %in% colnames(data_census))
lista_censo <- list(data_census, variables_information)
return(lista_censo) ### returns a list with two datasets -> variables information and data
}

```

Does it work? Let us try the function to obtain data from ACS 5-year for 2021 and the group of variables denoted by B01001. We obtain data estimates for each municipality. Observe that the variables have a suffix E, M, etc., which denotes the type of measure. Those ending with E are the estimates. The rest of the variables can be used to assess the quality of the estimate. More details [here](#).

```

acs_municipality <- get_census_data(product = "acs",
                                   subproduct = "acs5",
                                   year = "2021",
                                   variables = "B01001_001E",
                                   group = F,
                                   table_type = NULL,
                                   census_key = census_key)

head(acs_municipality[[1]])

```

##		NAME	COUNTY	B01001_001E	state	county
## 2		Adjuntas Municipio, Puerto Rico	001	18068	72	001
## 3		Aguada Municipio, Puerto Rico	003	38307	72	003
## 4		Aguadilla Municipio, Puerto Rico	005	55241	72	005
## 5		Aguas Buenas Municipio, Puerto Rico	007	24567	72	007
## 6		Aibonito Municipio, Puerto Rico	009	24565	72	009
## 7		Añasco Municipio, Puerto Rico	011	25859	72	011

Now we obtain population estimates by single year of age for 2019.

```

pop_est_SYA <- get_census_data(product = "pep",
                               subproduct = "charage",
                               year = "2019",
                               variables = c("AGE", "POP"),
                               group = F,
                               table_type = NULL,
                               census_key = census_key)

head(pop_est_SYA[[1]])

```

##		NAME	AGE	POP	state
## 2		Puerto Rico	0	21386	72
## 3		Puerto Rico	1	20596	72
## 4		Puerto Rico	2	22865	72
## 5		Puerto Rico	3	25461	72
## 6		Puerto Rico	4	27174	72
## 7		Puerto Rico	5	28973	72

Challenges and possible solutions

The most significant challenge is that products that are of main interest in our research might not necessarily be available through the API structure that the Census Bureau has built. For example, the population estimates for single year of age are available up to 2019 and the municipality population estimates are only available through the 5 year ACS. As specified in the Census website, the 1 and 3 year ACS are available for counties for which the population is 65,000 or higher, which is not the case for most PR municipalities. Furthermore, selecting variables is a difficult task due to the massive amount of variables available. There are many possible api calls one can create, and given the challenges mentioned, we explore the `tidycensus` package and see how we can modify their functions to accommodate for the data of interest.

Tidycensus exploration

We try to obtain the same data we obtained with the function we wrote.

```
library(tidycensus)

census_api_key(census_key)

## To install your API key for use in future sessions, run this function with 'install = TRUE'.

### ACS 5 year

head(get_acs(geography = "county", variables = "B01001_001E", year = 2021, state = "puerto rico"))

## Getting data from the 2017-2021 5-year ACS

## # A tibble: 6 x 5
##   GEOID NAME                variable  estimate  moe
##   <chr> <chr>                <chr>    <dbl> <dbl>
## 1 72001 Adjuntas Municipio, Puerto Rico B01001_001 18068   NA
## 2 72003 Aguada Municipio, Puerto Rico  B01001_001 38307   NA
## 3 72005 Aguadilla Municipio, Puerto Rico B01001_001 55241   NA
## 4 72007 Aguas Buenas Municipio, Puerto Rico B01001_001 24567   NA
## 5 72009 Aibonito Municipio, Puerto Rico  B01001_001 24565   NA
## 6 72011 Añasco Municipio, Puerto Rico    B01001_001 25859   NA
```

For the population estimates we could not find the single year estimates. This is because the tidyverse function does not accommodate for the granularity we are seeking. We would need to modify this function to obtain the estimates we want. In the following example note that the age groups are denoted by integer numbers not to be confused with single year of age.

```
### Population estimates

get_estimates(geography = "county", product = "characteristics", breakdown = c("AGEGROUP"),
              year = 2019, state = "puerto rico", output = "tidy")

## # A tibble: 2,496 x 4
##   GEOID NAME                value AGEGROUP
##   <chr> <chr>                <dbl>    <dbl>
## 1 72009 Aibonito Municipio, Puerto Rico 22108      0
## 2 72009 Aibonito Municipio, Puerto Rico  863       1
```

##	3	72009	Aibonito Municipio, Puerto Rico	1084	2
##	4	72009	Aibonito Municipio, Puerto Rico	1239	3
##	5	72009	Aibonito Municipio, Puerto Rico	1284	4
##	6	72009	Aibonito Municipio, Puerto Rico	1325	5
##	7	72009	Aibonito Municipio, Puerto Rico	1435	6
##	8	72009	Aibonito Municipio, Puerto Rico	1334	7
##	9	72009	Aibonito Municipio, Puerto Rico	1215	8
##	10	72009	Aibonito Municipio, Puerto Rico	1227	9
##	#	... with 2,486 more rows			