Accessing Data with the Census Bureau API

Alex Shum

May 9, 2014

Abstract

The American Community Survey is an ongoing survey by the US Census Bureau that provides yearly demographic data. The Census Bureau has various tools for giving the public access to their data. Recently the Census Bureau released an online API to give the public access to the Decennial census and the American Community Census. We discuss how to find available datasets and available demographic variables as well as some of the shortcomings of the online API. The main shortcomings are related to how the data is organized and the limited number of variables accessible from the online API. We also provide examples on what is possible using the online API.

for all the tables: numbers should be aligned right.

1 Introduction

The United States Census Bureau has been conducting a decennial census since 1790. Originally this census was simply to count the population across the country. More recently the decennial census includes a short-form asking for name, sex, age, and a few other demographic variables. About one in six households also received a long-form of the census consisting of additional socioeconomic questions. After the 2000 decennial census many of the long-form questions were collected as part of a new survey: the American Community Survey (ACS).

The ACS is an ongoing yearly survey that collects additional demographic variables including but not limited to age, sex, race, income and education. Unlike the decennial census, the American Community Survey is distributed based on a random selection of addresses every year. Although the ACS is only sent to a sample of all US households, this data is meant to provide more up to date information than the Census Bureau's decennial census. Completion of both the decennial census and the American Community Survey are required by law (Title 18 U.S.C Section 3571 and Section 3559); however, it should be noted that the Census Bureau has not opted to prosecute anyone for failure to complete the decennial census or the ACS. Despite the lack of enforcement, the ACS still reports a response rate of 97%.

Both the decennial survey and the ACS data are used in part by federal, state and local agencies as basis for policy decisions and to allocate state funding. The Census Bureau has also released some of this data for public use. Many of the data sets are available directly in a compressed format from the Census Bureau's FTP site: `http://ftp2.census.gov/`. The Census Bureau has created numerous other tools to make it easier for the public to access their data. These tools include Easy Stats (`http://www.census.gov/easystats/`), QuickFacts (`http://quickfacts.census.gov/qfd/index.html`) and DataFerrett (`http://dataferrett.census.gov/`). Since 2012, the Census Bureau has also included an online developer's API in order to improve accessibility of the ACS and decennial census datasets. The Census Bureau's online API can be accessed online: `http://api.census.gov`.

We discuss how to access data from the Census Bureau's online developer's API and how the ACS data is structured. We also discuss what kind of variables are available and some limitations with the API. We base this discussion on a paper by Stangl, Rundel, and Morgan [5] as a starting point on some of the limitations of the API. This article explores some multivariate frequency distributions using data from the ACS dataset; however, there are some gaps in what we can access as well as inconsistencies in the database.

2 Requesting Data

To access data through the Census Bureau's online API we need to construct an HTTP GET request. A valid GET request is formed through a constructed web URL and includes a specific dataset, year, variable and geographic region of interest. The basic structure of an HTTP GET request for the decennial census and for the ACS is as follows:

`http://api.census.gov/data/[YEAR]/[DATASET]?key=[KEY]&get=[VARIABLES]&for=[GEOGRAPHY]`

The bracketed expressions in the above URL represent parameters that need to be specified depending on dataset, time frame, geographic region of interest and demographic variables. Each of the bracketed expressions is detailed in the following sections.

2.1 Key

`[KEY]` is an id code required to perform a valid GET request. A developer's key uniquely identifies anyone who requests data from the API. Requesting a key can be done by registering at `http://www.census.gov/developers/tos/key_request.html`.

2.2 Year and Dataset

`[YEAR]` and `[DATASET]` specify the dataset and year of the data. The available datasets include the decennial census and the ACS. ACS datasets are provided in form of 1-year, 3-year and 5-year aggregates. The `[YEAR]` variable for the ACS datasets indicates the final year in the aggregate. For example, the 2012 5-year ACS dataset is the ACS dataset that spans 2008-2012 and the 2012 3-year ACS dataset is the ACS dataset that spans 2010-2012. For the decennial census the `[YEAR]` indicates the year the census data was collected. The `[DATASET]` parameter is an abbreviation for the dataset assigned by the Census Bureau. For example, the ACS 5-year dataset is *acs*5 and the ACS 1-year dataset is *acs*1. Table 1 gives an overview of timeframes, datasets and the associated abbreviations assigned by the census bureau.

DATASET	YEAR	Description
sf1	1990, 2000, 2010	Decennial Census
acs5	2010, 2011, 2012	ACS 5-year
acs3	2012	ACS 3-year
acs1	2011, 2012	ACS 1-year

Table 1: Datasets and Years provided by the Census Bureau through the online API.

For the 2010 decennial census a valid HTTP GET request as follows (refer to the DATASET column in 1):

`http://api.census.gov/data/2010/sf1?key=[KEY]&get=[VARIABLES]&for=[GEOGRAPHY]`

Similarly requesting data from the 2011 ACS 3-year dataset requires the following HTTP GET request:

`http://api.census.gov/data/2011/acs3?key=[KEY]&get=[VARIABLES]&for=[GEOGRAPHY]`

2.3 Geography

`[GEOGRAPHY]` describes the geographic region of interest, and the `&for=[GEOGRAPHY]` tag is used to indicate this region. The geographic area can include the entire United States:

`http://api.census.gov/data/[YEAR]/[DATASET]?key=[KEY]&get=[VARIABLES]&for=us:*`

Alternatively, the geographic area of interest can be some or all states.

```
http://api.census.gov/data/[YEAR]/[DATASET]?key=[KEY]&get=[VARIABLES]&for=state:*
http://api.census.gov/data/[YEAR]/[DATASET]?key=[KEY]&get=[VARIABLES]&for=state:06
http://api.census.gov/data/[YEAR]/[DATASET]?key=[KEY]&get=[VARIABLES]&for=state:01,06
```

The above HTTP GET requests specify all states, a specific state (California, see figure 1), and a set of selected states (Alabama and California), respectively. Some geographic regions are nested within larger regions and a valid GET request may require us to specify the containing region. The `&in=[GEOGRAPHY]` tag is used to further specify an appropriate containing region for the region of interest. For example, counties are contained within states and the following statement is a GET request for a county or counties within a specific state:

```
http://api.census.gov/data/ [...] &for=county:037&in=state:06
http://api.census.gov/data/ [...] &for=county:*&in=state:06
```

The above HTTP GET requests Los Angeles County in California and all counties in California, respectively (see figure 2, note that we truncated above URL for display purposes). There are even smaller geographic regions and for some of these geographic entities multiple containing regions have to be specified for a valid GET request. For example, census tracts are contained within both counties and states. Here is how to specify a valid GET request to access data for a census tract or multiple census tracts within a county and state:

```
http://api.census.gov/data/ [...] &for=tract:*&in=state:06+county:037
http://api.census.gov/data/ [...] &for=tract:101110&in=state:06+county:037
```

The above HTTP GET requests all census tracts within Los Angeles County and census tract 1011.10 within Los Angeles County, respectively.

The Census Bureau has a very sophisticated system of hierarchy for geographic entities. At the top level of the ACS is the entire nation, followed by region, division, state, county, county-subdivision, tract, block group, place, congressional district, zip code area, school district and a few other geographic divisions. See table 2 for a table of geographic entities available on the census API for the 2012 ACS along with an example of the correct `[GEOGRAPHY` formatting for a valid GET request. The summary levels in table 2 are three-digit codes used internally by the Census Bureau to indicate different geographic regions or levels. Note that different datasets will have different geographies and thus different summary levels available; unfortunately there does not appear to be a master list of summary levels available to the public.

Summary Level	Description	[GEOGRAPHY]
010	us	&for=us:*
020	region	&for=region:*
030	division	&for=division:*
040	state	&for=state:*
050	state-county	&for=county:*
060	state-county-county subdivision	&for=county+subdivision:*&in=state:01+county:001
140	state-county-tract	&for=tract:*&in=state:01+county:001
150	state-county-tract-block group	&for=block+group:*&in=state:1+county:001+tract:020100
160	state-place	&for=place:*&in=state:01
250	american indian area/alaska native area hawaiian home land	&for=american+indian+area/alaska+native+area/hawaiian+home+land:*&in=state:01
310	metropolitan/micropolitan statistical area	&for=metropolitan+statistical+area/micropolitan+statistical+area:*
320	state-metropolitan/micropolitan statistical area	&for=metropolitan+statistical+area/micropolitan+statistical+area:*&in=state:01
330	combined statistical area	&for=combined+statistical+area:*
340	state-combined statistical area	&for=combined+statistical+area:*&in=state:01
350	new england city and town area	&for=new+england+city+and+town+area:*
400	urban area	&for=urban+area:*
500	state-congressional district	&for=congressional+district:*
510	state-congressional district-county	&for=county:*&in=state:01+congressional+district:01
610	state-state legislative district (upper chamber)	&for=state+legislative+district+(upper+chamber):*&in=state:01
620	state-state legislative district (lower chamber)	&for=state+legislative+district+(lower+chamber):*&in=state:01
795	state-public use microdata area	&for=public+use+microdata+area:*&in=state:01
860	zip code tabulation area	&for=zip+code+tabulation+area:*
950	state-school district (elementary)	&for=school+district+(elementary):*&in=state:01
960	state-school district (secondary)	&for=school+district+(secondary):*&in=state:01
970	state-school district (unified)	&for=school+district+(unified):*&in=state:01

Table 2: List of valid geographic regions for the 2012 ACS 5-year

From table 2, there is a specific hierarchy of geographic regions and specific valid combinations of geographic regions. Different ACS datasets have different geographic regions available and different requirements for geographic regions. For example, the 2010 decennial census requires that we specify a state for zip code tabulation areas. By contrast, the 2012 ACS 5-year dataset has zip code tabulation areas that do not require a containing state for a valid GET request. The 2010 ACS 5-year dataset simply does not have zip code tabulation areas available at all. In order to understand which geographic regions are available for datasets and which combination of geographic regions are valid we will discuss the Census Bureaus documentation scheme for each of their datasets in the section 3.1.

Finding available demographic variables and forming a valid GET request for `[VARIABLES]` requires a more detailed knowledge of how the census datasets are organized which variables are made available. We discuss finding demographic variables and their format and structure in section 3.2.

3 Metafiles

Each dataset on the Census Bureau API includes documentation for geography and variables in the form of JSON and XML files. JSON and XML are two different file formats that are machine generated datasets designed to store meta information including file descriptions and structural information. JSON stands for JavaScript Object Notation and is designed to be a human readable format for sending data. The JSON data structure is centered around name-value pairs. XML stands for Extensible Markup Language and is similar in structure to the HTML format used for webpages. XML is another format for sending and storing data. It is formatted in a tree-like structure with a hierarchy of categories and associated values. These JSON and XML files contain the specific requirements for geography and for the demographic variables.

3.1 Geography

Each dataset available on the Census Bureau online API includes an associated geography file formatted in JSON and a similar file formatted in XML. These files tell us which geographic regions can be included in a valid GET request and the combination of geographic regions that must be specified. The JSON formatted file for geographies has the following format:

```
{
  "name": "tract",
  "requires": [
    "state",
    "county"
  ],
  "optionalWithWCFor": "county"
}
```

This JSON file specifies that census tract level geography always requires a containing state. The `optionalWithWCFor` tag indicates that an additional containing county for census tracts is only sometimes required. An GET request for all census tracts within a state requires a specified state but only optionally requires a county. Requesting all census tracts within a particular county requires both a specified state and a specified county. Requesting a particular census tract instead of all census tracts requires both a specified state and county. These requirements are due to how census tracts are labelled by the Census Bureau. Census tracts are often labelled using a string of numbers and it is not uncommon for census tracts located in different states and counties to share the same label.

The same geographic information is also available in XML. The above geographic information is presented in the following XML format:

```
<fips name="tract">
  <requires name="state"/>
  <requires name="county" is-optional-with-wcfor="true"/>
```

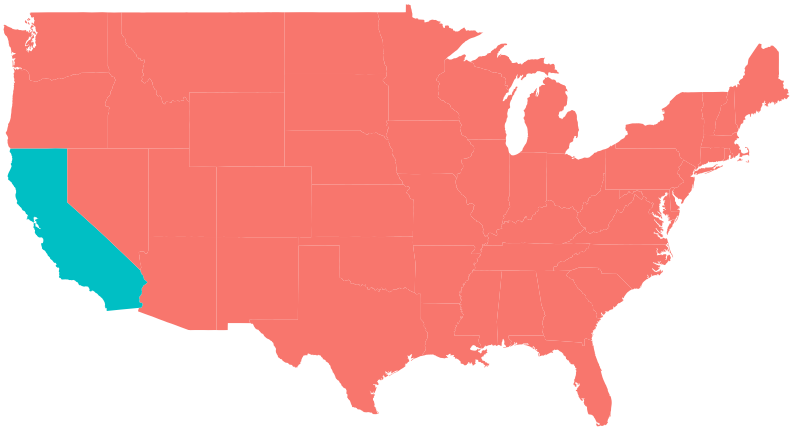


Figure 1: California Selected

`</fips>`

To make the above geographic requires more clear we will look at some examples of various geographies available from the online API and see which combinations of geographic regions are required. Our examples include the United States as a whole and various geographic areas within the state of California. Due to the sheer number of geographic combinations possible, our examples will include the more commonly used geographic entities.

The hierarchy of valid geographic regions for GET requests resembles a tree structure and at the top there is a country-wide geography; this is data aggregated among all states and corresponds to summary level 010 from table 2. Below a country-wide summary the Census Bureau has a state-level geography. At this level it is possible to request data for all states or for particular states; this is summary level 040. In figure 1 California is selected with the following HTTP GET request: `http://api.census.gov/data/[YEAR]/[DATASET]?key=[KEY]&get=[VARIABLES]&for=state:06`.

Below states there are counties and census tracts. County level requests require specifying a containing state. After specifying our containing state California it is possible to request county-level data (summary level 050). From figure 2, Los Angeles county within the state of California is selected. This corresponds to an HTTP GET request of the following form: `http://api.census.gov/data/[YEAR]/[DATASET]?key=[KEY]&get=[VARIABLES]&for=county:037&in=state:06`. There are a few other geographic regions avaiable within a state such as ZIP code tabulation area; for example it is possible to form a valid GET request to select ZIP code region 90210 which corresponds to Beverly Hills, California.

There are a number of valid geographic entities below the state and county level. For example, there are school districts, county subdivision, metropolitan statistical areas and legislative districts. In order to form a valid HTTP GET request, smaller geographic divisions often require specify the containing state or county. In figure 3 Pasadena, a county subdivision within Los Angeles County is selected. County subdivisions are nested within county and state, however it is not possible to specify the Pasadena county subdivision without specifying both California and Los Angeles County. The reason for this is because the Census Bureau API does not recognize subdivision as a standalone geographic entity and state-subdivision is also not a valid geographic combination. From table 2 state-county-subdivision is a valid summary level and this means that only after specifying California and Los Angeles County do we have access to the Pasadena county subdivision. In this case county subdivisions are nested within states, but county subdivisions have a more

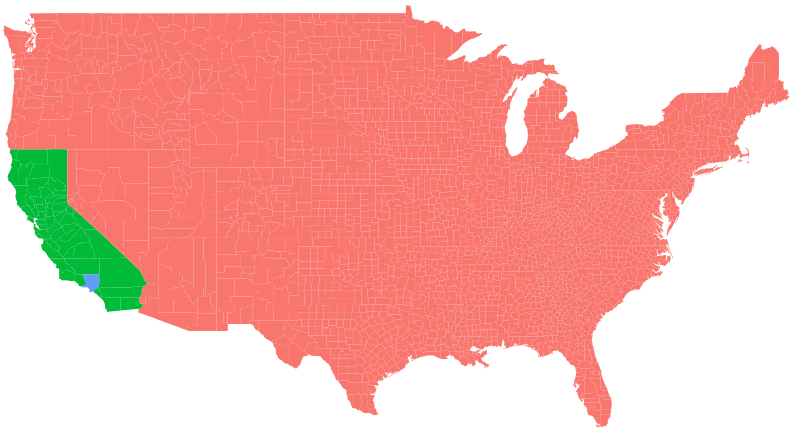


Figure 2: Los Angeles county selected

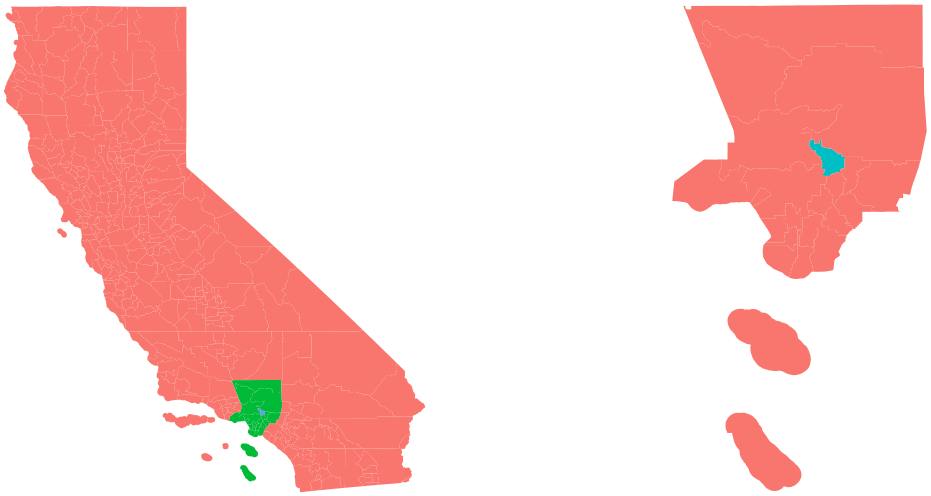


Figure 3: Pasadena county subdivision selected

immediate parent in the hierarchy: counties.

Although all the entries in table 2 are valid geographic combinations, the summary levels do not indicate that some parts of geographic combinations may be optional. The earlier discussion of JSON/XML formatting revealed that for census tracts certain elements in the containing geography can be omitted under certain cases. Recall that for census tracts one option is to specify both the containing state and the county (summary level 140). For census tracts, if the GET request is for all census tracts within a state, the specifying county is optional. This is in contrast to county subdivisions where specifying both state and county is required with no option to leave out county. This means that state-tract is also a valid geographic combination which is not indicated in table 2. It is best to refer to each datasets' JSON or XML file for geographic compatabilities.

Due to the sheer number of different geographic entities there are geographic combinations which cannot be used together for valid HTTP GET requests. Compatible geographies are generally nested. Certain geographic regions will have conflicting borders with other geographic regions and smaller geographic divisions are not necessarily nested in one of the larger geographic divisions; for example, ZIP code areas are generally used by the United States Postal Service and might span different counties or census tracts. Additionally, Legislative districts do not line up with county borders and school districts often do not line up with either legislative districts or county borders. See figure 4 to see a comparison of different geographic entities within Iowa.

From figure 4 state house districts and state senate districts share some common borders but some senate districts are comprised of multiple house districts. In this case house districts are nested within senate districts. Iowa's unified school districts are completely different from both house and senate districts and also do not line up with county borders. This is an example of incompatible geographies; unified school districts are not nested in house, senate or counties and are incompatible with these geographic entities.

For a more detailed look at which geographies need to be specified, refer to the census bureau list of summary levels for each dataset or the XML or JSON geography files. For the 2012 ACS dataset this is located at `http://api.census.gov/data/2012/acs5/geo.html`, the XML file is located at `http://api.census.gov/data/2012/acs5/geography.xml` and the JSON file is located at `http://api.census.gov/data/2012/acs5/geography.json`. The above maps are available in shapefile format from the Census Bureau website located at `http://www.census.gov/cgi-bin/geo/shapefiles2013/main`.

The map for Iowa's school districts can - after downloading the corresponding shapefile, e.g. be created using the following code, which is based on R packages ggplot2 [6], maps [4], and maptools [1]:

```
library(maps)
library(maptools)
library(ggplot2)
iowa.sd = readShapeSpatial("tl_2013_19_unsd.shp")
iowa.sd.geo = fortify(iowa.sd)

theme_map = theme(axis.ticks = element_blank(), axis.text.x = element_blank(),
                  axis.title.x = element_blank(), axis.text.y = element_blank(),
                  axis.title.y = element_blank(), panel.grid = element_blank(),
                  panel.border = element_blank(), legend.position = "none")

# plot the map
qplot(data=iowa.sd.geo, long, lat,
      group=group, order=order, geom="path") +
  theme_bw() + coord_map() +
  theme_map + ggtitle("Iowa Unified School Districts")
```

3.2 Finding Data Sets and Tables Structure

The `[VARIABLES]` parameters depends on dataset. To track which datasets are available there is a master index of available dataset formatted in both JSON and XML provided by the Census Bureau. The JSON file is available online at `http://api.census.gov/data.json` and the XML file is available online at `http:`

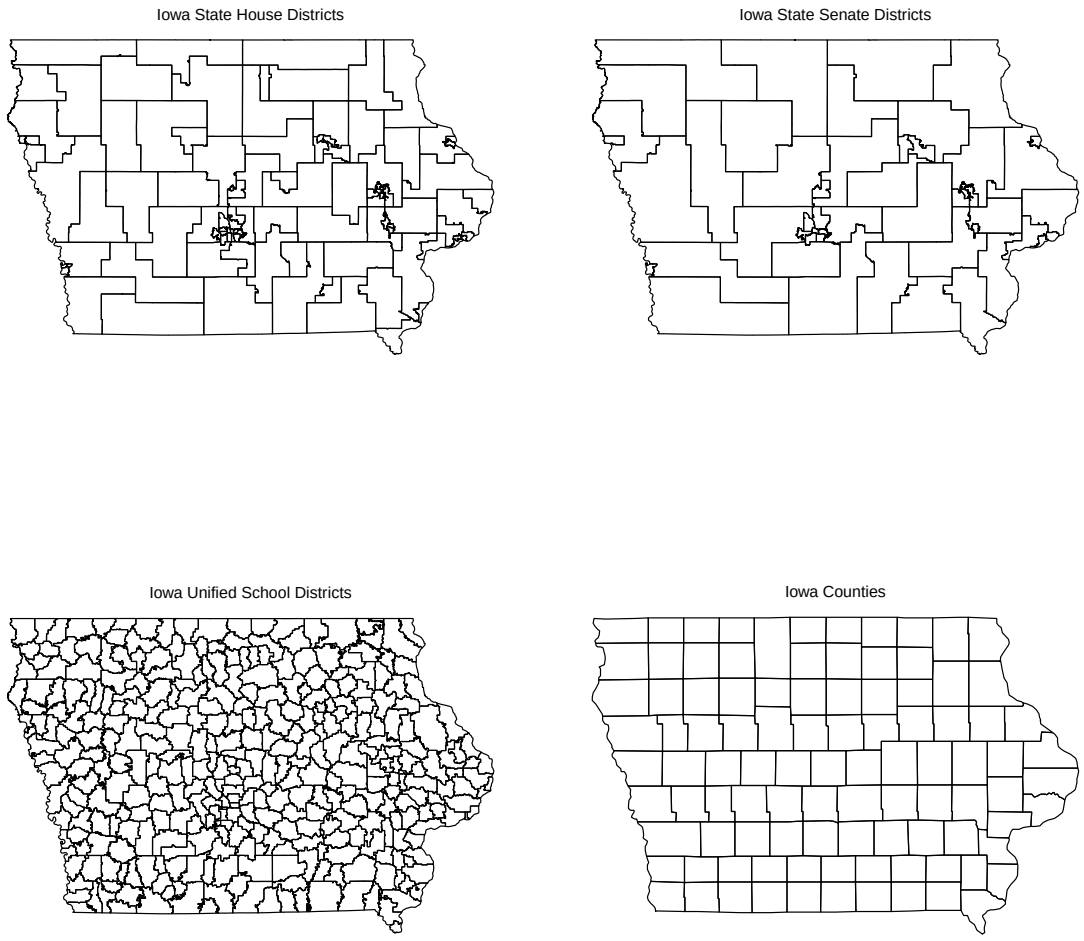


Figure 4: Various geographic entities in Iowa

//api.census.gov/data.xml. This master index includes necessary meta-information about each dataset including description, links to geography and variable information and contact information for maintainer of datasets. The JSON format is formatted as follows:


```
{
  "c_vintage": 2012,
  "c_dataset":[
    "acs5"
  ],
  "c_geographyLink": "http://api.census.gov/data/2012/acs5/geography.json",
  "c_variablesLink": "http://api.census.gov/data/2012/acs5/variables.json",
  "c_tagsLink": "http://api.census.gov/data/2012/acs5/tags.json",
  "c_examplesLink": "http://api.census.gov/data/2012/acs5/examples.json",
  "c_documentationLink": "http://www.census.gov/developers/",
  "c_isAggregate": true,
  "title": "2012 American Community Survey: 5-Year Estimates",
  "webService": "http://api.census.gov/data/2012/acs5",
  "accessLevel": "public",
  "bureauCode": [
    "006:07"
  ],
  "contactPoint": "Census Bureau Call Center",
  "description": "The American Community Survey (ACS) is a nationwide survey...",
  "identifier": "2012acs5",
  "mbox": "pio@census.gov",
  "publisher": "US Census Bureau",
  "references": [
    "http://www.census.gov/developers/"
  ],
  "spatial": "US",
  "temporal": "2012"
  },
}
```


The above excerpt from http://api.census.gov/data.json is the meta-information about the 2008-2012 ACS 5-year dataset. For this dataset there are links to the associated geography file and to another JSON file *variables.json* which is a list of variables available from this dataset. To see the same meta-information in XML format please see appendeix in section 8.1.

After selecting a dataset listed in the master index, the user will lookup what variables are available for that dataset. The JSON and XML master index of datasets contain the location of variables.json and variables.xml respectively which list available variables available for that dataset. Available variables are organized into tables referred to as *concepts*; a *concept* is a combination of factors. For example, "Health Insurance Coverage Status by Sex by Age" is a concept from the 2012 ACS. Within each concept are *labels*; a *label* is a combination of levels for the factors within a concept.

Within each concept there are multiple labels that provide information on different levels of each of the factors. For example, the concept "Health Insurance Coverage Status by Sex by Age" contains a label for males over 70 with health insurance. This concept also contains labels for each combination of gender (male, female), age group (under 6, 6 to 17, 18 to 24, 25 to 34, 35 to 44, 45 to 54, 55 to 64, 65 to 74 and 75 and over) and health insurance coverage (with and without health insurance). Some of the labels may also contain summary information in the form of totals. For this concept there are labels for total number of males, total number of females and totals for males/females in each age group.

To lookup what variables are available, the variables.json and variables.xml files contain a list of

all concepts and labels along with a description. The JSON formatted `variables.json` is formatted as follows:

```
"B27001_056E": {
  "label": "Female:!!75 years and over:!!With health insurance coverage",
  "concept": "B27001.  Health Insurance Coverage Status by Sex by Age"
},
```

This describes label 056E of concept B27001. This is a table of 75 and older females with health insurance. Label 057E describes the corresponding number of females aged 75 and older without health insurance. Both of these numbers are given in table 3. The associated XML formatted version can be found the appendix in section 8.2.

In both the XML and JSON formats the different levels of factors in the labels are separated by !!. To form a valid GET request we take the header from the JSON file (or the *id* from the XML file) and that will be the parameter we use for `VARIABLES`. The concept-label ID `B27001_056E` will request the "75 and older female with health insurance" label from the "health insurance coverage status by sex by age" concept. A valid HTTP GET request for this is `http://api.census.gov/data/[YEAR]/[DATASET]?key=[KEY]&get=B27001_056E,B27001_057E&for=[GEOGRAPHY]`.

	B27001_056E	B27001_057E	NAME	state	county
2	45052	302	Alameda County, California	06	001
3	18	0	Alpine County, California	06	003
4	1733	0	Amador County, California	06	005
5	9431	47	Butte County, California	06	007
6	1938	0	Calaveras County, California	06	009
7	582	0	Colusa County, California	06	011

Table 3: Number of women 75 years and older with (056E) and without (057E) health insurance in California by county from 2012 ACS 5-year data

Table 3 indicates that there are 45,052 women over 75 in Alameda county that have health insurance and there are 302 women over 75 in Alameda county without health insurance.

The files `variables.json` and `variables.xml` not only contain a list of variables as described above but also include margin of error. Estimates from the ACS are based on random sampling thus each of the estimates has a standard error. The Census Bureau uses a 90% confidence level for their margin of error. To find the associated margin of error for our variable we replace the last letter of label with an "M" instead of an "E". For example, `B27001_056M` and `B27001_057M` are the associated standard errors for the previous example. Table 4 gives a list of standard errors for the data in table 3; standard errors have the exact same formatting as any other variable from the ACS.

	B27001_056M	B27001_057M	NAME	state	county
2	356	135	Alameda County, California	06	001
3	13	13	Alpine County, California	06	003
4	51	27	Amador County, California	06	005
5	134	40	Butte County, California	06	007
6	99	27	Calaveras County, California	06	009
7	34	24	Colusa County, California	06	011

Table 4: Margin of error for health insurance status of women 75 years and older in California by county from 2012 ACS 5-year data

Based on tables 3 and 4, 45052 ± 356 is a 90% confidence interval for the number of women in Alameda County 75 and over with health insurance.

4 Limitations of the API

The paper by Stangl, Rundel, and Morgan [5] discusses the use of the American Community Survey as a dataset for classroom exercises (at the undergraduate level) with an emphasis on how data collected from these surveys is used to make important policy decisions. The paper contains a number of classroom exercises that ask the reader to calculate basic proportions about various demographic data such as health insurance coverage, marriage status and income. The particular dataset used by Stangl, Rundel, and Morgan [5] is a 1000 observation random subset of the 2010 ACS public use microdata sample which constitutes a 0.005% subsample of the 2010 ACS, or about a 0.0003% representation of the US population. It would be advantageous by far to get corresponding answers based on the whole of the ACS 5-year dataset. XXX I will try to find an educational reference that realistic examples in the classroom are better for the learning outcome. We will attempt to use the Census Bureau online API to examine the same demographic variables.

Using the Census Bureau's online API for these exercises presents us with three main problems:

1. the structure of data in the public use microdata sample is different from data from the online API
2. proportions and standard errors are not included
3. some combination of variables are simply not available from the online API

We highlight each of these issues in this section following along some of Stangl et al's examples.

4.1 Structure of the Data

Sex	Age	Married	Income	HoursWk	Race	USCitizen	HealthInsurance	Language
0	31	0	60.00	40	white	1	1	1
1	31	0	0.36	12	black	1	1	0
1	75	0	0.00	NA	white	1	1	0
0	80	0	0.00	NA	white	1	1	0
1	64	1	0.00	NA	white	1	1	0
1	14	0	NA	NA	white	1	1	0

Table 5: Random subset of 2010 ACS public use microdata sample

Data from the ACS public use microdata sample is organized to describe individuals. Each row of the dataset describes an anonymized individual and each column represents a different demographic variable. See table 5 for a small subset of the data used by Stangl et al.

	Total	M Total	M <6	M <6 w/insurance	M <6 w/o insurance
Alabama	4693822	2256713	186155	176591	9564
Alaska	686905	349855	33053	29026	4027
Arizona	6304406	3099407	279297	249163	30134
Arkansas	2862023	1394466	120828	115053	5775
California	36783532	18138870	1561623	1461559	100064
Colorado	4949633	2457605	210948	193227	17721

Table 6: Health Insurance Coverage information from Census API. Each of the M columns indicate male, subsequent numbers indicate age; there are corresponding F columns for females that are not shown to conserve space.

By contrast the online API does not provide individual specific data. When we perform a HTTP GET request we must specify what geographic level of detail we want. The geographic level of detail does not go below the county subdivision or census tract level. Instead of individual specific data we have data that has

been aggregated for an entire geographic region. This is likely due to privacy reasons; if we have individual specific data for a dozen demographic variables along with geographic information it might be possible to reveal this individual's identity.

We can access data from the API using the acs (Glenn [2]) package in R. In table 6 we have used the acs package to find state level summaries of health insurance coverage status for various age groups by gender (from the census API this is table B27001). Each variable is in the form of a table and as previously mentioned we refer to this as a *c*oncept and each column in table 6 is referred a *l*abel. The first label of each concept is an overall total; the total number of people in the specified geographic region that answered questions relating to the requested concept. Subsequent labels are subsets of this overall total. We have conveniently renamed column headers in table 6; the naming convention from data provided by the API is to enumerate each label with the table name: `B27001_001`, `B27001_002` etc.

The way this data is organized is not in a tidy format, see [8]. Instead it appears that the columns contain additional categorical information: gender, age and insurance. In table 7 we have tidied up the data so that each row is an observation and each column is a variable. Originally table B27001 contains a number of *total* columns: overall total, total number of males, total number of females, and within each gender a total number of people within an age group. In reshaping this data, we felt that these total columns were redundant once the data is in a tidy form.

state	gender	age	insurance	freq
Alabama	m	<6	yes	176591
Alaska	m	<6	yes	29026
Arizona	m	<6	yes	249163
Arkansas	m	<6	yes	115053
California	m	<6	yes	1461559
Colorado	m	<6	yes	193227

Table 7: Reshaped Health Insurance Coverage data.

4.2 Calculating Proportions

The Census Bureau has historically provided information in the form of counts and for a majority of available datasets data is presented as a nominal frequency instead of a proportion. Nominal frequency of variables may not hold much meaning as different geographic entities contain different populations. For example, in table 8 we have university enrollment by age and gender for each state from the 2012 5-year ACS dataset. This dataset is reshaped into a tidy form. In this dataset we have gender, age and enrollment status (public university, private university or not enrolled). California had over 700,000 male students at public universities while Colorado only had around 80,000 students in the same category. These nominal enrollment numbers mean very little without knowing the size of the population.

state	gender	school	age	freq
Alabama	m	public	18.to.24	73085
Alaska	m	public	18.to.24	7852
Arizona	m	public	18.to.24	98686
Arkansas	m	public	18.to.24	41490
California	m	public	18.to.24	713506
Colorado	m	public	18.to.24	84460

Table 8: University enrollment by gender, age and university type

The size of the population depends on the topic of interest. One potential topic of interest is gender differences in public university enrollment in college aged individuals. In this case our population is the total number

of college aged students enrolled in a public university. Proportions are not available for most variables and must be calculated by the user; in table 9, we modify the above table and calculate the percentage of males/females for each combination of school and age group for each state.

state	gender	school	age	pct
Alabama	m	public	18.to.24	44.09
Alaska	m	public	18.to.24	44.71
Arizona	m	public	18.to.24	46.42
Arkansas	m	public	18.to.24	46.06
California	m	public	18.to.24	47.80
Colorado	m	public	18.to.24	48.65

Table 9: University enrollment by gender, age and university type

These percentages tell us a lot more about the data; when it comes to public university enrollment for college-aged invidividuals both California and Colorado have fewer males enrolled. Finding these proportions was not difficult but it requires a bit more work to correctly add up the frequencies in our population.

A more difficult problem lies in finding standard errors for confidence intervals. Recall that the Census Bureau provides a margin of error for each variable at the 90% confidence level. Finding standard errors (SE) for the estimates from the margin of error (MOE) is straight forward: $SE = \text{MOE}/1.645$ where 1.645 is the critical value from the standard normal distribution that corresponds to a central area of 0.90. Finding the standard error for proportions requires further calculations:

$$SE(A/B) = 1/B\sqrt{SE(A)^2 - (A/B)^2\, SE(B)^2}$$

In the above formula, the proportion A/B is a ratio where the numerator is a subset of the denominator and both A and B are estimates from the ACS dataset. For example, if A/B is the proportion of people with health insurance, then A would be the total number of people with health insurance and B would be the overall total number of people

In calculating proportions we must often add different estimates together for different labels. Combining the standard error for a sum of estimates: $SE(A+B) = \sqrt{SE(A)^2 + SE(B)^2}$. Note that previous two formulas for standard errors are approximations and do not consider any covariances. In fact the standard error for a sum of estimates might be increasingly biased when summing a large number of estimates and will not match standard errors calculated using other methods.

4.3 Examples from Stangl and Data Access

We examine some of the exercises listed in Stangl, Rundel, and Morgan [5] and later discuss the difficulty in extending these exercises by subsetting on additional factors. The first example is exercise 3 from Stangl, Rundel, and Morgan [5] and the exercise is to generate confidence intervals for health insurance data. The exercise is listed below:

> In our sample of 1000 people, 861 have health insurance and 139 do not. Generate a confidence interval for the proportion of U.S. residents who do not have health insurance.

First we calculate an estimate for the proportion of people who do not have health insurance using Stangl's dataset Stangl, Rundel, and Morgan [5] and the corresponding 2012 5-year ACS numbers from the online API. The 1000 observation subset used in Stangl, Rundel, and Morgan [5] is in a tidy format and calculating insurance coverage can be done easily by summing rows.

```
acsdat = read.csv("datasetSTANGL2013.csv")  #Stangl's dataset
p_no_insurance = sum(!acsdat$HealthInsurance)/length(acsdat$HealthInsurance)
print(p_no_insurance)
```

```
[1] 0.139
```

The same calculation can be done using the ACS data from the online API but because the data is not in a tidy form it requires summing the columns together and more work from the user. The following code prints out the first 10 column names from the health insurance coverage status by age and gender table from the 2012 5-year ACS. Finding health care coverage percentage requires summing together the correct columns to find the total number of people without health insurance. Because the column headers store information about gender, age group and insurance status we must manually select the correct columns to aggregate.

```
# First ten column names for this table
head(colnames(health_acs@estimate), 10)
```

```
 [1] "Health Insurance Coverage Status by Sex by Age: Total:"
 [2] "Health Insurance Coverage Status by Sex by Age: Male:"
 [3] "Health Insurance Coverage Status by Sex by Age: Male: Under 6 years:"
 [4] "Health Insurance Coverage Status by Sex by Age: Male: Under 6 years: With health
     insurance coverage"
 [5] "Health Insurance Coverage Status by Sex by Age: Male: Under 6 years: No health
     insurance coverage"
 [6] "Health Insurance Coverage Status by Sex by Age: Male: 6 to 17 years:"
 [7] "Health Insurance Coverage Status by Sex by Age: Male: 6 to 17 years: With health
     insurance coverage"
 [8] "Health Insurance Coverage Status by Sex by Age: Male: 6 to 17 years: No health
     insurance coverage"
 [9] "Health Insurance Coverage Status by Sex by Age: Male: 18 to 24 years:"
[10] "Health Insurance Coverage Status by Sex by Age: Male: 18 to 24 years: With health
    insurance coverage"
```

```
# proportion with no insurance
p_no_insurance_acs = sum(health_acs@estimate[c(seq(5, 30, by = 3), seq(33, 57,
    by = 3))])/health_acs@estimate[1]
print(p_no_insurance_acs)
```

```
[1] 0.1487
```

Next we calculate 95% confidence intervals for proportion of people without health insurance using both the 1000 observation subset and the corresponding 2012 5-year ACS numbers from the online API. For Stangl's exercise, the confidence interval can be found using a formula familiar to introductory statistics students: $\hat{p} \pm 1.96\sqrt{\frac{\hat{p}(1-\hat{p})}{n}}$. From this formula, a 95% confidence interval for the proportion of people in the United States without health insurance is: (0.1176, 0.1604).

The corresponding 95% confidence interval for data from the online API is calculated from the available margin of error data for each table. Standard errors are calculated from the margin of error and then the formulas from section 4.2 are used to find the standard error for proportions. A 95% confidence interval using data from the online API for the proportion of people in the united states without health insurance is: (0.1483, 0.1492).

From this first example, the estimates of proportion of people without health insurance is similar in both Stangl's dataset and the ACS dataset from the online API. The standard errors are significantly different simply due to the size of Stangl's dataset compared to the size of the ACS. In the following example we try to expand further by subsetting on more factors. The next exercise from Stangl, Rundel, and Morgan [5] is to compare health insurance status by race:

> Of the 1000 people in our dataset, 761 are white, 106 are black, 70 are asian and 63 are other races. The sample proportions of people who have health insurance by racial group are $\hat{p}_{white} = 0.880$, $\hat{p}_{black} = 0.811$, $\hat{p}_{asian} = 0.843$, $\hat{p}_{other} = 0.730$. Create the two-way table for counts of health insurance by race.

Confirming the numbers in this exercise using the Stangl's 1000 observation subset is straight forward and requires a single line of code using the plyr [7] package in R. We can summarise percentage of people with health insurance by race as follows:

```
library(plyr)
acsdat = read.csv("datasetSTANGL2013.csv") #Stangl's dataset
head(acsdat)
```

```
  Sex Age Married Income HoursWk  Race USCitizen HealthInsurance Language
1   0  31       0  60.00      40 white         1               1        1
2   1  31       0   0.36      12 black         1               1        0
3   1  75       0   0.00      NA white         1               1        0
4   0  80       0   0.00      NA white         1               1        0
5   1  64       1   0.00      NA white         1               1        0
6   1  14       0     NA      NA white         1               1        0
```

```
acsdat_by_race = ddply(acsdat, .(Race), summarise,
                       insurance_pct = 100*sum(HealthInsurance)/length(Race))
print(acsdat_by_race)
```

```
   Race insurance_pct
1 asian          84.29
2 black          81.13
3 other          73.02
4 white          88.04
```

This same calculation can be done from the 2012 5-year ACS data provided by the online API. The data from the online API for health insurance coverage by race is separated into eight tables: insurance coverage for each race is located in a different table and requires a separate GET request for each race. The following code shows the first few columns of the insurance coverage by age table by race. Similar to the previous example, the columns of the data set store information on age and insurance status and to calculate health insurance coverage by race requires summing together the correct columns and dividing by the correct overall total. This operation must be repeated for multiple tables.

```
# column names from ACS
print(colnames(health_white))
```

```
 [1] "Health Insurance Coverage Status by Age (White Alone): Total:"
 [2] "Health Insurance Coverage Status by Age (White Alone): Under 18 years:"
 [3] "Health Insurance Coverage Status by Age (White Alone): Under 18 years: With health
     insurance coverage"
 [4] "Health Insurance Coverage Status by Age (White Alone): Under 18 years: No health
     insurance coverage"
 [5] "Health Insurance Coverage Status by Age (White Alone): 18 to 64 years:"
 [6] "Health Insurance Coverage Status by Age (White Alone): 18 to 64 years: With health
     insurance coverage"
 [7] "Health Insurance Coverage Status by Age (White Alone): 18 to 64 years: No health
     insurance coverage"
 [8] "Health Insurance Coverage Status by Age (White Alone): 65 years and over:"
 [9] "Health Insurance Coverage Status by Age (White Alone): 65 years and over: With health
     insurance coverage"
[10] "Health Insurance Coverage Status by Age (White Alone): 65 years and over: No health
     insurance coverage"
```

```
# sums columns together
calc_pct = function(df) {
    sum(df[c(3, 6, 9)])/df[, 1] * 100
}

# sum columns
a_pct = calc_pct(health_asian)
b_pct = calc_pct(health_black)
o_pct = calc_pct(health_other)
w_pct = calc_pct(health_white)
```

From the previous two examples, the format of ACS data from the online API is in a very rigid format making it difficult to subset on variables and difficult to sum over all levels variables. In the previous exercise to calculate health insurance coverage percentage for each race, the data from the online API only provides a table for health insurance by race and age so this requires the user to aggregate estimates over age. For datasets with more variables than just age and health insurance coverage, the non-tidy structure of this data means that summaries of some of the variables will requiring summing over all levels of other variables. Summing over variables is a column-wise operation because of the non-tidy format and this requires the user to manually select the correct columns. Similarly, finding proportions requires column-wise summing and dividing by the correct *t*otal column. By contrast, the 1000 sample dataset from Stangl, Rundel, and Morgan [5] is in tidy form and summing over levels of variables and subsetting are row-wise operations.

We have summed up health insurance coverage numbers over the different age groups for each race and found the overall health coverage percentage numbers:

```
[1] "asian insurance_pct: 85.1522"
```

```
[1] "black insurance_pct: 82.5249"
```

```
[1] "other insurance_pct: 73.3575"
```

```
[1] "white insurance_pct: 86.8986"
```

The rigid format of each table also makes it difficult to subset on additional variables. What if the user is now interested in health insurance coverage percentage by race, gender and further variables? From the dataset in Stangl, Rundel, and Morgan [5], it is possible to subset further: health insurance coverage percentage by race and gender. This calculation is a row-wise operation and requires one line of code using the plyr package in R:

```
library(plyr)
acsdat = read.csv("datasetSTANGL2013.csv") #Stangl's dataset
head(acsdat)
```

```
  Sex Age Married Income HoursWk  Race USCitizen HealthInsurance Language
1   0  31       0  60.00      40 white         1               1        1
2   1  31       0   0.36      12 black         1               1        0
3   1  75       0   0.00      NA white         1               1        0
4   0  80       0   0.00      NA white         1               1        0
5   1  64       1   0.00      NA white         1               1        0
6   1  14       0     NA      NA white         1               1        0
```

```
acsdat_gender_race = ddply(acsdat, .(Race, Sex), summarise,
                           insurance_pct = 100*sum(HealthInsurance)/length(Race))
print(acsdat_gender_race)
```

```
   Race Sex insurance_pct
1 asian   0          93.33
2 asian   1          77.50
3 black   0          88.33
4 black   1          71.74
5 other   0          72.73
6 other   1          73.33
7 white   0          87.20
8 white   1          89.05
```

From the online API there are two related tables for health insurance coverage percentage: health insurance by race and age,and health insurance by gender and age. Unfortunately it is not possible to combine information from these two tables without further information. Our only option is to look for a table that includes all combinations of variables of interest. This is a huge limitation for ACS data from the online API; we are limited in the combinations of variables available and must depend on a table having the exact variables of interest.

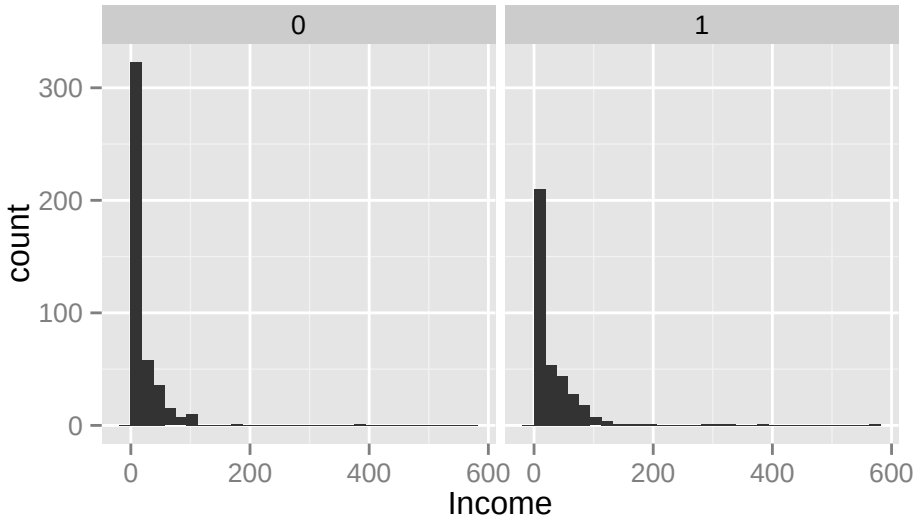


Figure 5: Income by Gender

The above limitation is demonstrated by modifying another exercise from Stangl, Rundel, and Morgan [5]. Exercise 7 from Stangl, Rundel, and Morgan [5] is working with income and gender data and asks the follow:

> Produce a plot showing the relationship between income and gender, comment on what you see, and summarize the relationship with appropriate summary statistics. We can easily generate a histogram of income for each gender using the dataset in Stangl, Rundel, and Morgan [5]:

```
library(ggplot2)
acsdat = read.csv("datasetSTANGL2013.csv") #Stangl's dataset
qplot(Income, data=acsdat, facets=~Sex)
```

The 2012 5-year ACS dataset has many different tables involving different measures of income: aggregate income, household income, per-capita income and many more. To recreate the above exercise, the ACS dataset from the online API has a table for median income in the past 12 months by gender and work hours for the population above 15 years old. This data is displayed in table 10. This data set displays hours worked as full-time workers or other and note that these numbers are displayed in 2012 inflation-adjusted dollars.

total	male_total	male_full	male_other	female_total	female_full	female_other
22318	29556	45667	14669	17432	33430	10954
31005	39792	60673	19015	24258	44522	12552
26388	31794	46774	17777	21357	37703	12699
21604	26978	40919	14622	17171	31297	10945
27129	32630	52364	16553	22057	44022	12677
30084	36651	52889	17561	23759	41712	13003
32842	42570	64027	19285	26149	49595	14738
29752	36587	52286	19780	24136	41909	13244
38230	42189	67488	14267	35105	59940	12835
24683	29829	43405	17141	20400	35699	12526

Table 10: Median Income in the past 12 months by Gender and Hours

From this data we can generate a similar histogram with one observation per state, this histogram is in figure 6. However, if instead we are interested in median income for full-term works average over both genders,

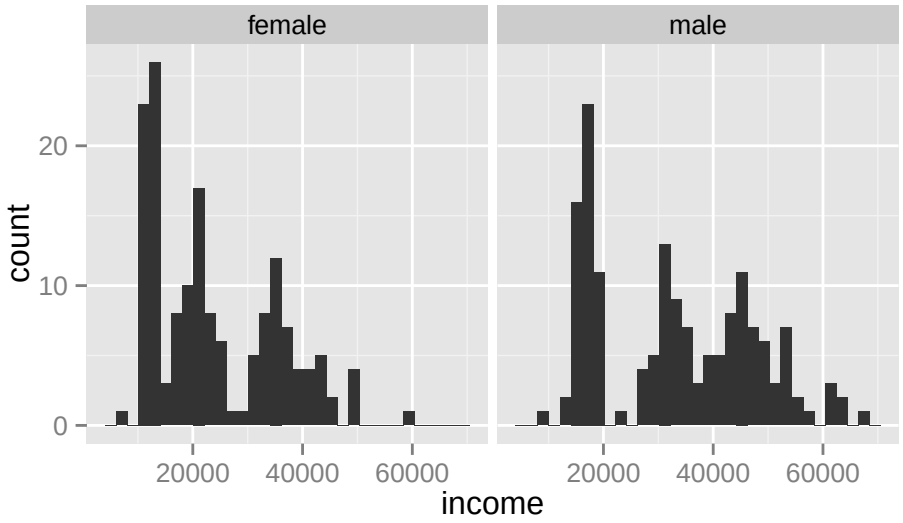


Figure 6: Median Income by Gender

this data table would not be entirely useful. From table 10, it is clear that it is not possible to calculate the exact median income for full-time works averaged over both genders without more information. In the previous example the main limitation was not being able to subset on additional variables, in this example the main limitation is not being able to average over the available variables.

5 Other Issues

Although the ACS is a yearly survey, in its current form it is difficult to compare variables across years. The online API has ACS data available for 2010 to 2012 but even for these years it is difficult to track changes across these years. Recall that the 2012 ACS 5-year dataset is data aggregated from 2008 - 2012. This means that the 2010, 2011 and 2012 ACS 5-year datasets will include overlapping time periods; some of the data used aggregated to form these 5-year estimates will overlap.

It is possible to circumvent this overlapping data problem by combining the ACS 5-year estimate with the ACS 1-year estimate. The 2011 ACS 5-year and the 2012 ACS 1-year do not contain overlapping data. However, datasets from different years contain different demographic variables. For example, questions regarding healthcare were first added in 2008 and datasets prior to 2008 will not contain healthcare variables. The ACS 1-year estimates have smaller sample sizes and as a result not all geographic regions are available. The ACS 1-year estimates only have data available for geographic regions with populations of 65,000 or greater.

As a guideline, the Census Bureau recommends comparing similar length datasets that do not overlap. For example, comparing 1-year estimates from different years or 3-year estimates for two periods that don't overlap. Unfortunately, the only option for comparing data across years out of the available datasets from the online API is to compare 2011 and 2012 1-year estimates.

6 Examples from ACS

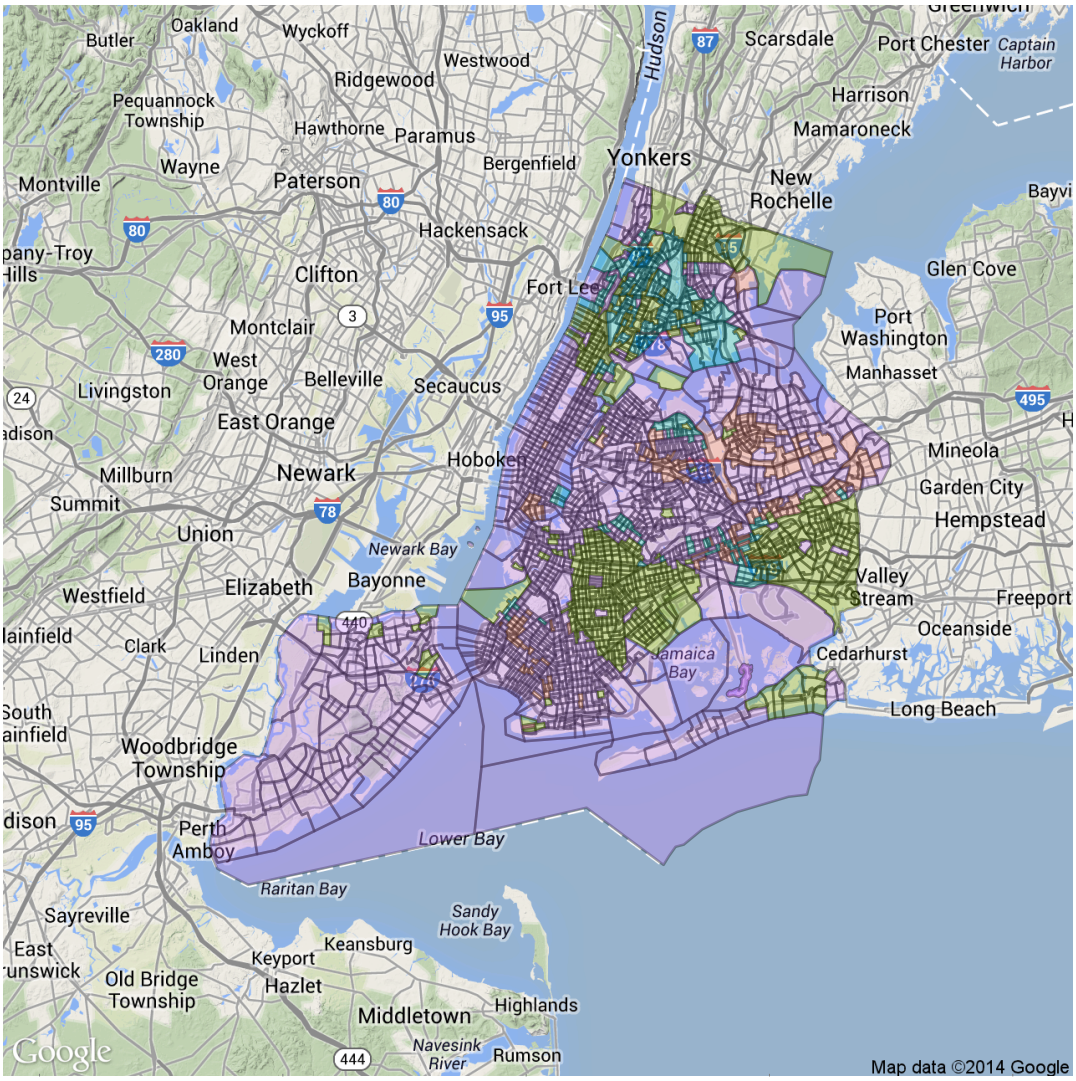
Although the Census Bureau's API has some drawbacks the available data is available at a very detailed level. For the 5-year ACS datasets, the geographic regions available include county-level and smaller regions. The full extent of the data is available by a census tract by census tract level of detail. For areas with high

population density, such as New York City, census tract level data allows for a block-by-block level of detail.

In the following example we look at the majority race in New York City for each census tract. The shape files for New York census tracts is available from `http://www.census.gov/cgi-bin/geo/shapefiles2013/main`; from this shape file we extract the coordinates for the five boroughs of New York City: New York County (Manhattan), Bronx County (Bronx), Kings County (Brooklyn), Queens County (Queens) and Richmond County (Staten Island). We perform a GET request using the acs package (Glenn [2]) and we plot this on top of google maps using ggmaps (Kahle and Wickham [3]). The following code plots majority race for each census tract in New York City:

```
library(maptools)
library(ggplot2)
library(plyr)
library(ggmap)
library(acs)

#get shape file for NY
ny.shape = readShapeSpatial("tl_2013_36_tract.shp")
ny.shape@data$id = rownames(ny.shape@data)
ny.points = fortify(ny.shape)
ny.df = join(ny.points, ny.shape@data, by="id")
ny.df$NAME = as.numeric(as.character(ny.df$NAME))
ny.df$COUNTYFP = as.numeric(as.character(ny.df$COUNTYFP))
ny.df = subset(ny.df,  COUNTYFP %in% c(61, 5, 47, 81, 85))
#NAME corresponds to census tracts

#get data from ACS
NYC = geo.make(state="NY", county=c("New York County",
                                    "Bronx County",
                                    "Kings County",
                                    "Queens County",
                                    "Richmond County"), tract="*")
f.nyc = acs.fetch(endyear = 2012, span = 5,
                  geography = NYC,
                  table.name = c("In Combination With One Or More Other Races"),
                  col.names="pretty")

#cleanup ACS names and dataframe
#note: kings county = brooklyn,
#new york county = manhattan
#richmond county = staten island
f.nyc = data.frame(f.nyc@estimate)
split=ldply(strsplit(rownames(f.nyc), ","))
f.nyc$state = tolower(sub("^.*, ", "", rownames(f.nyc)))
f.nyc$tract = tolower(gsub("Census Tract ", "", split[,1]))
f.nyc$county = tolower(gsub("^[ ]","",split[,2]))
f.nyc$county = gsub(" county", "", f.nyc$county)
rownames(f.nyc) = NULL
f.nyc$tract = as.numeric(f.nyc$tract)

#lookup county names based on COUNTYFP number
ny2.df = merge(x = ny.df, y = subset(fips.county, State == 'NY'),
               by.x = "COUNTYFP", by.y = "County.ANSI")
ny2.df$County.Name = tolower(ny2.df$County.Name)
ny2.df$County.Name = gsub(" county", "", ny2.df$County.Name)

#merge shape files with ACS data by county name
final = merge(x = ny2.df, y = f.nyc,
              by.x = c("NAME", "County.Name"),
              by.y = c("tract", "county"), all.x=TRUE)
names(final)[24:29] = c("white", "black", "indian", "asian", "hawaiian", "other")
final$race = unlist(apply(final[,24:29], 1, function(x){
  if(sum(is.na(x)) == length(24:29)) return("NA")
  else return(names(which.max(x)))
}))
```

```
#plot map
qmap("brooklyn new york city", zoom=10) +
  geom_path(aes(x = long, y = lat, group=group, order=order, width = 0.7),
            data = final, alpha = 0.35) +
  geom_polygon(aes(x = long, y = lat, group = group, order = order, fill = race),
               data = final, alpha = 0.3)
```



7 Conclusion

Among the various tools to access data from the Census Bureau the online API is the most complicated and some features are not entirely well documented. Although the online API is more difficult to use it allows more comprehensive access to the ACS and decennial census datasets compared to the other web tools provided by the Census Bureau. Who is the target audience for the online developer's API?

Due to the technical knowledge required to access the online API, clearly the audience is not local business leaders or congressional staffers. It is unlikely someone without significant programming knowledge would be able to understand the JSON and XML formats required to find datasets and variables. Due to the structure of the data and the emphasis on frequency instead of proportions, the online API is also not designed for statisticians. The form of the data limits the available variables for each table and makes it difficult to subset on variables. Right now the online API is useful only to application developers who cannot store the contents of the entire ACS datasets.
We occasionally used the acs package (Glenn [2]) in R instead of working directly with the JSON output. However, the acs package still requires a lot of data processing: both in reshaping the data to tidy form and in dealing with the default column names from the raw dataset. The online API makes available a lot of census data but it requires the user to do additional processing to be useful for any statistical analysis. With some restructuring the API could be a useful tool for statisticians since it allows for a detailed look down to the census tract level and it allows access to data without downloading the entire dataset.

8 Appendix

8.1 Meta-information for datasets

XML formatted meta-information about the 2008-2012 ACS 5-year dataset:

```
<dataset vintage="2012"
  geographyLink="http://api.census.gov/data/2012/acs5/geography.xml"
  variablesLink="http://api.census.gov/data/2012/acs5/variables.xml"
  tagsLink="http://api.census.gov/data/2012/acs5/tags.xml"
  examplesLink="http://api.census.gov/data/2012/acs5/examples.xml"
  documentationLink="http://www.census.gov/developers/"
  pod:webService="http://api.census.gov/data/2012/acs5" isAggregate="true"
  pod:accessLevel="public"
  dcat:contactPoint="Census Bureau Call Center"
  dct:identifier="2012acs5"
  pod:mbox="pio@census.gov"
  dct:publisher="US Census Bureau"
  dct:spatial="US"
  dct:temporal="2012">
<dataset-name> <part name="acs5"/> </dataset-name>
<dct:title>2012 American Community Survey: 5-Year Estimates</dct:title>
<pod:bureauCode> 006:07 </pod:bureauCode>
<dct:description>
The American Community Survey (ACS) is a nationwide survey...
</dct:description>
<pod:reference link="http://www.census.gov/developers/"/>
</dataset>
```

8.2 Meta-information for variables

```
<var xml:id="B27001_056E"
  label="Female:!!75 years and over:!!With health insurance coverage"
  concept="B27001. Health Insurance Coverage Status by Sex by Age"/>
```

References

[1] Roger Bivand and Nicholas Lewin-Koh. *maptools: Tools for reading and handling spatial objects*. R package version 0.8-27. 2013. URL: http://CRAN.R-project.org/package=maptools.

[2] Ezra Haber Glenn. *acs: Download, manipulate, and present data from the US Census American Community Survey*. R package version 1.2. 2014. URL: http://CRAN.R-project.org/package=acs.

[3] David Kahle and Hadley Wickham. *ggmap: A package for spatial visualization with Google Maps and OpenStreetMap*. R package version 2.3. 2013. URL: http://CRAN.R-project.org/package=ggmap.

[4] Original S code by Richard A. Becker and Allan R. Wilks. R version by Ray Brownrigg. Enhancements by Thomas P Minka <tpminka@media.mit.edu>. *maps: Draw Geographical Maps*. R package version 2.3-6. 2013. URL: http://CRAN.R-project.org/package=maps.

[5] Dalene Stangl, Mine Çetinkaya Rundel, and Kari Lock Morgan. "Taking a Chance in the Classroom: The American Community Survey". In: *CHANCE* 26.1 (2013), pp. 42–46. DOI: 10.1080/09332480.2013.772392. URL: http://www.tandfonline.com/doi/abs/10.1080/09332480.2013.772392.

[6] Hadley Wickham. *ggplot2: elegant graphics for data analysis*. Springer New York, 2009. ISBN: 978-0-387-98140-6. URL: http://had.co.nz/ggplot2/book.

[7] Hadley Wickham. "The Split-Apply-Combine Strategy for Data Analysis". In: *Journal of Statistical Software* 40.1 (2011), pp. 1–29. URL: http://www.jstatsoft.org/v40/i01/.

[8] Hadley Wickham. "Tidy Data". In: *Under review by the Journal of Statistical Software* (2014). URL: http://vita.had.co.nz/papers/tidy-data.pdf.