Accessing Data with the Census Bureau API

Alex Shum

April 17, 2014

Abstract

Placeholder.
your solution to cut down on the length of the commands looks great!

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/`. 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 will discuss how the ACS data is structured when we request data and how to access data from the Census Bureau's online developer's API. We will also discuss what kind of variables are available and some limitations with the API. We will base this discussion on a paper by Stangl, Rundel, and Morgan [3] 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 (see table 1) is formed as follows:

`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 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 we can 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 complete table of geographic entities available on the census API for the 2012 ACS. XXX I know that I've asked before, and I know that you had a reason - but I forgot: what does the number of the summary level tell us? Could you include that in the description (or leave it out - it's a bit confusing). Could you extend the table by including the valid 'for' and any required 'in' geographic regions? I'm for example not quite sure how I would ask for a county subdivision - how o you deal with the white space? include a plus symbol?

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

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"/>
```

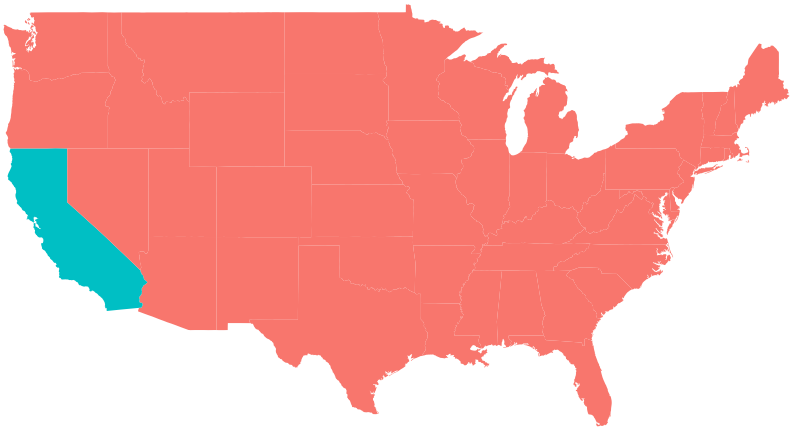

Figure 1: California Selected

`</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

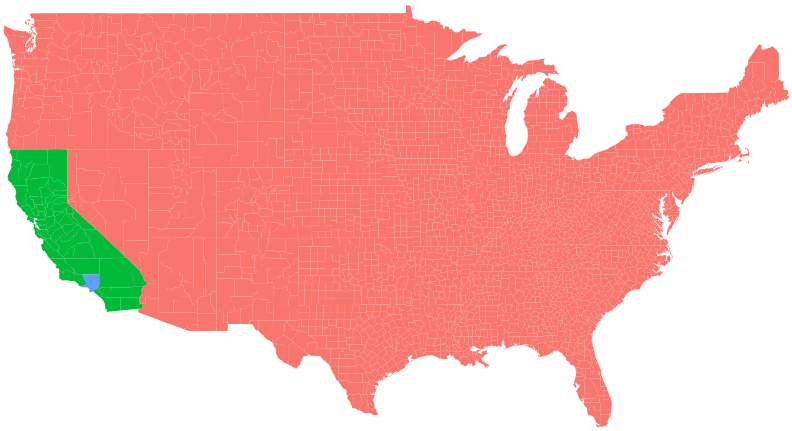

Figure 2: Los Angeles county selected

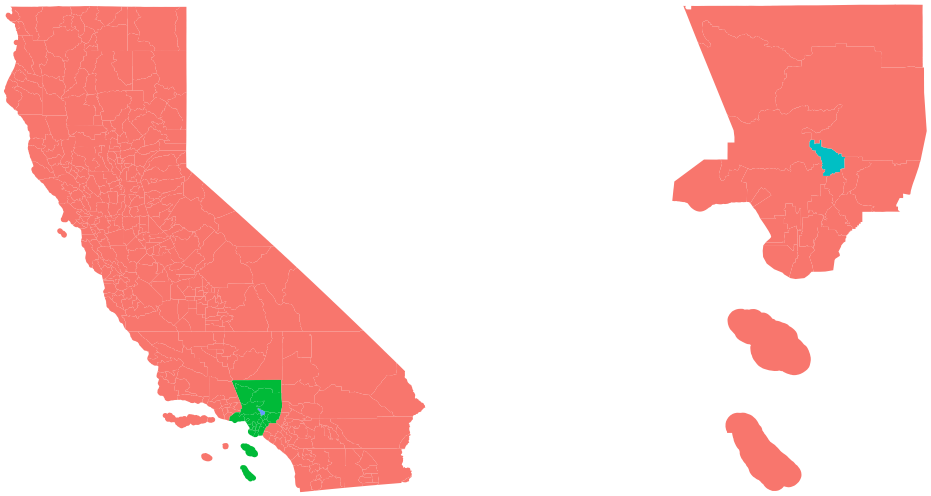

Figure 3: Pasadena county subdivision selected

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 [4], maps [2], 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

XXX could you go over this section again and remove all of the duplicates? - we only need HTML or XML descriptions, and we also only need either the JSON output or a table (e.g. table 3)

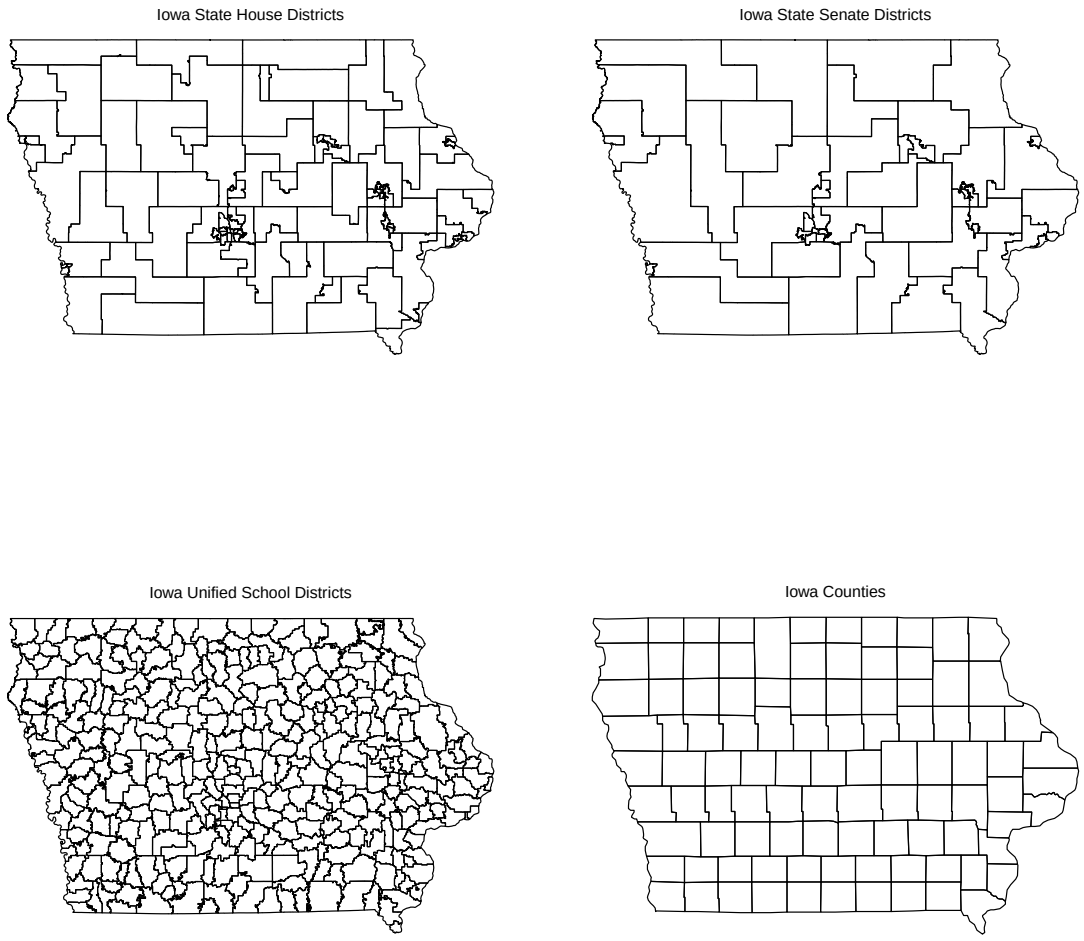

Figure 4: Various geographic entities in Iowa

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://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. For the above example the same meta-information is formatted in XML as follows:

```
<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>
```

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. The associated XML formatted version is formatted as follows:

```
<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"/>
```

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&for=[GEOGRAPHY]`.

```
## Warning:  incomplete final line found on 'http://api.census.gov/data/2012/acs5?key=b59354d11f96afaa8
```

Table 3 is information on health insurance for women 75 and older in California from the 2012 ACS 5-year data listed by county. The HTTP GET request for this is `http://api.census.gov/data/2012/acs5?key=[KEY]&get=B27001_056E,B27001_057E,NAME&for=county:*&in=state:06`. Labels 056E and 057E are labels for women 75 and older with health insurance and women 75 and older without health insurance respectively. The result of our HTTP GET request is a JSON formatted file in the following format:

```
[["B27001_056E","B27001_057E","NAME","state","county"],
```

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

Table 3: Health insurance status of women 75 years and older in California by county from 2012 ACS 5-year data

```
["668","144","Alameda County, California","06","001"],
["15","69","Alpine County, California","06","003"],
["88","131","Amador County, California","06","005"],
["238","127","Butte County, California","06","007"],
["45","102","Calaveras County, California","06","009"],
["98","26","Colusa County, California","06","011"],
```

This indicates that there are 668 women over 75 in Alameda county that have health insurance and 144 women over 75 in Alameda county without health insurance.

The list of variables, *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. Standard errors have the exact same formatting as any other variable from the ACS: XXX I think it should be B27001_057M instead of B27001_057E

```
[["B27001_056M","B27001_057E","NAME","state","county"],
["356","302","Alameda County, California","06","001"],
["13","0","Alpine County, California","06","003"],
["51","0","Amador County, California","06","005"],
["134","47","Butte County, California","06","007"],
["99","0","Calaveras County, California","06","009"],
["34","0","Colusa County, California","06","011"],
```

Using the above table, 668 ± 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 [3] talks about using the ACS as a dataset for classroom exercises (at the undergraduate level) with an emphasis on how data collected from these surveys are 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 [3] 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 2010 ACS. 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. certain combination of variables are simply not available from the online API.

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		white	1	1	0
0	80	0	0.00		white	1	1	0
1	64	1	0.00		white	1	1	0
1	14	0			white	1	1	0

Table 4: 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 4 for a small subset of the data used by Morgan et al.

```
## Error:  trying to get slot "estimate" from an object of a basic class ("logical") with no
slots
## Error:  object 'dat.table' not found
## Error:  object 'dat.table' not found
## Error:  object 'tab1' not found
## Error:  object 'tab1' not found
```

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.

In table **??** we specify state level summaries for 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 *concept* and each column in table **??** is referred a *label*. 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've conveniently renamed column headers in table **??**; 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 tidy [6]. Instead it appears that the columns contain additional categorical information: gender, age and insurance. In table **??** 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.

```
## Error:  trying to get slot "estimate" from an object of a basic class ("logical") with no
slots
## Error:  object 'dat.tidy' not found
```

```
## Error:  object 'dat.tidy' not found
## Error:  object 'dat.tidy' not found
## Error:  object 'dat.tidy' not found
## Error:  object 'dat.tidy' not found
## Error:  object 'dat.tidy.melt' not found
## Error:  object 'v' not found
## Error:  object 'v' not found
## Error:  object 'v' not found
## Error:  object 'v' not found
## Error:  object 'v' not found
## Error:  object 'dat.tidy.melt' not found
## Error:  object 'final' not found
## Error:  object 'tab2' not found
## Error:  object 'tab2' not found
```

The Census Bureau has historically provided information in the form of counts. However, the nominal frequency of variables may not hold much meaning as different geographic entities contain different populations. For example, in table **??** 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.

```
## Error:  trying to get slot "estimate" from an object of a basic class ("logical") with no
slots
## Error:  incorrect number of dimensions
## Error:  arguments imply differing number of rows:  0, 1
## Error:  object 'univ' not found
## Error:  object 'univ' not found
## Error:  object 'univ' not found
## Error:  object 'univ' not found
## Error:  object 'univ' not found
## Error:  object 'univ2' not found
## Error:  object 'tab1' not found
## Error:  object 'tab1' not found
```

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. In table **??**, we modify the above table and calculate the percentage of males/females for each combination of school and age group for each state.

```
## Error:  object 'univ2' not found
## Error:  object 'univ3' not found
## Error:  object 'tot' not found
## Error:  object 'univ4' not found
## Error:  object 'toPrint' not found
## Error:  object 'tab1' not found
```

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: $SE = \frac{MOE}{1.645}$ 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:

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

In calculating proportions we must often add different estimates together for different labels. Combining the standard error for a sum of estimates: $SE(A+B) = \sqrt{SE(A)^2 + SE(B)^2}$.

Due to how the data is provided by the Census Bureau compared with the subset of the public use microsample that is used in Stangl, Rundel, and Morgan [3] there are limited combinations of variables available. One of the exercises in Stangl, Rundel, and Morgan [3] is to compare health insurance status by race. This exercise using the public use microsample is straight forward and requires a single line of code using the plyr [5] 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 = sum(HealthInsurance)/length(Race))
print(acsdat_by_race)

##    Race insurance_pct
## 1 asian        0.8429
## 2 black        0.8113
## 3 other        0.7302
## 4 white        0.8804
```

This same calculation can be done from data provided by the online API. Recall that the data at the individual data is not available with the online API but it is still to calculate overall insurance rates using state or country level data. An additional step is required because the online API separates race into eight separate tables. Due to the structure of the data from the ACS package, without putting the data in tidy form, finding the percentage with health insurance requires summing columns and element-wise division between columns.

```
## Error:  trying to get slot "estimate" from an object of a basic class ("logical") with no
slots
## Error:  trying to get slot "estimate" from an object of a basic class ("logical") with no
slots
## Error:  trying to get slot "estimate" from an object of a basic class ("logical") with no
slots
## Error:  trying to get slot "estimate" from an object of a basic class ("logical") with no
slots
```

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

## Error:  object 'health_white' not found

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

## Error:  object 'health_asian' not found

b_pct = calc_pct(health_black)

## Error:  object 'health_black' not found

o_pct = calc_pct(health_other)

## Error:  object 'health_other' not found

w_pct = calc_pct(health_white)

## Error:  object 'health_white' not found
```

The column headers of the health insurance by race dataset from the online API reveal one big issue: the available variables are strictly set and cannot be easily changed. The original exercise was to calculate health insurance coverage percentage for each race and age group is not a variable of interest. In the subset of the public use microsample in Stangl, Rundel, and Morgan [3] there were additional variables but because this dataset is in a tidy format it is possible to easily subset the data by row. The dataset from the online API has different levels of factors in each column, age and health insurance status, and requires an extra step to sum health insurance numbers over the different levels of the age variable. 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. For each of the different datasets for races we have summed up health insurance coverage numbers over the different age groups and found the overall health coverage percentage numbers:

```
## Error:  object 'a_pct' not found
## Error:  object 'b_pct' not found
## Error:  object 'o_pct' not found
## Error:  object 'w_pct' not found
```

Since the available variables are strictly set, it is not possible to subset further. From the dataset in Stangl, Rundel, and Morgan [3], it is possible to subset further: health insurance percentage by race and gender. Again, this calculation 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 = sum(HealthInsurance)/length
print(acsdat_gender_race)
```

```
##    Race Sex insurance_pct
## 1 asian   0        0.9333
## 2 asian   1        0.7750
## 3 black   0        0.8833
## 4 black   1        0.7174
## 5 other   0        0.7273
## 6 other   1        0.7333
## 7 white   0        0.8720
## 8 white   1        0.8905
```

The dataset from the online API for health insurance by age and race does not include gender and because the available variables are strictly set for each table it is not possible to calculate health insurance coverage percentage by race and gender. Our only option is to look at other tables and find one that includes all combinations of variables of interest. There is another table available from the online API for health insurance: table number $B27001$. This table is health insurance coverage status by age and gender. It is possible to use both of these tables on health insurance to calculate estimates of health insurance coverage by gender and race but constructing exact figures is not possible.

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

```
## Error:  Error opening SHP file
## Error:  object 'ny.tracts' not found
## Error:  object 'ny.tracts' not found
## Error:  object 'ny.points' not found
```

```
## Error:  trying to get slot "estimate" from an object of a basic class ("logical") with no
slots
## Error:  non-character argument
## Error:  object of type 'closure' is not subsettable
## Error:  object of type 'closure' is not subsettable
## Error:  object 'ny.df' not found
## Error:  object 'ny.df' not found
## Error:  error in evaluating the argument 'x' in selecting a method for function 'merge':
Error:  object 'ny.df' not found
## Error:  object 'final' not found
## Error:  error in evaluating the argument 'X' in selecting a method for function 'apply':
Error:  object 'final' not found
## Error:  object 'final' not found
```

```
## Error:  Error opening SHP file
## Error:  object 'ny.shape' not found
## Error:  object 'ny.shape' not found
## Error:  object 'ny.points' not found
## Error:  object 'ny2.df' not found
```

7 Conclusion

8 Appendix

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] 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.

[3] 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. eprint: http://www.tandfonline.com/doi/pdf/10.1080/09332480.2013.772392. URL: http://www.tandfonline.com/doi/abs/10.1080/09332480.2013.772392.

[4] 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.

[5] 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/.

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