# Census Subject Tables

ANALYZING US CENSUS DATA IN PYTHON

**Lee Hachadoorian**

Asst. Professor of Instruction, Temple University

# Census Data Products

- **Decennial Census of Population and Housing**
- **American Community Survey (annual)**
- Current Population Survey (monthly)
- Economic Survey (5 years)
- Annual Survey of State and Local Government Finances

# Course Prerequisites

- Lists
- Dictionaries
- Package imports
- Control flow, looping
- List comprehensions
- `pandas` data frames

# Introduction to Census Topics

**Decennial Census of Population and Housing**

- Demographics (age, sex, race, family structure)
- Housing Occupancy and Ownership (vacant/occupied, rent/own)
- Group Quarters Population (prisons, college dorms)

**American Community Survey**

- Educational Attainment
- Commuting (mode, time leaving, time travelled)
- Disability Status

# Structure of a Subject Table

**P5. HISPANIC OR LATINO ORIGIN BY RACE [17]**

*Universe: Total population*

| | |
|---|---|
| Total: | P0050001 |
| Not Hispanic or Latino: | P0050002 |
| White alone | P0050003 |
| Black or African American alone | P0050004 |
| American Indian and Alaska Native alone | P0050005 |
| Asian alone | P0050006 |
| Native Hawaiian and Other Pacific Islander alone | P0050007 |
| Some Other Race alone | P0050008 |
| Two or More Races | P0050009 |
| Hispanic or Latino: | P0050010 |
| White alone | P0050011 |
| Black or African American alone | P0050012 |

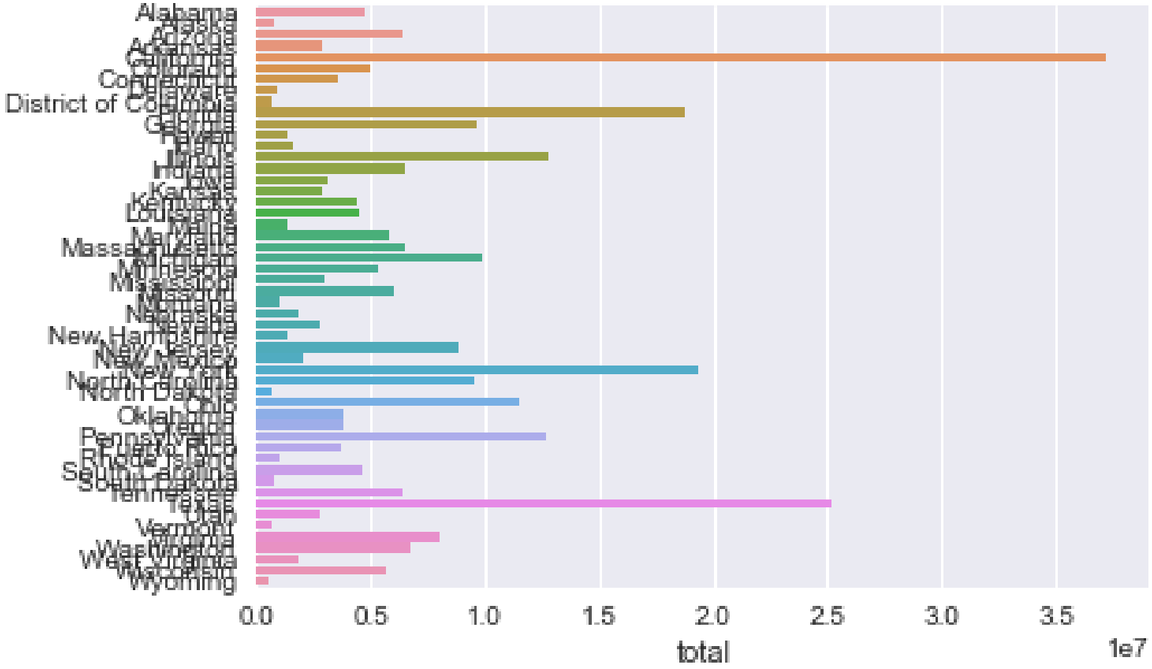
# Subject Table to Data Frame

```
states.head()
```

```
                total  ...  hispanic_multiracial
Alabama       4779736  ...                 10806
Alaska         710231  ...                  6507
Arizona       6392017  ...                103669
Arkansas      2915918  ...                 11173
California   37253956  ...                846688

[5 rows x 17 columns]
```

# Basic Data Visualization

```
import seaborn as sns

sns.set()

sns.barplot(
    x = "total",
    y = states.index,
    data = states
    )
```




Going further: **Data Visualization with Seaborn**

# Let's practice!

ANALYZING US CENSUS DATA IN PYTHON

# Using the Census API

ANALYZING US CENSUS DATA IN PYTHON

**Lee Hachadoorian**

Asst. Professor of Instruction, Temple University

# Structure of a Census API Request

```
https://api.census.gov/data/2010/dec/sf1?get=NAME,P001001,&for=state:*
```

# Structure of a Census API Request

`https://api.census.gov/data/2010/dec/sf1?`

- Base URL
  - Host = `https://api.census.gov/data`
  - Year = `2010`
  - Dataset = `dec/sf1`

# Structure of a Census API Request

`https://api.census.gov/data/2010/dec/sf1?get=NAME,P001001,&for=state:*`

- Base URL
  - Host = `https://api.census.gov/data`
  - Year = `2010`
  - Dataset = `dec/sf1`
- Parameters
  - `get` - List of variables
  - `for` - Geography of interest

# The requests Library

```
import requests

HOST = "https://api.census.gov/data"
year = "2010"
dataset = "dec/sf1"

base_url = "/".join([HOST, year, dataset])

predicates = {}

get_vars = ["NAME", "AREALAND", "P001001"]

predicates["get"] = ",".join(get_vars)

predicates["for"] = "state:*"

r = requests.get(base_url, params=predicates)
```

# Examine the Response

```
print(r.text)
```

```
[["NAME","AREALAND","P001001","state"],
["Alabama","131170787086","4779736","01"],
["Alaska","1477953211577","710231","02"],
["Arizona","294207314414","6392017","04"],
...
```

# Response Errors

```
print(r.text)
```

```
error: unknown variable 'nonexistentvariable'
```

# Create User-Friendly Column Names

```
print(r.json()[0])
```

```
['NAME', 'AREALAND', 'P001001', 'state']
```

Create easy to remember column names using snake_case:

```
col_names = ["name", "area_m2", "total_pop", "state"]
```

# Load into Pandas Data Frame

```
import pandas as pd

df = pd.DataFrame(columns=col_names, data=r.json()[1:])

# Fix data types
df["area_m2"] = df["area_m2"].astype(int)
df["total_pop"] = df["total_pop"].astype(int)

print(df.head())
```

```
        name        area_m2  total_pop state
0    Alabama   131170787086    4779736    01
1     Alaska  1477953211577     710231    02
2    Arizona   294207314414    6392017    04
```

# Find 3 Most Densely Settled States

```
# Create new column
df["pop_per_km2"] = 1000**2 * df["total_pop"] / df["area_m2"]

# Find top 3
df.nlargest(3, "pop_per_km2")
```

| | name | area_m2 | total_pop | state | pop_per_km2 |
|---|---|---|---|---|---|
| 8 | District of Columbia | 158114680 | 601723 | 11 | 3805.611218 |
| 30 | New Jersey | 19047341691 | 8791894 | 34 | 461.581156 |
| 51 | Puerto Rico | 8867536532 | 3725789 | 72 | 420.160547 |

# Let's practice!

ANALYZING US CENSUS DATA IN PYTHON

# Census Geography

## ANALYZING US CENSUS DATA IN PYTHON

**Lee Hachadoorian**

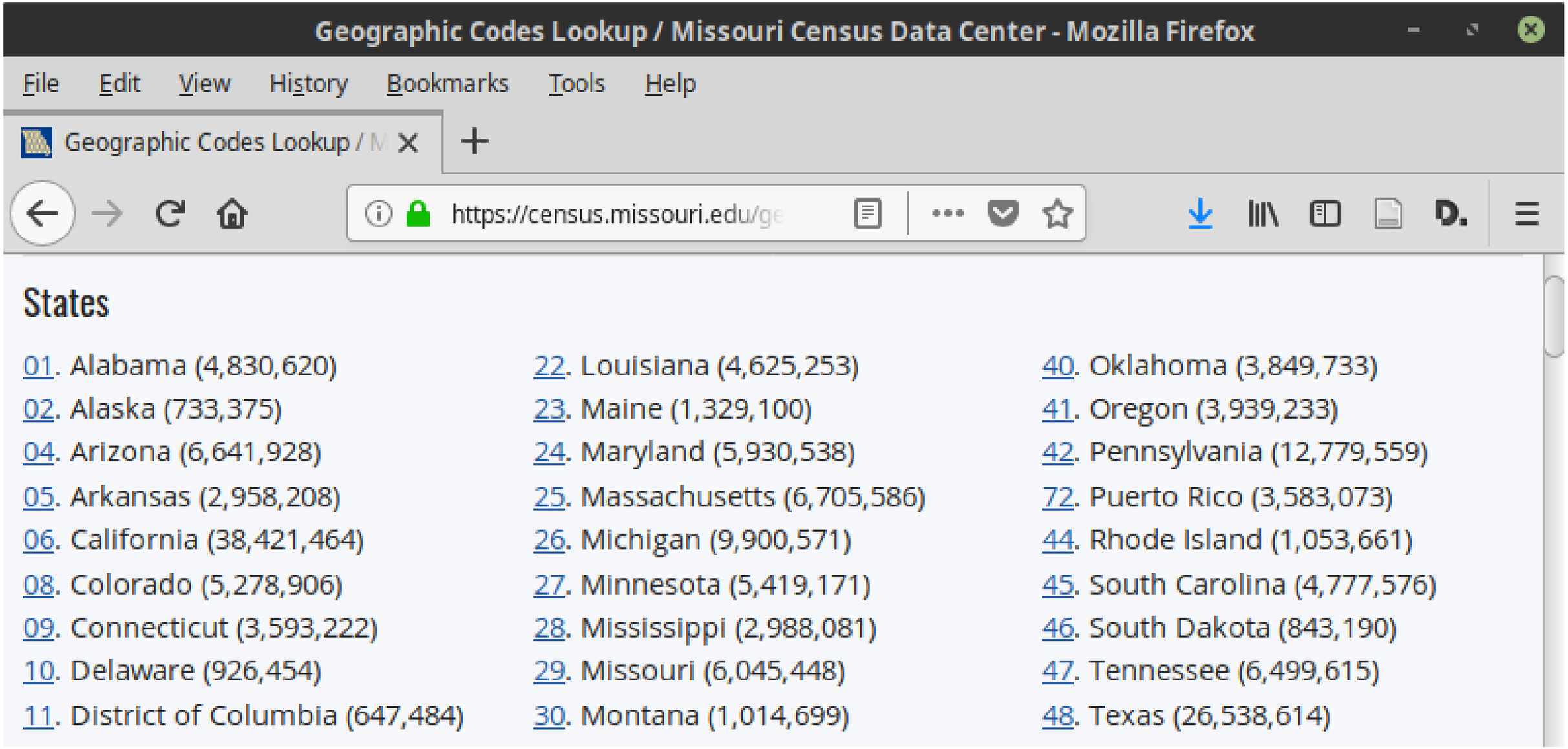
Asst. Professor of Instruction, Temple University

# Request All Geographies

```
import requests

HOST = "https://api.census.gov/data"
year = "2010"
dataset = "dec/sf1"
base_url = "/".join([HOST, year, dataset])

predicates = {}
predicates["get"] = "NAME,P001001"
predicates["for"] = "state:*"
r = requests.get(base_url, params=predicates)
```

# Request Specific Geographies

```
import requests

HOST = "https://api.census.gov/data"
year = "2010"
dataset = "dec/sf1"
base_url = "/".join([HOST, year, dataset])

predicates = {}
predicates["get"] = "NAME,P001001"
predicates["for"] = "state:42"
r = requests.get(base_url, params=predicates)
```




[1] https://census.missouri.edu/geocodes/

# Geographic Entities

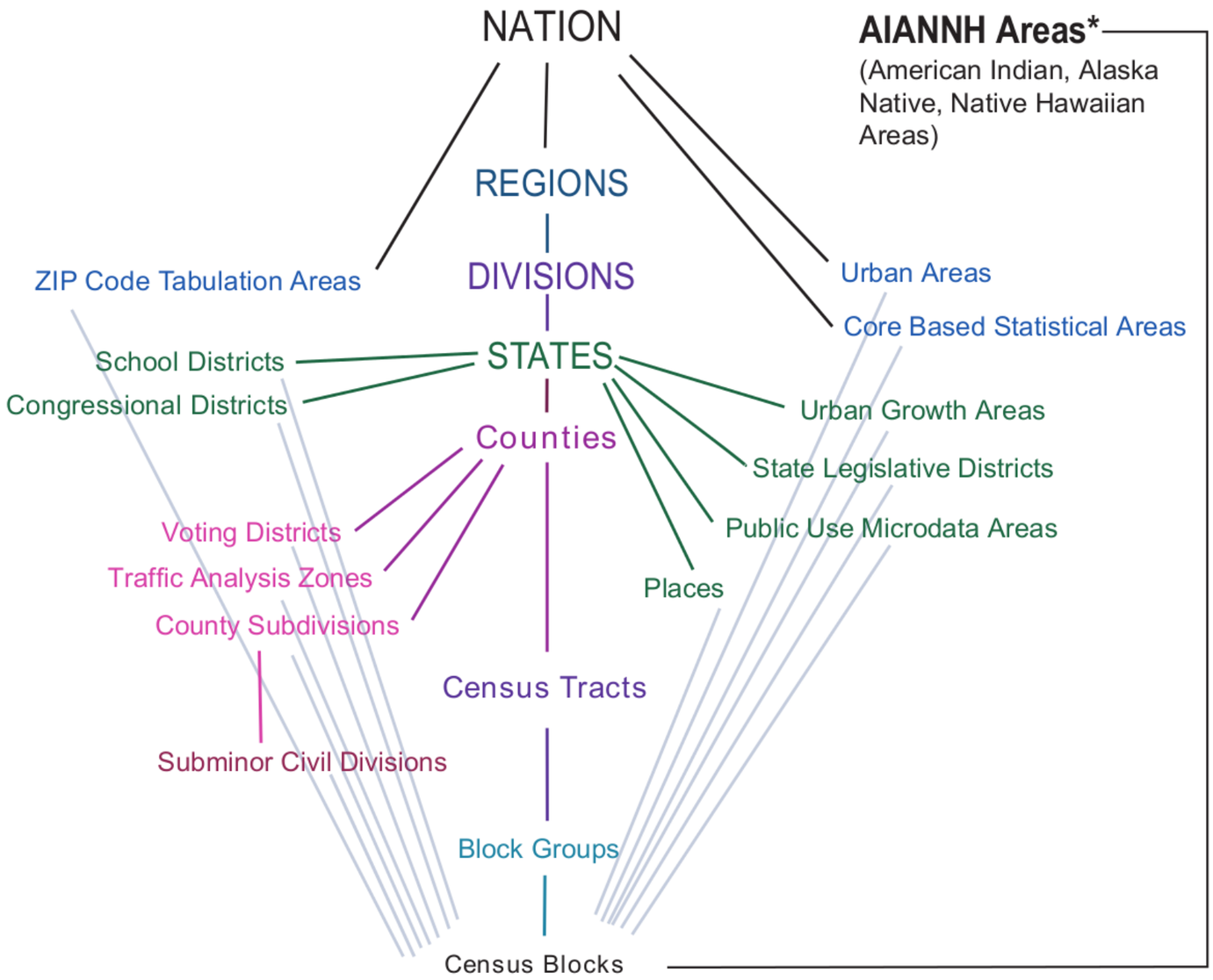
Legal/Administrative

- State
- County
- Congressional Districts
- School Districts
- etc.

Statistical

- Block
- (Census) Tract
- Metropolitan/Micropolitan Statistical Area
- ZIP Code Tabulation Area
- etc.

[1] https://www.census.gov/geo/education/legstat_geo.html


NATION
AIANNH Areas*
(American Indian, Alaska Native, Native Hawaiian Areas)
REGIONS
DIVISIONS
ZIP Code Tabulation Areas
Urban Areas
Core Based Statistical Areas
STATES
School Districts
Congressional Districts
Urban Growth Areas
Counties
State Legislative Districts
Public Use Microdata Areas
Voting Districts
Traffic Analysis Zones
County Subdivisions
Places
Census Tracts
Subminor Civil Divisions
Block Groups
Census Blocks

# The "in" Predicate

Request all counties in specific states:

```
predicates["for"] = "county:*"
predicates["in"] = "state:33,50"
```

Request specific counties in **one** state:

```
predicates["for"] = "county:001,003"
predicates["in"] = "state:33"
```

```
r = requests.get(base_url, params=predicates)
```

# Places

- "An **incorporated place** is established to provide governmental functions for a concentration of people…. An incorporated place usually is a city, town, village, or borough, but can have other legal descriptions."
- "**Census Designated Places (CDPs)** are the statistical counterparts of incorporated places, and are delineated to provide data for settled concentrations of population that are identifiable by name but are not legally incorporated under the laws of the state in which they are located."

Source: https://www.census.gov/geo/reference/gtc/gtc_place.html

| Geography Level | Geography Hierarchy |
|---|---|
| 40 | state |
| 50 | state› county |
| 60 | state› county› county subdivision |
| 101 | state› county› tract› block |
| 140 | state› county› tract |
| 150 | state› county› tract› block group |
| 160 | state› place |

https://api.census.gov/data/2010/dec/sf1/geography.html

# Part Geographies

state› congressional district› county (or part)

```
predicates = {}
predicates["get"] = "NAME,P001001"
predicates["for"] = "county (or part):*"
predicates["in"] = "state:42;congressional district:02"
r = requests.get(base_url, params=predicates)
print(r.text)
```

```
[["NAME","P001001","state","congressional district","county"],
["Montgomery County (part)","36793","42","02","091"],
["Philadelphia County (part)","593484","42","02","101"]]
```

# Let's practice!

ANALYZING US CENSUS DATA IN PYTHON