# Analyzing Census Data Using R

*Corey S. Sparks, Ph.D.*

*February 18, 2019*

## Contents

## Welcome!

## Structure of workshop

1) About me
2) Types of census data
3) The Census API
4) Using `tidycensus` and `censusapi` to get data on places
5) Using the `ipumsr` package to get data on people

## About me

I am an associate professor in the UTSA Department of Demography and have been at UTSA since 2006.

Research interests include data science, Bayesian methods, education demography and health disparities.

## Types of Census Data

Decennial Census

American Community Survey

County Business Patterns

Population Estimates Program - SAIPE and SAHIE

## Decennial Census Summary File 1

The Census Summary File 1(SF 1) contains the data compiled from the questions asked of all people and about every housing unit.

Population items include sex, age, race, Hispanic or Latino origin, household relationship, household type, household size, family type, family size, and group quarters. Housing items include occupancy status, vacancy status, and tenure (whether a housing unit is owner-occupied or renter-occupied).

SF 1 includes population and housing characteristics for the total population, population totals for an extensive list of race (American Indian and Alaska Native tribes, Asian, and Native Hawaiian and Other Pacific Islander) and Hispanic or Latino groups, and population and housing characteristics for a limited list of race and Hispanic or Latino groups.

The decennial Census summary file 1 ***does not*** contain information on education, socioeconomic conditions or other detailed characteristics of the population.

Up until 2010, the Census bureau surveyed 1 out of every 6 households to measure these characteristics, which was referred to as the Census "long form", which was tabulated into a product called the Summary File 3. The year 2000 was the last year this survey was conducted, and beginning in 2005, the American Community Survey replaced the "long form" as the tool to measure socioeconomic characteristics of the population. (US Census Bureau, 2010)

## American Community Survey Summary File

The American Community Survey (ACS) is part of the U.S. Census Bureau's Survey Program and is designed to provide current demographic, social, economic, and housing estimates throughout the decade.

The ACS provides information on more than 40 topics, including educational attainment, language spoken at home, ability to speak English, the foreign born, marital status, migration, and many more. Each year the survey randomly samples around 3.5 million addresses and produces statistics that cover 1-year and 5-year periods for geographic areas in the United States and Puerto Rico, ranging from neighborhoods to congressional districts to the entire nation.

The **ACS 1-year** estimates are published for areas that have populations of 65,000 or more.

The **ACS 5-year** estimates are published for all geographic areas, including Census tracts, block groups, American Indian areas, core-based statistical areas, combined statistical areas, Congressional districts, and state legislative districts.

The American Community Survey Summary File (ACSSF) is a unique data product that includes all the estimates and margins of error from the detailed tables and geographies that are published for the ACS. Data contained in the ACS Summary File cover demographic, social, economic, and housing subject areas.

These data represent totals of population counts, along with the measurement errors in those counts for places, not for individuals, which is the subject of the ACS Public Use Microdata.

## American Community Survey Public Use Microdata

The Public Use Microdata Sample ***(PUMS)*** contains a sample of actual responses to the American Community Survey (ACS). The PUMS dataset includes variables for nearly every question on the survey, as well as many new variables that were derived after the fact from multiple survey responses (such as poverty status).

Each record in the file represents a ***single person, or–in the household-level dataset–a single housing unit***. In the person-level file, individuals are organized into households, making possible the study of people within the contexts of their families and other household members.

PUMS files for an individual year, such as 2015, contain data on approximately one percent of the United States population. PUMS files covering a five-year period, such as 2011-2015, contain data on approximately five percent of the United States population.

The PUMS files are much more flexible than the aggregate data produced in the ACS summary files, though the PUMS also tend to be more complicated to use. Working with PUMS data generally involves downloading large datasets onto a local computer and analyzing the data using statistical software such as R, SPSS, Stata, or SAS.

Since all ACS responses are strictly confidential, many variables in the PUMS files have been modified in order to protect the confidentiality of survey respondents. For instance, particularly high incomes are "top-coded," uncommon birthplace or ancestry responses are grouped into broader categories, and the PUMS files provide a very limited set of geographic variables, including state, metropolitan area and public use microdata area, or PUMA.

While PUMS files contain cases from nearly every town and county in the country, towns and counties (and other low-level geography) are not identified by any variables in the PUMS datasets. The most detailed unit of geography contained in the PUMS files is the ***Public Use Microdata Area (PUMA)***. PUMAs are special non-overlapping areas that partition each state into contiguous geographic units containing no fewer than 100,000 people each. The 2011-2015 5-year ACS PUMS files rely on PUMA boundaries that were drawn by state governments after the 2000 and 2010 Census.

The PUMS data are most easily accessed from the University of Minnesota's ***Integrated Public Use Microdata Series (IPUMS)*** data archive. This data source processes the data produced by the Census Bureau into more easily comparable and readable data files that are available for all years of the decennial Census and the ACS. (Ruggles et al., 2015)

# Using the tidycensus package

They tidycensus is part of the `tidyverse`, and was written and maintained by Dr. Kyle Walker at TCU.

It allows you to dynamically download and map data from the decennial Census and ACS for any level of Census geography, except blocks!

If you want data on places, this is the easiest way to get it.

### Census table names

The Census publishes data for places in ***summary tables***. These follow a pattern for their names, you can find a description of this here. The biggest problem with finding data from the Census is knowing the table name you want.

You can find table names for the ACS here

### What kind of table do you want?

There are several types of tables the Census publishes.

The Detailed tables are very detailed summaries of the data for places, in the 2015 data there were more than 64,000 tables published. These can be a little overwhelming to use, but we'll see an example below

Subject tables take some of the detailed tables and compute summaries of them around certain demographic, social or economic subjects. Basically this is one way to get more data related to a subject without having to know all of the individual detail tables you need.

Data Profile tables contain broad social, economic, housing, and demographic information. The data are presented as both counts and percentages. There are over 2,400 variables in this dataset. These are very useful summaries and what I personally rely on for most of my data extracts.

### Get a Census developer API Key

Obtain one at http://api.census.gov/data/key_signup.html

### Save your API key to your working directory

I recommend you install your API key in your Rprofile, just so you don't have to keep pasting it into your code. To do this, type `tidycensus::census_api_key(key =  "yourkeyhere", install = T)` one time to install your key for use in `tidycensus`.

### Let's do some stuff!

### Look at available ACS variables

As I mentioned above, finding the right table can be a challenge, especially for new data users. `tidycensus` has the `load_variables()` function that will load all of the available tables for a specific table type.

For example, if we are interested in variables from the ACS data profile tables, we can load all available variables then use R to search for what we need.

One of the best ways to search is to use `grep()`, which is a tool for searching for patterns within text, and is ***SUPER USEFUL!***

```r
library(tidycensus)
library(dplyr)
library(sf)
library(ggplot2)
library(censusapi)
```

```r
?load_variables
v15_Profile <- load_variables(year = 2015 , dataset = "acs5/profile",
                              cache = TRUE) #demographic profile tables

#Open the data for examination
View(v15_Profile)

#Search for variables by keywords in the label
v15_Profile[grep(x = v15_Profile$label, "Median household"), c("name", "label")]
```

```
## # A tibble: 2 x 2
##   name       label
##   <chr>      <chr>
## 1 DP03_0062  Estimate!!INCOME AND BENEFITS (IN 2015 INFLATION-ADJUSTED DOL~
## 2 DP03_0062P Percent!!INCOME AND BENEFITS (IN 2015 INFLATION-ADJUSTED DOLL~
```

Also, if you want the names and info for the subject tables, change the `dataset =` argument to `acs5/subject`

```r
v15_subject <- load_variables(year = 2015 ,dataset= "acs5/subject",
                              cache = TRUE) #demographic subject tables
```

Finally, to view variables in the detailed tables, change the `dataset =` argument to `acs5`

```r
v15_detailed <- load_variables(year = 2015 , dataset = "acs5",
                               cache = TRUE) #demographic detail tables
```

If you are interested in variables from the decennial census, change the `dataset =` argument to `sf1` or `sf3` depending on which decennial summary file you want.

## Extract from ACS summary file data profile variables from 2015 for Bexar County, TX Census Tracts

Here is a real example

The data profile tables are very useful because they contain lots of pre-calculated variables.

Here is a query where we extract the median household income in census tracts from the 2015 ACS for Bexar County, Texas. We can also get the spatial data by requesting `geometry=TRUE`. Using `output="wide"` will put each variable in a column of the data set, with each row being a census tract.

```
sa_acs<-get_acs(geography = "tract", state="TX", county = "Bexar",
                year = 2015,
                variables=c( "DP03_0062E") ,
                geometry = T, output = "wide")
```

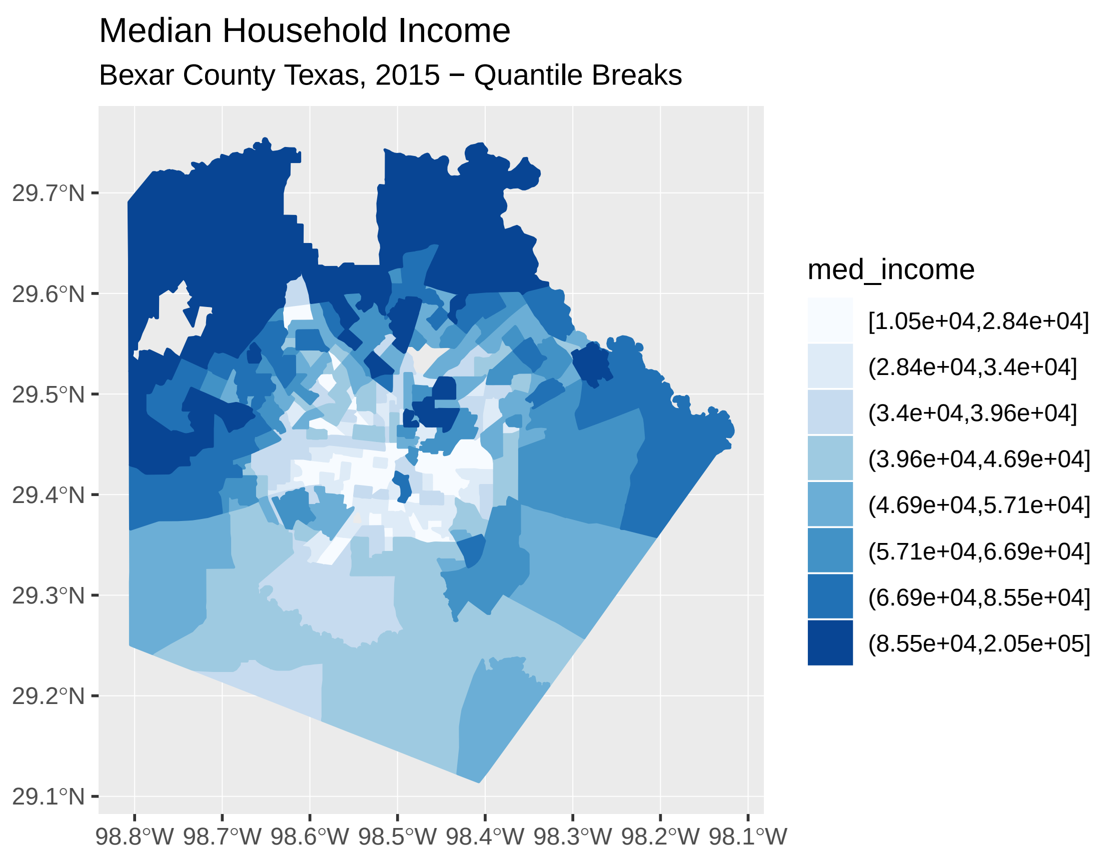
```
## Getting data from the 2011-2015 5-year ACS

## Downloading feature geometry from the Census website.  To cache shapefiles for use in future session

## Using the ACS Data Profile
```

```
#create a county FIPS code - 5 digit
sa_acs$county<-substr(sa_acs$GEOID, 1, 5)

#rename variables and filter missing cases
sa_acs2<-sa_acs%>%
  mutate( medhhinc=DP03_0062E) %>%
  na.omit()

#take a peek at the first few lines of data
head(sa_acs2)
```

We can immediately map these data as well, because `tidycensus` can get you the geography corresponding to your data.

Here, I use the `dplyr` pipe "`%>%`" to feed the data into `ggplot` and map the median household income for each census tract in Bexar County in 2015, using a quantile break system.

```
sa_acs2 %>%
  mutate(med_income=cut(medhhinc,
                        breaks = quantile(medhhinc, na.rm=T, p=seq(0,1,length.out = 9)),include.lowest 
  ggplot( aes(fill = med_income, color = med_income)) +
  geom_sf() +
  ggtitle("Median Household Income",
          subtitle = "Bexar County Texas, 2015 - Quantile Breaks")+
  scale_fill_brewer(palette = "Blues") +
  scale_color_brewer(palette = "Blues")
```



Median Household Income
Bexar County Texas, 2015 − Quantile Breaks
29.7°N
29.6°N
29.5°N
29.4°N
29.3°N
29.2°N
29.1°N
98.8°W 98.7°W 98.6°W 98.5°W 98.4°W 98.3°W 98.2°W 98.1°W
med_income
[1.05e+04,2.84e+04]
(2.84e+04,3.4e+04]
(3.4e+04,3.96e+04]
(3.96e+04,4.69e+04]
(4.69e+04,5.71e+04]
(5.71e+04,6.69e+04]
(6.69e+04,8.55e+04]
(8.55e+04,2.05e+05]

### ACS margins of error

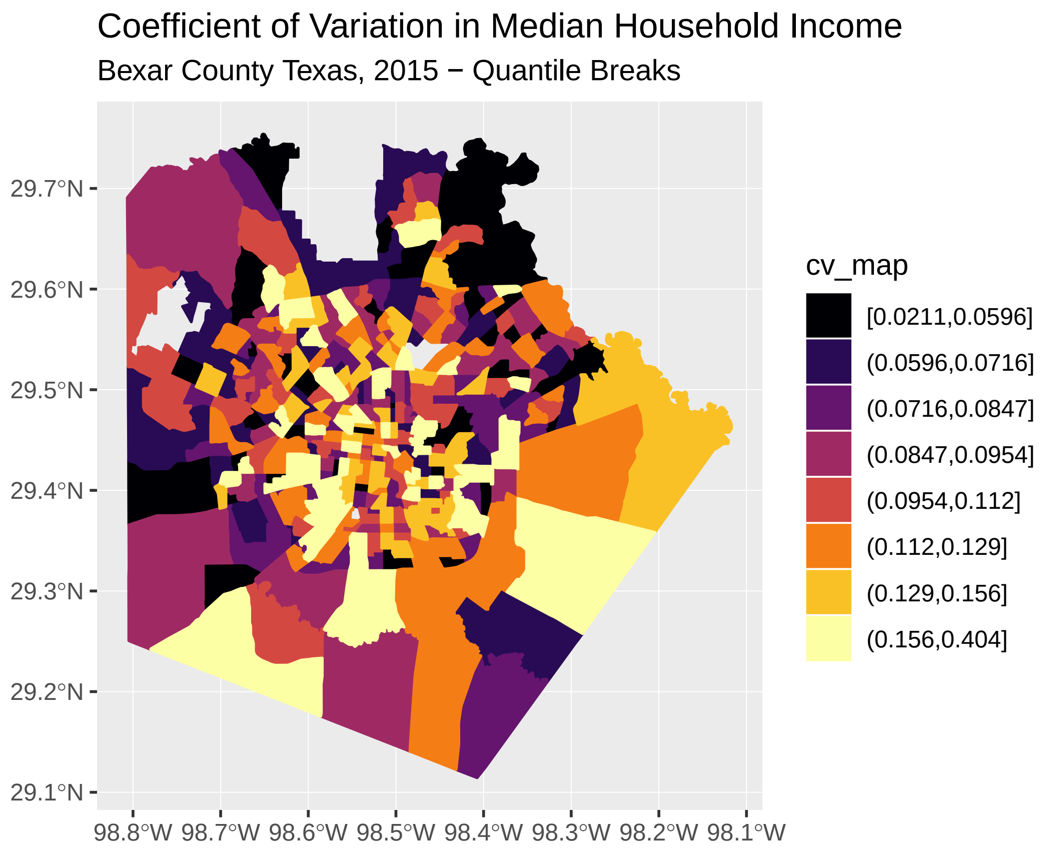
The ACS is a survey and in areas where the sample size is small, the errors in estimates can be quite large. As a way to let users know how uncertain the estimates are for any given area, the Census publishes ***Margins of Error*** for each estimate published. In states, counties and cities, these margins of error are usually small since the overall sample size in those larger areas is large enough to provide more certainty about the population estimates.

In areas that are smaller, tracts and especially block groups, the margins of error can be be very large relative to the estimates. One way to visualize this is to map the ***coefficient of variation*** in the estimates, which is: $CV = \frac{\sigma}{\theta}$, where $\theta$ is the estimate of interest.

### mapping of errors in variables

Here I generate a quantile break for the coefficient of variation in census tract income estimates

```
sa_acs2 %>%
  mutate(cv =(DP03_0062M/1.645)/DP03_0062E)%>%
  mutate(cv_map=cut(cv,
                    breaks = quantile(cv, na.rm=T, p=seq(0,1,length.out = 9)),include.lowest = T))%>%
  ggplot( aes(fill =cv_map, color = cv_map)) +
  geom_sf() +
  ggtitle("Coefficient of Variation in Median Household Income",
          subtitle = "Bexar County Texas, 2015 - Quantile Breaks")+
  scale_fill_viridis_d(option="B")+
  scale_color_viridis_d(option="B")
```



### Metro area incomes

Here is another example where we get data for metro/micropolitan areas in the US. I use a detailed table request this time.

```r
v15_acs<- load_variables(year = 2015 , dataset = "acs5",
                         cache = TRUE) #regular ACS profile tables

View(v15_acs)

#Search for variables by keywords in the label
v15_acs[grep(x = v15_Profile$label, "Median household"), c("name", "label")]

metro<-get_acs(geography = "metropolitan statistical area/micropolitan statistical area",
               year = 2015,
                variables=c( "B19013_001E") ,
                geometry = F, output = "wide")
```

```
## Getting data from the 2011-2015 5-year ACS
```

For this geography, `tidycensus` won't download the geometrys automatically (maybe in a later release), so we use Kyle Walker's other library `tigris` for downloading Census geographic data.

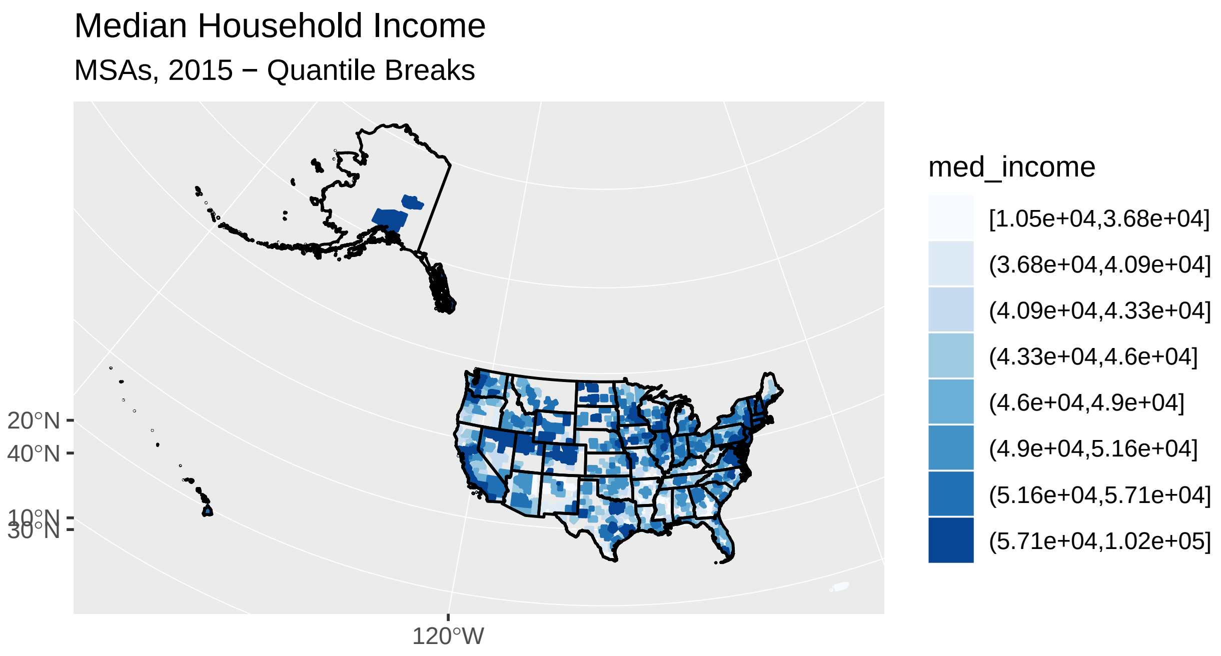
```r
library(tigris)
```

```
## To enable
## caching of data, set `options(tigris_use_cache = TRUE)` in your R script or .Rprofile.

##
## Attaching package: 'tigris'

## The following object is masked from 'package:graphics':
##
##     plot
```

```r
options(tigris_class = "sf") #for use with ggplot2

met_geo<-core_based_statistical_areas(cb=T, year = 2015)

#Filter out territories
sts<-states(cb = T, year = 2015)%>%
  filter(!STATEFP%in%c( "60", "66", "69", "72", "78"))

#merge geographic data to table data
met_join<-geo_join(met_geo, metro, by="GEOID")
```

```
## Warning: st_crs<- : replacing crs does not reproject data; use st_transform
## for that
```

```r
#rename variables and filter missing cases
met_join<-met_join%>%
  mutate( medhhinc=B19013_001E) %>%
  mutate(med_income=cut(medhhinc,
                        breaks = quantile(medhhinc, na.rm=T, p=seq(0,1,length.out = 9)),include.lowest

##Create our map

map1<-met_join%>%
  st_transform(crs = 102740)%>%
  ggplot( aes(fill = med_income, color = med_income)) +
  geom_sf() +
  ggtitle("Median Household Income",
          subtitle = "MSAs, 2015 - Quantile Breaks")+
  scale_fill_brewer(palette = "Blues") +
  scale_color_brewer(palette = "Blues")+
  geom_sf(data=sts, fill=NA, color="black")

map1
```



If you would like to zoom in, we can use the `mapview` library. This is very useful for teaching and for presentations.

```
library(mapview)

pal <- colorRampPalette(viridisLite::viridis(n=6)) #set colors

mapview(met_join, zcol="med_income", col.regions=pal, legend=T,map.types="OpenStreetMap", layer.name="M

## PhantomJS not found. You can install it with webshot::install_phantomjs(). If it is installed, pleas
```

## Using the censusapi package

The Census has lots of APIs that are available. `tidycensus` is great for accessing the ACS and decennial census, but if you want data from another API, the more general `censusapi` package gives you access to all of the Census APIs

First, since this is a different package, you'll need to add your API key to your `.Renviron` data. You can do this by following the example below:

```
Sys.setenv(CENSUS_KEY="yourkeyhere")
# Reload .Renviron
readRenviron("~/.Renviron")
# Check to see that the expected key is output in your R console
Sys.getenv("CENSUS_KEY")
```

Now we load all of the available APIs into an R object so we can look what's available.

```
apis <- listCensusApis()
View(apis)
```

So, we can see there are a lot. For a concrete example, let's examine the effect of the Patient Protection and Affordable Care Act on uninsurance rates in US states.

For this, we will use the Small Area Health Insurance Estimates program. This program combines data from the ACS and data from individual states on other health insurance programs to produce model-based estimates of the overall rates of insurance and uninsurance for states and counties.

Census maintains a nice website for browsing the data and downloading images of data visualizations, but anything custom can be a process. In R this is no problem.

Below, we first look up the variables available in the SAHIE data, then we do an extract and create a dataset for Texas and California for unsinsurance rates by race/ethnicity. Next, we create a line plot of the uninsurance time series for each state and demographic group.

```
#timeseries/poverty/saipe
sahevars<-listCensusMetadata(name = "timeseries/healthins/sahie", type = "v")
head(sahevars)
```

```
##       name                                                          label
## 1 AGE_DESC                                       Age Category Description
## 2 NUI_LB90       Number Uninsured, Lower Bound for 90% Confidence Interval
## 3    STATE                                                State FIPS Code
## 4  NIC_MOE                                 Number Insured, Margin of Error
## 5  NIPR_PT Number in Demographic Group for Selected Income Range, Estimate
## 6  RACECAT                                                  Race Category
##                concept predicateType group limit          required
## 1      Demographic ID           int   N/A     0              <NA>
## 2 Uncertainty Measure           int   N/A     0              <NA>
## 3       Geographic ID           int   N/A     0              <NA>
## 4 Uncertainty Measure           int   N/A     0              <NA>
## 5            Estimate           int   N/A     0              <NA>
## 6      Demographic ID           int   N/A     0 default displayed
```

```
#SAEPOVRTALL_PT
```

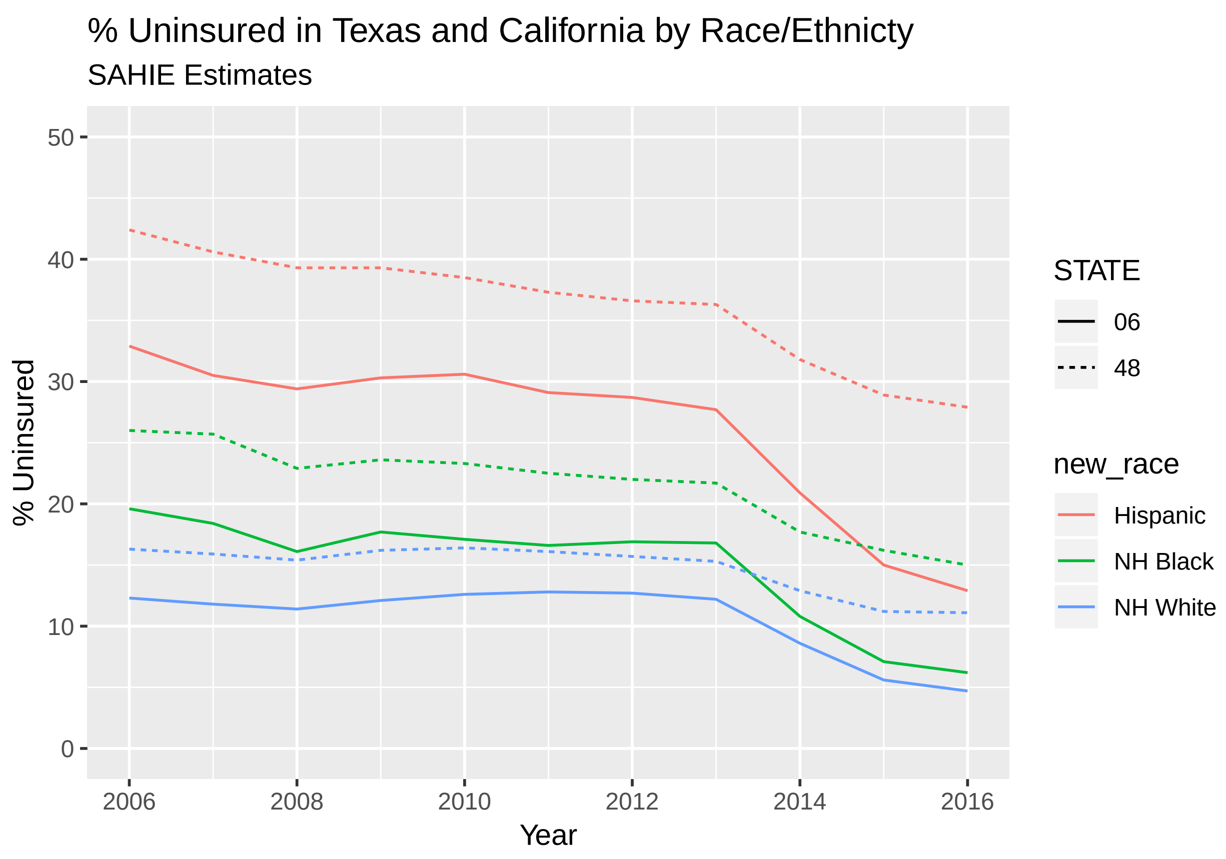
**Get the data for states**

```
ui_rates<-getCensus(name = "timeseries/healthins/sahie",
          vars = c("STATE", "YEAR","RACECAT", "PCTUI_PT"),
          region = "state:*")



head(ui_rates)
```

```
##   state STATE YEAR RACECAT PCTUI_PT
## 1    01    01 2006       0     15.7
## 2    01    01 2007       0     14.6
## 3    01    01 2008       0     15.3
## 4    01    01 2009       0     15.8
## 5    01    01 2010       0     16.9
## 6    01    01 2011       0     16.6
```

**Clean up the data and make a line plot:**

```
ui_rates%>%
  mutate(yr=as.numeric(YEAR), uninsure_rate=as.numeric(PCTUI_PT))%>%
  filter(STATE%in%c("06", "48"), RACECAT!="0")%>%
  select(STATE,  yr,RACECAT, uninsure_rate)%>%
  mutate(new_race=case_when( .$RACECAT=='1' ~ "NH White",
                   .$RACECAT=='2' ~ "NH Black",
                   .$RACECAT=='3' ~ "Hispanic"))%>%
  ggplot(aes(x=yr, y=uninsure_rate))+geom_line(aes(color= new_race, linetype=STATE, group=interaction(S
```



% Uninsured in Texas and California by Race/Ethnicty
SAHIE Estimates
% Uninsured
Year
50
40
30
20
10
0
2006
2008
2010
2012
2014
2016
STATE
06
48
new_race
Hispanic
NH Black
NH White

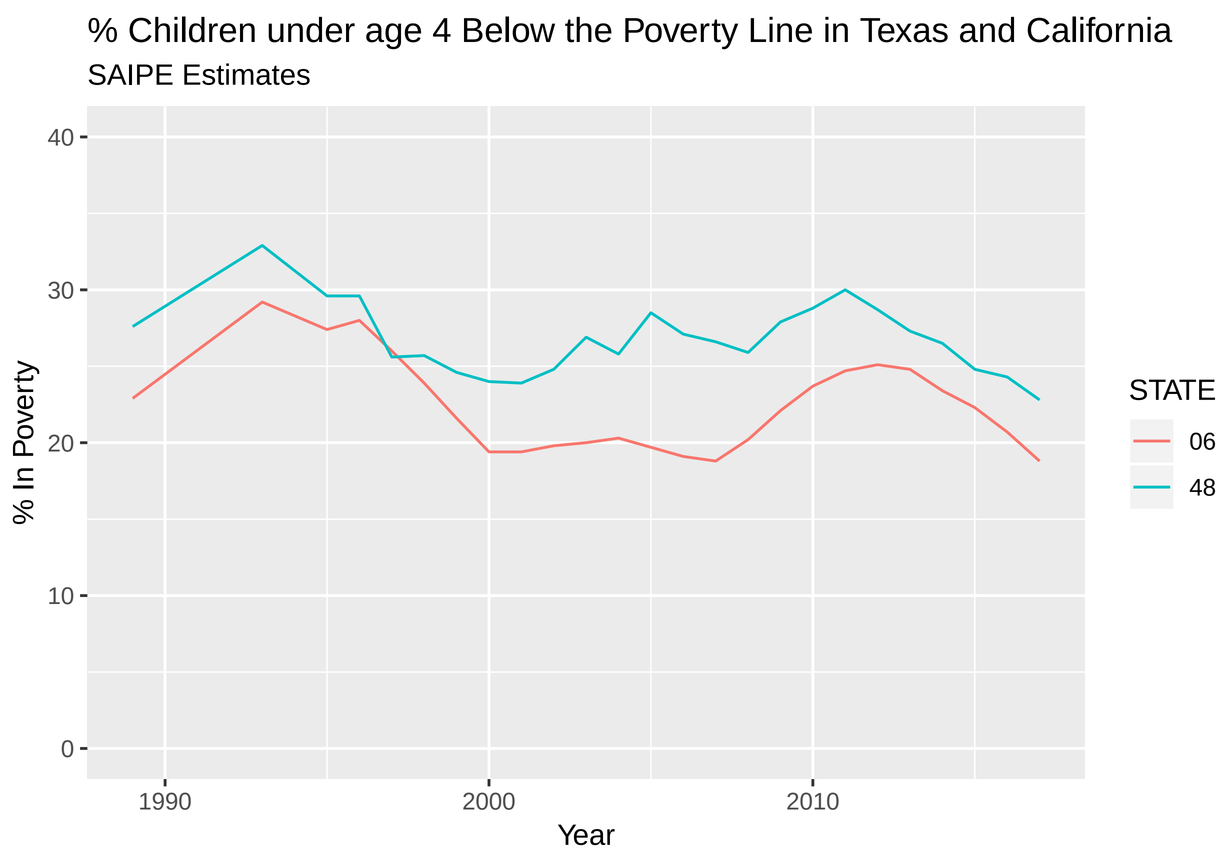
Another big Census estimate program focus on poverty and income, and is called the Small Area Income and Poverty Estimate, or SAIPE program. Like the SAHIE, the SAIPE creates model based estimates of income and poverty for states, counties and school districts on an annual basis.

Below, we do an extract from the SAIPE for Texas and California and create a line plot for the under-4 and total poverty rates for each state over time.

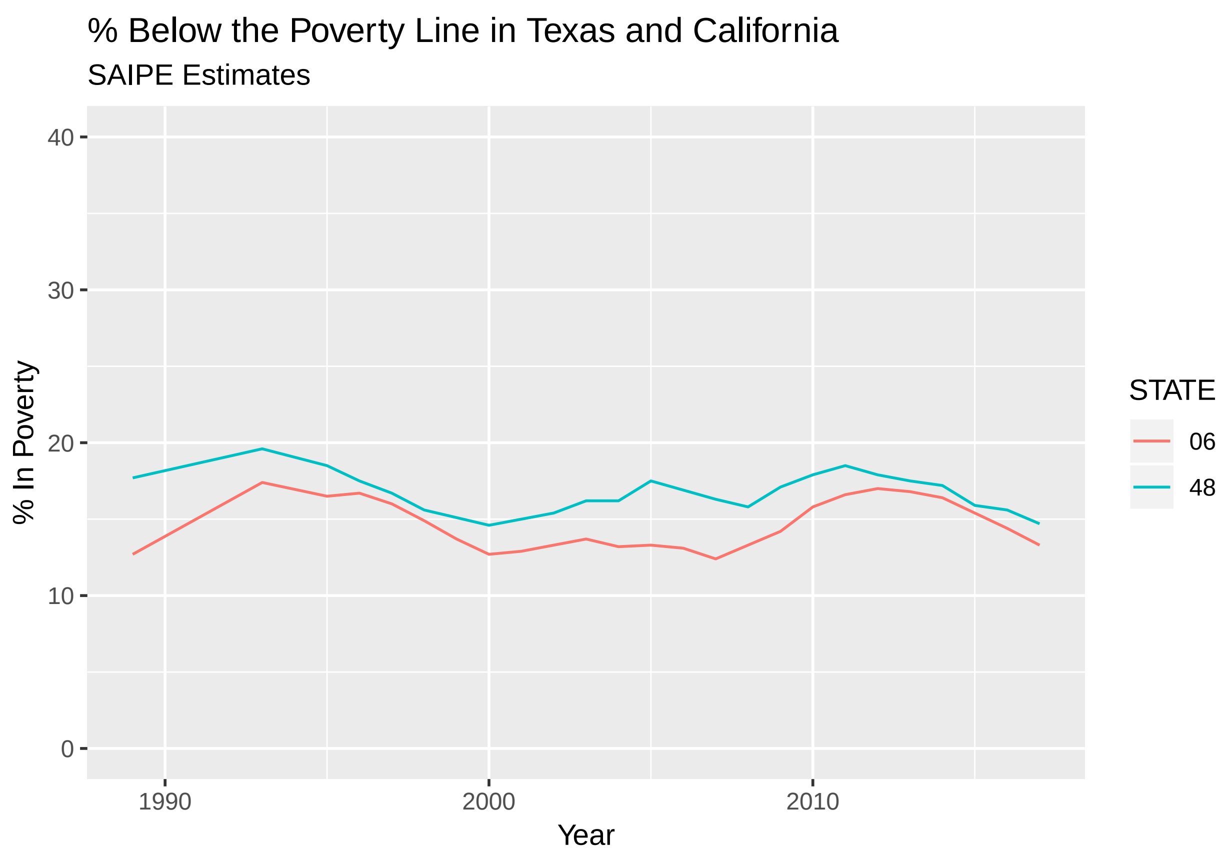
```r
saipvars<-listCensusMetadata(name = "timeseries/poverty/saipe", type = "v")
head(saipvars)
```

```
##                 name
## 1              STATE
## 2  SAEPOVRT0_4_UB90
## 3   SAEPOV5_17R_MOE
## 4       SAEMHI_UB90
## 5 SAEPOVRT0_17_LB90
## 6               YEAR
##                                                                 label
## 1                                                     FIPS State Code
## 2  Ages 0-4 in Poverty, Rate Upper Bound for 90% Confidence Interval
## 3             Ages 5-17 in Families in Poverty, Count Margin of Error
## 4    Median Household Income Upper Bound for 90% Confidence Interval
## 5 Ages 0-17 in Poverty, Rate Lower Bound for 90% Confidence Interval
## 6                                                       Estimate Year
##                  concept group limit predicateType required
## 1 Selectable Geographies   N/A     0          <NA>     <NA>
## 2    Uncertainty Measure   N/A     0           int     <NA>
## 3    Uncertainty Measure   N/A     0           int     <NA>
## 4    Uncertainty Measure   N/A     0           int     <NA>
## 5    Uncertainty Measure   N/A     0           int     <NA>
## 6       Reference Period   N/A     0           int     <NA>
```

```r
#SAEPOVRTALL_PT

p_rates<-getCensus(name = "timeseries/poverty/saipe",
          vars = c("STATE", "YEAR", "SAEPOVRTALL_PT", "SAEPOVRT0_4_PT"),
          region = "state:*")

p_rates%>%
  mutate(yr=as.numeric(YEAR), tot_pov_rate=as.numeric(SAEPOVRTALL_PT), child04_poverty=as.numeric(SAEPOV
  filter(STATE%in%c("06", "48"))%>%
  select(STATE,  yr, tot_pov_rate, child04_poverty)%>%
  ggplot(aes(x=yr, y=child04_poverty))+geom_line(aes(color= STATE, group=STATE))+ylim(low=0,high=40)+yla
```



```
p_rates%>%
  mutate(yr=as.numeric(YEAR), tot_pov_rate=as.numeric(SAEPOVRTALL_PT), child04_poverty=as.numeric(SAEPOV
  filter(STATE%in%c("06", "48"))%>%
  select(STATE,  yr, tot_pov_rate, child04_poverty)%>%
  ggplot(aes(x=yr, y=tot_pov_rate))+geom_line(aes(color= STATE, group=STATE))+ylim(low=0,high=40)+ylab(
```


% Below the Poverty Line in Texas and California
SAIPE Estimates
% In Poverty
40
30
20
10
0
1990
2000
2010
Year
STATE
06
48

## Using the ipumsr package

The previous examples focused on getting data for places from several Census program. This section focuses on getting data on *individuals*.

Each year, when the ACS is done, or on a decennial year when the Census is carried out, each individual survey is published in a ***Public Use Microdata*** file. For example, in the 2011 - 2016 ACS 5 year data release, there were 15,681,972 persons in the data, while the 1 year file contained 3,156,487 persons.

*So, you can get a lot of data!*

To get access to the microdata, you can get the files directly from Census, or use the data that has been cleaned and homogenized by the IPUMS project at the University of Minnesota Population Center.

In addition to the ACS, you can get decennial Census data going back to the 1790 Census, international census data for nearly 100 countries, Current Population Survey data, Demographic and Health Survey data, the National Historic GIS, and more.

Basically, you can spend your entire career working with the data at the IPUMS project.

The benefit to using IPUMS is that you can download multiple years of data at a single time, with as many or as few variables as you want, for places you want. Moreover, you can get the data in multiple formats (SAS, Stata, SPSS, and R), with a fully formatted codebook for everything.

The staff at IPUMS have created a R package to read in IPUMS data (and they're working on an API too!!) called `ipumsr`. It will read data from any IPUMS project.

Here is a link to an introduction to the package written by IPUMS staff.

Here is a CPS example done by IPUMS staff and a NHGIS example.

### Getting your own IPUMS data

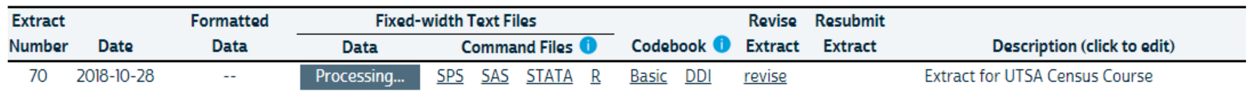
First create an account with the IPUMS project, let them know who you are and what you will be using their data for, and most importantly that you will not be using it for EVIL!

Browse the IPUMS data source of your choice

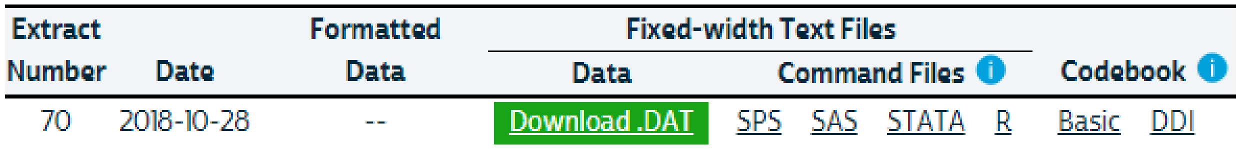
Submit your data extract request

Wait. . . .

Download the data file and the XML file, and save them somewhere you will remember. To use the `ipumsr` package, download the DDI codebook and the .DAT unformatted data file. Otherwise, we could download the stata or SPSS format files and read them in using the `haven` library, easy peasy.



| Extract Number | Date | Formatted Data | Fixed-width Text Files Data | Fixed-width Text Files Command Files | Codebook | Revise Extract | Resubmit Extract | Description (click to edit) |
|---|---|---|---|---|---|---|---|---|
| 70 | 2018-10-28 | -- | Processing... | SPS SAS STATA R | Basic DDI | revise | | Extract for UTSA Census Course |



| Extract Number | Date | Formatted Data | Fixed-width Text Files Data | Fixed-width Text Files Command Files | Codebook |
|---|---|---|---|---|---|
| 70 | 2018-10-28 | -- | Download .DAT | SPS SAS STATA R | Basic DDI |

Then use `ipumsr`!

```
if (!require("ipumsr")) stop("Reading IPUMS data into R requires the ipumsr package. It can be installe
```

```
## Loading required package: ipumsr
```

```r
ddi <- read_ipums_ddi("~/Google Drive/talks/usa_00071.xml")
data <- read_ipums_micro(ddi)


names(data)
```

So, now we have the 2016 ACS 1 year data read into memory in R, and we could do anything we want now.

I use these data for lots of reasons, but I'll show a few general things that you need to do in order to use them.

### Recoding IPUMS variables

***Survey data are a mess!*** They have missing data codes, top codes, numerical values when you want categorical values, and vice versa. Generally, you have to recode variables in order to have them be useful.

Here, I recode several things into standards used in social demographic research.

```r
names(data)<-tolower(names(data))
data<-haven::zap_labels(data)
library(car)
```

```
## Loading required package: carData
```

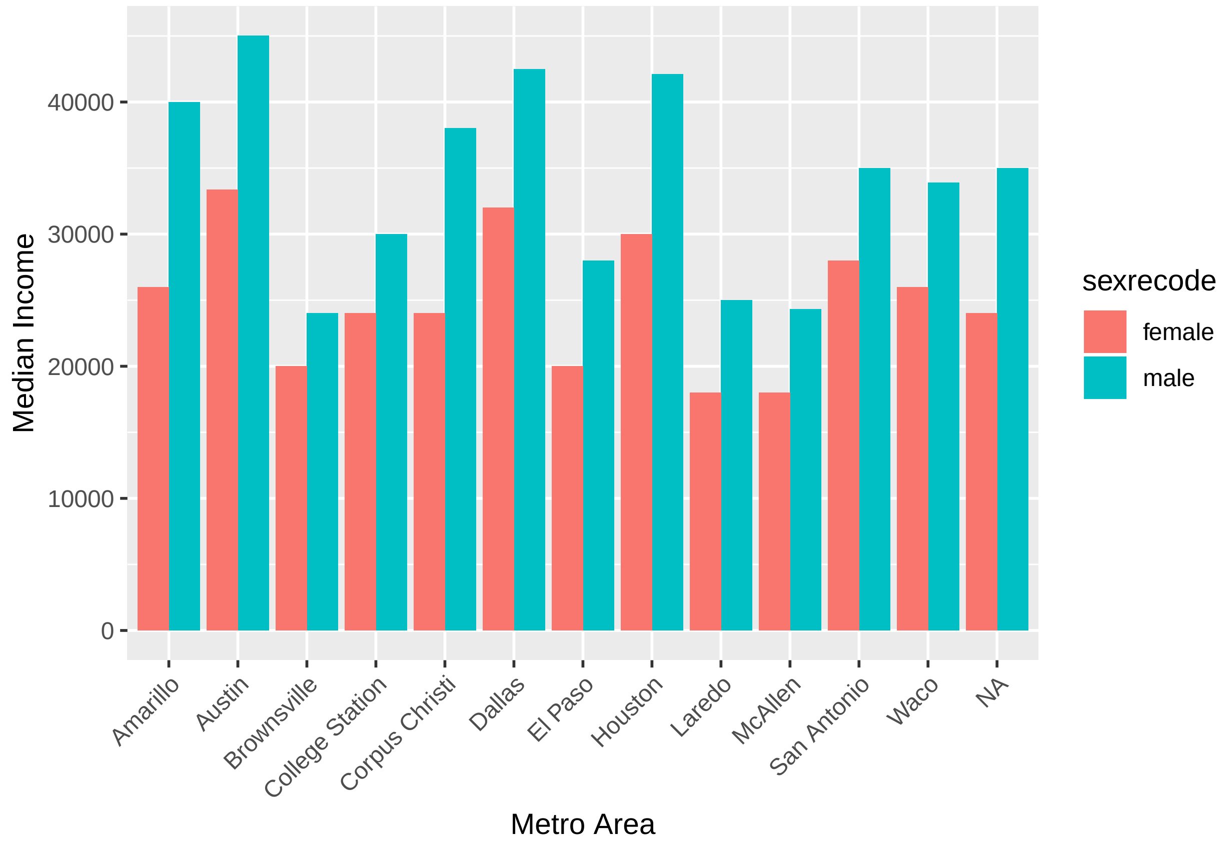
```
##
## Attaching package: 'car'
```

```
## The following object is masked from 'package:dplyr':
##
##     recode
```

```r
#survey weights
data$pwt <- data$perwt/100
data$hwt <- data$hhwt/100

#race/ethnicity
data$hisp <- Recode(data$hispan, recodes = "9=NA; 1:4='Hispanic'; 0='NonHispanic'")
data$race_rec <- Recode(data$race, recodes = "1='White'; 2='Black'; 3='Other'; 4:6='Asian'; 7:9='Other'
data$race_eth <- interaction(data$hisp, data$race_rec, sep = "_")
data$race_eth  <- as.factor(ifelse(substr(as.character(data$race_eth),1,8) == "Hispanic", "Hispanic", a
data$race_eth <- relevel(data$race_eth, ref = "NonHispanic_White")

#Gender
data$male <- ifelse(data$sex == 1,1,0)

#foreign born status
data$usborn <- Recode(data$bpl, recodes = "1:120=1; 121:900=0; else=NA")

#educational attainment
data$educ_level<- Recode(data$educd, recodes = "2:61='0LT_HS';62:64='1_HSD/GED';65:80='2_somecoll';90:1


#Employment status
data$employed <- Recode(data$empstatd, recodes = "10:12=1; 20:22=0; else=NA")

data$inc_wage<- Recode(data$incwage, recodes = "0=NA")
```

So now we have lots of things recoded, we can do some descriptive analysis. For example, let's look at income by gender in Texas cities

```r
data<-data%>%
  filter(labforce==2,statefip=="48", age>18) %>%
  mutate(newwage= ifelse(incwage%in%c(999998,999999), NA, incwage),
         sexrecode=ifelse(sex==1, "male", "female"),
         cityrec = case_when(.$met2013==11100~"Amarillo",
                             .$met2013 == 12420~"Austin",
                             .$met2013==15180 ~"Brownsville",
                             .$met2013== 17780 ~ "College Station",
                             .$met2013== 18580 ~ "Corpus Christi",
                             .$met2013== 21340 ~ "El Paso",
                             .$met2013== 26420 ~ "Houston",
                             .$met2013== 29700~ "Laredo",
                             .$met2013== 32580~ "McAllen",
                             .$met2013== 47380~ "Waco",
                             .$met2013== 41700 ~ "San Antonio",
                             .$met2013== 19100 ~ "Dallas"))

data%>%
  group_by(sexrecode, cityrec)%>%
  summarise(med_income=median(newwage, na.rm=T), n=n())%>%
  ggplot(aes(cityrec, med_income))+geom_bar(aes(fill=sexrecode),position="dodge", stat="identity")+ylab
```



We can do a regression analysis of income differences by various factors.

Before we do this, since the ACS is a complex survey, we cannot simply use regular methods for regression analysis, we must cluster standard errors by sampling cluster and strata and factor in person-weights in the analysis.

This means we must define a formal survey design object.

I restrict the analysis to only Texas and control for various demographic and socioeconomic factors, then compare across cities.

```
data<-data%>%
  filter(employed==1)%>%
  na.omit()

library(survey)
```

```
## Loading required package: grid
```

```
## Loading required package: Matrix
```

```
## Loading required package: survival
```

```
##
## Attaching package: 'survey'
```

```
## The following object is masked from 'package:graphics':
##
##     dotchart
```

```
library(splines)
design<-svydesign(ids=~cluster, strata=~strata, weights=~pwt, data=data)

#Base city difference model
inc_mod_1<- svyglm(log(inc_wage)~cityrec,
                 design = design)

#city difference after controlling for demographic factors
inc_mod<- svyglm(log(inc_wage)~male+usborn+educ_level+bs(age)+cityrec,
                 design = design)

summary(inc_mod_1)
```

```
##
## Call:
## svyglm(formula = log(inc_wage) ~ cityrec, design = design)
##
## Survey design:
## svydesign(ids = ~cluster, strata = ~strata, weights = ~pwt, data = data)
##
## Coefficients:
##                         Estimate Std. Error t value Pr(>|t|)
## (Intercept)            10.290456   0.037672 273.157  < 2e-16 ***
## cityrecAustin           0.213217   0.040130   5.313 1.08e-07 ***
## cityrecBrownsville     -0.272902   0.051877  -5.261 1.44e-07 ***
## cityrecCollege Station -0.158502   0.061568  -2.574   0.0100 *
## cityrecCorpus Christi   0.003041   0.047182   0.064   0.9486
## cityrecDallas           0.167271   0.038339   4.363 1.29e-05 ***
## cityrecEl Paso         -0.250072   0.045551  -5.490 4.04e-08 ***
## cityrecHouston          0.172000   0.038785   4.435 9.24e-06 ***
## cityrecLaredo          -0.238425   0.056807  -4.197 2.71e-05 ***
## cityrecMcAllen         -0.260690   0.047899  -5.442 5.28e-08 ***
## cityrecSan Antonio     -0.038517   0.040055  -0.962   0.3363
## cityrecWaco            -0.119537   0.054446  -2.196   0.0281 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for gaussian family taken to be 1.084504)
##
## Number of Fisher Scoring iterations: 2
```

```
summary(inc_mod)
```

```
##
## Call:
## svyglm(formula = log(inc_wage) ~ male + usborn + educ_level +
##     bs(age) + cityrec, design = design)
##
## Survey design:
## svydesign(ids = ~cluster, strata = ~strata, weights = ~pwt, data = data)
##
## Coefficients:
##                                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)                     8.412593   0.038627 217.793  < 2e-16 ***
## male                            0.427110   0.007469  57.183  < 2e-16 ***
## usborn                          0.189738   0.009944  19.080  < 2e-16 ***
## educ_level1_HSD/GED             0.236177   0.014200  16.632  < 2e-16 ***
## educ_level2_somecoll            0.395092   0.014812  26.674  < 2e-16 ***
## educ_level3_AssocDegree         0.572162   0.017539  32.621  < 2e-16 ***
## educ_level4_bachelordegree      0.876107   0.014677  59.692  < 2e-16 ***
## educ_level4_BAplus_GradDegree   1.129654   0.016038  70.436  < 2e-16 ***
## bs(age)1                        2.661177   0.062358  42.676  < 2e-16 ***
## bs(age)2                        0.714695   0.077986   9.164  < 2e-16 ***
## bs(age)3                        0.422052   0.132645   3.182  0.00146 **
## cityrecAustin                   0.064425   0.033828   1.904  0.05685 .
## cityrecBrownsville             -0.194302   0.044226  -4.393 1.12e-05 ***
## cityrecCollege Station         -0.125369   0.050279  -2.493  0.01265 *
## cityrecCorpus Christi          -0.020340   0.040996  -0.496  0.61978
## cityrecDallas                   0.075777   0.032368   2.341  0.01923 *
## cityrecEl Paso                 -0.230730   0.037979  -6.075 1.25e-09 ***
## cityrecHouston                  0.093941   0.032716   2.871  0.00409 **
## cityrecLaredo                  -0.153910   0.047836  -3.217  0.00129 **
## cityrecMcAllen                 -0.182294   0.039815  -4.579 4.69e-06 ***
## cityrecSan Antonio             -0.085836   0.033841  -2.536  0.01120 *
## cityrecWaco                    -0.102159   0.044747  -2.283  0.02243 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for gaussian family taken to be 0.758806)
##
## Number of Fisher Scoring iterations: 2
```

## Other Resources

### Miktek

To build pdf's you'll need a version of Latex installed, Miktek is a good option

### R-markdown cheat sheets

Rstudio keeps a variety of cheat sheets for various topics, they can be helpful in a pinch

### UCLA Statistical computing help

This page has lots of examples of using R for various types of analysis.

### Other examples

On my Rpubs page I have lots of examples of various types of analysis using R and you can get the data for these on my Github data page

## Thanks!

Let me know how I can help!

[@Coreysparks1](https://twitter.com/CoreySparks1)

Github

Rpubs

UTSA Demography