Web Scraping

Will Doyle

Introduction

Many large web sites host a huge amount of information. This information is encoded and delivered on demand to the user within a web page, which is really just a markup language that a browser can understand. We can take this data and analyze it using R, via a variety of different means. Today we'll cover scraping web tables and interacting via Automated Programming Interfaces.

Ethics and Responsiblity

Many of the tools we'll cover can be quite powerful ways to interact with data stored online and gather it for analysis. Because they're powerful, you need to be careful with them. In particular, try to request information in a way that will not burden the website owners. What constitutes a burden depends on the website. Google, Twitter, Facebook, all of the big websites have protections in place to guard against too many requests and have a huge amount of capacity for taking the requests they grant. Smaller websites may not have either. Always strive to be minimally intrusive: you're usually getting this data for free.

Ways of getting data from the web

We will cover several different ways you can get data from the web

1. Directly downloading web pages via the `url()` command.
2. Scraping simple web tables via `read_html()` and `html_table()` command
3. Interacting with Application Programming Interfaces (APIs) via R libraries that have been designed as "wrappers" for these interfaces, like the awesome `acs` library and the `tigris` library for geographic shapes.
4. Interacting with APIs directly,

Libraries

We will use multiple new libraries today. Among the ones you'll need:

- `rvest` for scraping websites
- `acs` for accessing American Community Survey data via the census API

```
library(tidyverse)
```

```
## Warning: package 'tidyverse' was built under R version 3.5.3
```

```
## -- Attaching packages ----------------------------------------------------------- tidyverse 1.2.1
```

```
## v ggplot2 3.2.1     v purrr   0.3.2
## v tibble  2.1.3     v dplyr   0.8.1
## v tidyr   0.8.3     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.4.0
```

```
## Warning: package 'ggplot2' was built under R version 3.5.3
```

```
## Warning: package 'tibble' was built under R version 3.5.3
```

```
## Warning: package 'tidyr' was built under R version 3.5.3
```

```
## Warning: package 'purrr' was built under R version 3.5.3
## Warning: package 'dplyr' was built under R version 3.5.3
## Warning: package 'stringr' was built under R version 3.5.3
## Warning: package 'forcats' was built under R version 3.5.3
## -- Conflicts ---------------------------------------------------------------- tidyverse_conflicts()
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(rvest)
```

```
## Warning: package 'rvest' was built under R version 3.5.3
## Loading required package: xml2
##
## Attaching package: 'rvest'
## The following object is masked from 'package:purrr':
##
##     pluck
## The following object is masked from 'package:readr':
##
##     guess_encoding
```

```
library(acs)
```

```
## Warning: package 'acs' was built under R version 3.5.3
## Loading required package: XML
## Warning: package 'XML' was built under R version 3.5.3
##
## Attaching package: 'XML'
## The following object is masked from 'package:rvest':
##
##     xml
##
## Attaching package: 'acs'
## The following object is masked from 'package:dplyr':
##
##     combine
## The following object is masked from 'package:base':
##
##     apply
```

```
library(lubridate)
```

```
##
## Attaching package: 'lubridate'
## The following object is masked from 'package:base':
##
##     date
```

```
library(noncensus)
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
```

API keys

You will also need an API key.

- The Census API, available here: https://api.census.gov/data/key_signup.html

Basics of interacting with information stored online

R can understand a web connection via the `url` command. Once that connection is established, we can download whatever we'd like.

```
#Web connections: url
# example
r_home = url("http://www.r-project.org/")
r_home
```

```
## A connection with
## description "http://www.r-project.org/"
## class       "url-wininet"
## mode        "r"
## text        "text"
## opened      "closed"
## can read    "yes"
## can write   "no"
```

```
# Pulling text from a website using readlines
# url of Moby Dick (project Gutenberg)
moby_url = url("http://www.gutenberg.org/files/2701/2701-h/2701-h.htm")
# reading the content (first 1500 lines)
moby_dick = readLines(moby_url, n = 1500)
moby_dick[1205:1220]
```

```
##  [1] "      Call me Ishmael. Some years ago&mdash;never mind how long precisely&mdash;having"
##  [2] "      little or no money in my purse, and nothing particular to interest me on"
##  [3] "      shore, I thought I would sail about a little and see the watery part of"
##  [4] "      the world. It is a way I have of driving off the spleen and regulating the"
##  [5] "      circulation. Whenever I find myself growing grim about the mouth; whenever"
##  [6] "      it is a damp, drizzly November in my soul; whenever I find myself"
##  [7] "      involuntarily pausing before coffin warehouses, and bringing up the rear"
##  [8] "      of every funeral I meet; and especially whenever my hypos get such an"
##  [9] "      upper hand of me, that it requires a strong moral principle to prevent me"
```

```
## [10] "      from deliberately stepping into the street, and methodically knocking"
## [11] "      people&rsquo;s hats off&mdash;then, I account it high time to get to sea as soon"
## [12] "      as I can. This is my substitute for pistol and ball. With a philosophical"
## [13] "      flourish Cato throws himself upon his sword; I quietly take to the ship."
## [14] "      There is nothing surprising in this. If they but knew it, almost all men"
## [15] "      in their degree, some time or other, cherish very nearly the same feelings"
## [16] "      towards the ocean with me."
```

Scraping web tables

When we talk about "scraping" a web table, we're talking about pulling a table that exists on a website and turning it into a usable data frame for analysis. Below, I take the table from `http://en.wikipedia.org/wiki/Marathon_world_record_progression` for men's marathon times and plot the change in speed in m/s as a function of the date that the world record was set.

```
marathon_wiki = "https://en.wikipedia.org/wiki/Marathon_world_record_progression"

# Get the page, pull the tables via html_table
marathon <- read_html(marathon_wiki)%>%html_table(fill=TRUE)

#Men's is the first table
marathon<-tbl_df(data.frame(marathon[[1]]))

#Convert time to seconds
marathon<-marathon%>%
  mutate(Time2=hms(as.character(Time)))%>%
  mutate(Time2=period_to_seconds(Time2))

#Marathons are 42,200 meters long
marathon$speed<-(4.22e4)/marathon$Time2

#Get dates in a usable format usin lubridate::mdy
marathon$date<-mdy(marathon$Date)
```

```
## Warning: 1 failed to parse.
```

```
marathon_men<-marathon
marathon_men$gender="men"
```

```
#women's is the first table


# Get the page, pull the tables via html_table
marathon <- read_html(marathon_wiki)%>%html_table(fill=TRUE)


marathon<-tbl_df(data.frame(marathon[[2]]))

#Convert time to seconds
marathon<-marathon%>%
  mutate(Time2=hms(as.character(Time)))%>%
  mutate(Time2=period_to_seconds(Time2))
```

```
## Warning in .parse_hms(..., order = "HMS", quiet = quiet): Some strings
```

```
## failed to parse, or all strings are NAs
```

```
#Marathons are 42,200 meters long
marathon$speed<-(4.22e4)/marathon$Time2

#Get dates in a usable format usin lubridate::mdy
marathon$date<-mdy(marathon$Date)
```

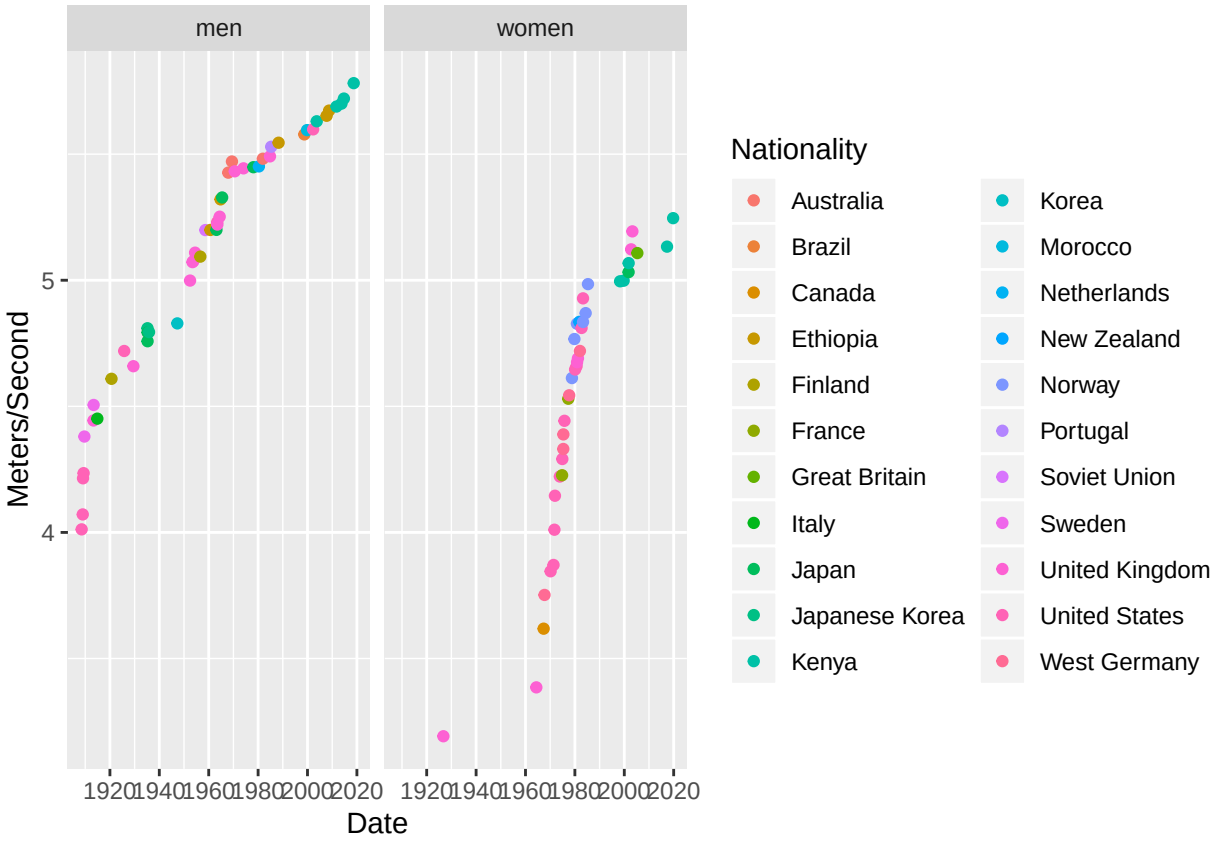
```
## Warning: 2 failed to parse.
```

```
marathon_women<-marathon
marathon_women$gender="women"

marathon_combine=bind_rows(marathon_men,marathon_women)
```

Progression of World Record Marathon Speed in Meters/Second

```
marathon<-marathon%>%mutate(Nationality=fct_reorder(.f=as.factor(Nationality),.x=-speed,.fun = max))

g1<-ggplot(data=marathon_combine,
           aes(y=speed,x=date,
               #Reorder nationality by fastest times
               color=Nationality)
           )

g1<-g1+geom_point()+
      facet_wrap(~gender)+
           xlab("Date")+
           ylab("Meters/Second")

g1
```

```
## Warning: Removed 4 rows containing missing values (geom_point).
```



Quick Exercise Repeat the above analysis for women's world record progression.

Interacting via APIs

Many websites have created Application Programming Interfaces, which allow the user to directly communicate with the website's underlying database without dealing with the intermediary web content. These have been expanding rapdily and are one of the most exciting areas of development in data access for data science.

Today, we'll be working with the American Community Survey from the census. Please go to: `http://www.census.gov/developers/` and click on "Get a Key" to get your census key.

YOU NEED TO PAY ATTENTION TO TERMS OF USE WHEN USING APIS. DO NOT VIOLATE THESE.

With these keys in hand, we can interact with these various databases. Let's say we have information on zip codes for students, and we want to know their likely income level. We can do this by using the American Community Survey API.

Zip Code Level Data from the American Community Survey

The first step is to create a list of all zip codes in Davidson County. We can do this by using another dataset that includes a comprehensive listing of zip codes by county and city.

We start by using the lookup_code from the `tigris` package to get the fips codes for Davidson County in TN (Davidson is home to Vanderbilt).

```
## Look up fips code for county
lookup_code("CA","Los Angeles")
```

```
## [1] "The code for California is '06' and the code for Los Angeles County is '037'."
```

```
state_fips<-"06"
county_stub<-"037"
```

Next, we'll combine the state and county fips into a single object

```
county_fips<-paste0(state_fips,county_stub)
```

```
# Get dataset that matches all zip codes to cities, counties and states.
county_to_zip<-read_csv("http://www2.census.gov/geo/docs/maps-data/data/rel/zcta_county_rel_10.txt")
```

```
## Parsed with column specification:
## cols(
##   .default = col_double(),
##   ZCTA5 = col_character(),
##   COUNTY = col_character()
## )
```

```
## See spec(...) for full column specifications.
```

```
save(county_to_zip,file="county_to_zip.Rdata")

#easier names to work with
names(county_to_zip)<-tolower(names(county_to_zip))

#Just zip codes in selected county
county_to_zip<-county_to_zip%>%
  filter(state==as.numeric(state_fips),county==county_stub)%>%
  select(zcta5,state,county)

#list of zip codes
ziplist<-county_to_zip$zcta5

#City names
data(zip_codes)

city_zip<-zip_codes%>%filter(zip%in%ziplist)%>%select(zip,city)

#Arrange in order
city_zip<-city_zip%>%arrange(as.numeric(zip))
```

Next, we'll turn to the American Community Survey. This includes a large number of tables (available here in excel file form: https://www.census.gov/programs-surveys/acs/technical-documentation/summary-file-documentation.html) that cover many demographic and other characteristics of the population, down to the level of zip codes. We'll use the `acs` package to get two tables for the zip codes we're interested in: levels of education and income. We'll turn these tables into two variables: the proportion of the population with incomes above $75,000, and the proportion of the population with at least a bachelor's degree.

The first step is to get the table from ACS. Below, I submit a request using my key to get table B15002, which contains information on education levels.

```
# Get your own key and save as my_acs_key.txt
my_acs_key<-readLines("my_acs_key.txt",warn = FALSE)
acs_key<-my_acs_key
```

```
# Or just paste it here.
#acs_key<-"<your_acs_key_here>"

#List of tables: https://www.census.gov/programs-surveys/acs/technical-documentation/summary-file-docum
# b15002: education of pop over 25, by sex
# b19001: household income over last 12 months

api.key.install(acs_key, file = "key.rda")

select_zip<-geo.make(zip.code=ziplist)

county_educ=acs.fetch(geography=select_zip,
                      endyear=2016,
                      table.number="B15002",
                      col.names="pretty",verbose=T)
```

```
## Warning in (function (endyear, span = 5, dataset = "acs", keyword, table.name, : acs.lookup for endy
##   (See ?acs.lookup for details)
```

```
## Warning in (function (endyear, span = 5, dataset = "acs", keyword, table.name, : temporarily downloa
##   since this is *much* slower, recommend running
##   acs.tables.install()
```

```
save(county_educ,file="county_educ_la.Rdata")  # <---- this may take a long time :(
acs.colnames(county_educ)
```

```
##  [1] "Sex by Educational Attainment for the Population 25 Years and over: Total:"
##  [2] "Sex by Educational Attainment for the Population 25 Years and over: Male:"
##  [3] "Sex by Educational Attainment for the Population 25 Years and over: Male: No schooling complet
##  [4] "Sex by Educational Attainment for the Population 25 Years and over: Male: Nursery to 4th grade
##  [5] "Sex by Educational Attainment for the Population 25 Years and over: Male: 5th and 6th grade"
##  [6] "Sex by Educational Attainment for the Population 25 Years and over: Male: 7th and 8th grade"
##  [7] "Sex by Educational Attainment for the Population 25 Years and over: Male: 9th grade"
##  [8] "Sex by Educational Attainment for the Population 25 Years and over: Male: 10th grade"
##  [9] "Sex by Educational Attainment for the Population 25 Years and over: Male: 11th grade"
## [10] "Sex by Educational Attainment for the Population 25 Years and over: Male: 12th grade, no diplo
## [11] "Sex by Educational Attainment for the Population 25 Years and over: Male: High school graduate
## [12] "Sex by Educational Attainment for the Population 25 Years and over: Male: Some college, less t
## [13] "Sex by Educational Attainment for the Population 25 Years and over: Male: Some college, 1 or m
## [14] "Sex by Educational Attainment for the Population 25 Years and over: Male: Associate's degree"
## [15] "Sex by Educational Attainment for the Population 25 Years and over: Male: Bachelor's degree"
## [16] "Sex by Educational Attainment for the Population 25 Years and over: Male: Master's degree"
## [17] "Sex by Educational Attainment for the Population 25 Years and over: Male: Professional school
## [18] "Sex by Educational Attainment for the Population 25 Years and over: Male: Doctorate degree"
## [19] "Sex by Educational Attainment for the Population 25 Years and over: Female:"
## [20] "Sex by Educational Attainment for the Population 25 Years and over: Female: No schooling compl
## [21] "Sex by Educational Attainment for the Population 25 Years and over: Female: Nursery to 4th gra
## [22] "Sex by Educational Attainment for the Population 25 Years and over: Female: 5th and 6th grade"
## [23] "Sex by Educational Attainment for the Population 25 Years and over: Female: 7th and 8th grade"
## [24] "Sex by Educational Attainment for the Population 25 Years and over: Female: 9th grade"
## [25] "Sex by Educational Attainment for the Population 25 Years and over: Female: 10th grade"
## [26] "Sex by Educational Attainment for the Population 25 Years and over: Female: 11th grade"
## [27] "Sex by Educational Attainment for the Population 25 Years and over: Female: 12th grade, no dip
## [28] "Sex by Educational Attainment for the Population 25 Years and over: Female: High school gradua
```

```
## [29] "Sex by Educational Attainment for the Population 25 Years and over: Female: Some college, less
## [30] "Sex by Educational Attainment for the Population 25 Years and over: Female: Some college, 1 or
## [31] "Sex by Educational Attainment for the Population 25 Years and over: Female: Associate's degree
## [32] "Sex by Educational Attainment for the Population 25 Years and over: Female: Bachelor's degree"
## [33] "Sex by Educational Attainment for the Population 25 Years and over: Female: Master's degree"
## [34] "Sex by Educational Attainment for the Population 25 Years and over: Female: Professional schoo
## [35] "Sex by Educational Attainment for the Population 25 Years and over: Female: Doctorate degree"
```

Organizing ACS data

The trick with ACS data is organizing it in a way that's going to make sense. For us to get the proportion of indivudals with a college degree or more, we're going to need to take the numbers of people who are in each of the various age levels for education, and then divide by the total number of people in the zip code. Below I include code to calculate the proportion of individuals in each zip code who have at least a bachelor's degree.

```
## Proprtion of individuals at college or above=
## number with college degree/
## total number
prop_coll_above<-divide.acs(numerator=(county_educ[,15]+
                                       county_educ[,16]+
                                       county_educ[,17]+
                                       county_educ[,18]+
                                       county_educ[,32]+
                                       county_educ[,33]+
                                       county_educ[,34]+
                                       county_educ[,35]),
                            denominator=county_educ[,1]
)

#prop with educational attainment=number with level of ed attain/total pop
```

Family Income Data

```
# 19001-- family income
county_income<-acs.fetch(geography=select_zip,
                         endyear = 2016,
                         table.number="B19001",
                         col.names="pretty")
```

```
## Warning in (function (endyear, span = 5, dataset = "acs", keyword, table.name, : acs.lookup for endy
##   (See ?acs.lookup for details)
```

```
## Warning in (function (endyear, span = 5, dataset = "acs", keyword, table.name, : temporarily downloa
##   since this is *much* slower, recommend running
##   acs.tables.install()
```

```
acs.colnames(county_income)
```

```
##  [1] "B19001. Household Income in the Past 12 Months (in 2015 Inflation-Adjusted Dollars): Total:"
##  [2] "B19001. Household Income in the Past 12 Months (in 2015 Inflation-Adjusted Dollars): Less than
##  [3] "B19001. Household Income in the Past 12 Months (in 2015 Inflation-Adjusted Dollars): $10,000 t
##  [4] "B19001. Household Income in the Past 12 Months (in 2015 Inflation-Adjusted Dollars): $15,000 t
##  [5] "B19001. Household Income in the Past 12 Months (in 2015 Inflation-Adjusted Dollars): $20,000 t
##  [6] "B19001. Household Income in the Past 12 Months (in 2015 Inflation-Adjusted Dollars): $25,000 t
```

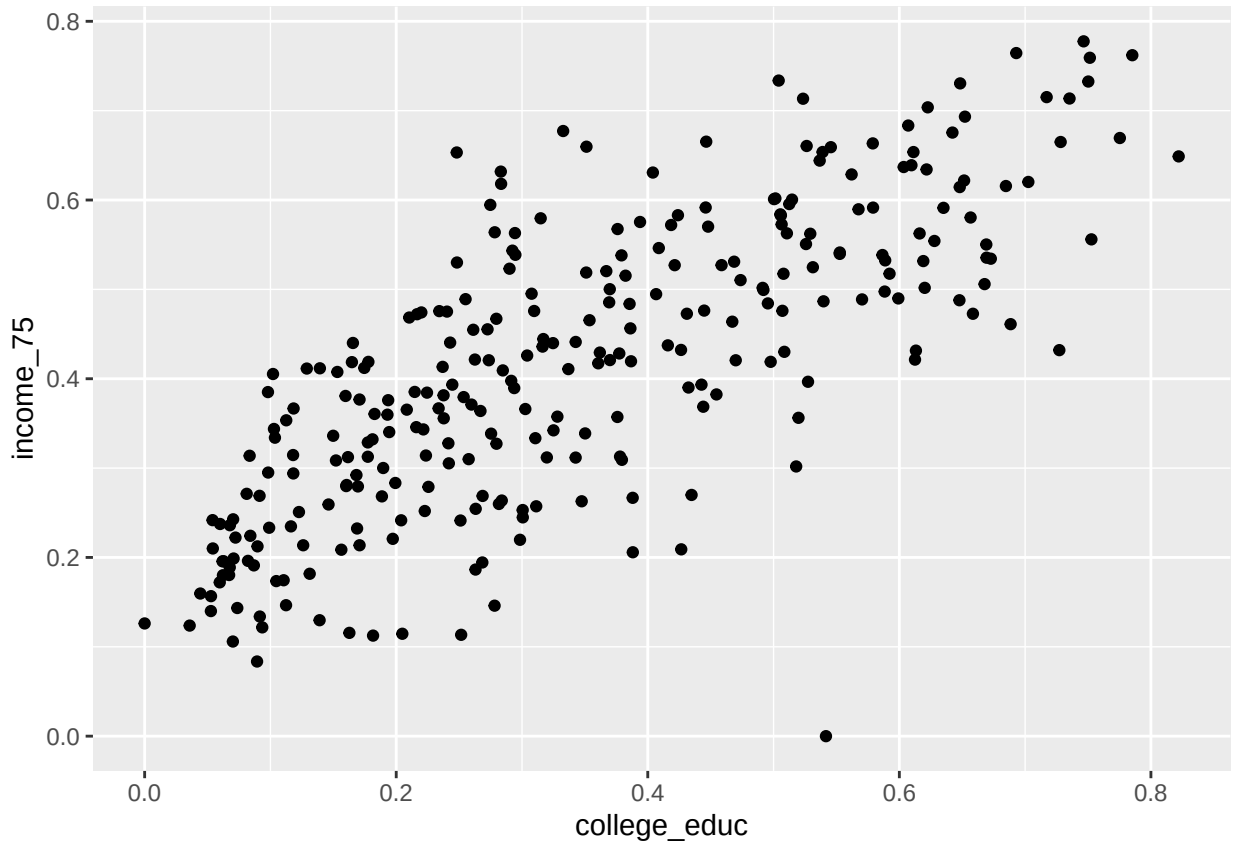
```
##  [7] "B19001. Household Income in the Past 12 Months (in 2015 Inflation-Adjusted Dollars): $30,000 to
##  [8] "B19001. Household Income in the Past 12 Months (in 2015 Inflation-Adjusted Dollars): $35,000 to
##  [9] "B19001. Household Income in the Past 12 Months (in 2015 Inflation-Adjusted Dollars): $40,000 to
## [10] "B19001. Household Income in the Past 12 Months (in 2015 Inflation-Adjusted Dollars): $45,000 to
## [11] "B19001. Household Income in the Past 12 Months (in 2015 Inflation-Adjusted Dollars): $50,000 to
## [12] "B19001. Household Income in the Past 12 Months (in 2015 Inflation-Adjusted Dollars): $60,000 to
## [13] "B19001. Household Income in the Past 12 Months (in 2015 Inflation-Adjusted Dollars): $75,000 to
## [14] "B19001. Household Income in the Past 12 Months (in 2015 Inflation-Adjusted Dollars): $100,000 t
## [15] "B19001. Household Income in the Past 12 Months (in 2015 Inflation-Adjusted Dollars): $125,000 t
## [16] "B19001. Household Income in the Past 12 Months (in 2015 Inflation-Adjusted Dollars): $150,000 t
## [17] "B19001. Household Income in the Past 12 Months (in 2015 Inflation-Adjusted Dollars): $200,000 o
```

```
#Proportion above 75k--
prop_above_75<-divide.acs(numerator=(county_income[,13]+
                            county_income[,14]+
                            county_income[,15]+
                            county_income[,16]+
                            county_income[,17]),
                          denominator=county_income[,1]
                          )

# Convert to tibble
county_df<-tibble(substr(geography(county_educ)[[1]],7,11),
                        as.numeric(estimate(prop_coll_above)),
                        as.numeric(estimate(prop_above_75))
)

# Give it easy to use names
names(county_df)<-c("zip","college_educ","income_75")
save(county_df,file="dav.RData")

head(county_df)
```

```
## # A tibble: 6 x 3
##   zip   college_educ income_75
##   <chr>        <dbl>     <dbl>
## 1 90001       0.0443     0.160
## 2 90002       0.0528     0.157
## 3 90003       0.0527     0.140
## 4 90004       0.347      0.263
## 5 90005       0.278      0.146
## 6 90006       0.163      0.116
```

```
gg<-ggplot(county_df,aes(x=college_educ,y=income_75))
gg<-gg+geom_point()
gg
```

```
## Warning: Removed 13 rows containing missing values (geom_point).
```



Quick Exercise Pull table B23001 "Sex by Age by Employment Status for the Population 16 and over" from ACS.

This resource is amazingly helpful. It means that with a list of zip codes you can get a huge amount of information about the area where the individual resides, including education, housing, income, medical care and other topics.