0. Setup - API Key, Load [R] Packages

Use the 'cloud' version of jupyter notebook with R

https://tmpnb.org

You Need a free Census API "Key" - Signup

http://api.census.gov/data/key_signup.html

Install [R] Packages that we will use latter

```
# install.packages(c('dplyr','httr','jsonlite','tidyr','ggplot2','foreign',
#                    'reshape','haven','packrat'),
#                    lib='/usr/local/lib/R/site-library',dependencies=TRUE)
```

Load [R] Packages that we will use latter

```
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(httr)
library(jsonlite)

library(tidyr)


library(ggplot2)

library(foreign)

# below aren't necessary, they're alternative 'Paths to Rome'
# library(reshape)
# library(haven)
# library(packrat)
```

```
# install.packages('packrat')
```

Mental Exercise: Reflect on how the Jupyter or Rstudio workflow can help transparency

Answer in the jupyter markdown block below

1. Obtain - Data Stream

Application Programming Interface - US Census API

In a nutshell, an API is a set of well-defined 'rules of engagement' to request items from a supplier.

https://en.wikipedia.org/wiki/Application_programming_interface

Most commonly, you glue together 3 items as a web URL

```
1. a standardized "base" web url
    (doesn't change for anyone or any request)

2. your specific request items
    (can change, up to the user)

3. your user-specific API Key
    (fixed for individual user, but different across individuals)
```

We will get our requested data from the US Census Bureau API

A one stop shop table of contents that shows the currently available Census datasets

http://api.census.gov/data.html

2015 Census Planning Database: Tract Level

We will use the US Census Bureau's API, to request the Census Planning Database: Tract Level

Variables:

http://api.census.gov/data/2015/pdb/tract/variables.html

API Syntax:

http://api.census.gov/data/2015/pdb/tract?get=FOO1&key=FOO2

NOTE: clicking the above URL will NOT work NOTE: the two "FOO1" and "FOO2" entries, which are placeholders, to be specified by YOU

Example - Glue: API base url + Request + ...

Michael Tzen Requests:

```
Scope - LA County (state:06+county:037)
Entity - Tract Level
Attributes - County, State, Total Population in 2010 Census, Tract Land Area, Females in Census 2010
```

Using the 'API rules of engagement' I shoehorn my requests into the WEB URL below:

http://api.census.gov/data/2015/pdb/tract?get=County_name,State_name,Tot_Population_CEN_2010, LAND_AREA,Females_CEN_2010&for=tract:*&in=state:06+county:037&key=YOUR_KEY_GOES_ HERE

NOTE: clicking the above URL will NOT work

NOTE: "YOUR_KEY_GOES_HERE" ("FOO2"), my user-specific API Key, is still NOT specified by me

[R] Example - Glue: API base url + Request + API Key

[R] 'dplyr' package https://cran.rstudio.com/web/packages/dplyr/vignettes/introduction.html

NOTE: Census switched to https so see examples

```
# library(dplyr)

# url_base = "http://api.census.gov/data/2015/pdb/tract?"

#
url_base = "http://api.census.gov/data/2015/pdb/tract?" %>%
  gsub(.,pattern="http",replacement="https")



request = "County_name,State_name,Tot_Population_CEN_2010,LAND_AREA,Females_CEN_2010&for=tract:*&in=sta

# put key inside quotes
# YOUR_KEY_GOES_HERE = ""

# pasted output has quotes
# paste0(url_base,'get=',request,'&key=',YOUR_KEY_GOES_HERE)

# paste output, THEN, noquotes
# paste0(url_base,'get=',request,'&key=',YOUR_KEY_GOES_HERE) %>% noquote()
```

NOTE: Census API returns your requested data in "JSON" format

https://en.wikipedia.org/wiki/JSON

JSON in a nutshell: a list of key-value pairs. The current standard format since it is BOTH machine readable and has human readable layout (looks like a shopping list)

On Your Own: assemble an API call for New York State

```
# url_base = ""

# request = ""

# ?paste0
```

Mental Exercise: Reflect on how the software steps make data obtainment a transparent process

Answer in the jupyter markdown block below

2. Input - Data into Software

Using [R] software, read and store the API returned JSON Data

```
# library(httr)
# library(jsonlite)

# url_fin = paste0(url_base,'get=',request,'&key=',YOUR_KEY_GOES_HERE)  # pasted output has quotes
# url_fin

url_fin_more = paste0(url_base,'get=',request,'&key=',YOUR_KEY_GOES_HERE)  # pasted output has quotes
url_fin = url_fin_more

# note, your local computer needs to be able to use https

# ?GET
req_url_fin = httr::GET(url_fin)

# ?content
# extract (json) content from a request

json_req = httr::content(req_url_fin, as = "text")

# agrees with 'point and click' return
# json_req %>% cat()

# human readable, "prettify" like a shopping list
# prettify(json_req,indent=1)
```

Null Values

[“Los Angeles County”,“California”,null,null,null,“06”,“037”,“137000”]

Convert JSON Data into Tabular Form - Characters and Numerics are Fundamental

Note: here, we manually copy+paste .json data returned from Census API

the cell above using json_req = httr::content() is the programatic [R] way of getting the API content

- caveat... we need network access for R and Jupyter (denied)

‘Error in curl::curl_fetch_memory(url, handle = handle): Couldn’t resolve host name’

So... let’s just get the content manually. what we are doing here below.

workaround: just copy and paste returned json (in browser tab) after ‘clicking’ the r assembled API call

- ‘patchwork’ but students can see that .json is nothing more than a format like .csv
- only used [R] to assemble the URL for API call
- couldn’t use R to ‘navigate’ the URL via `curl::curl_fetch_memory()`
- BEAUTY of jupyter being in-browser, the assembled URL can be clicked (this is completley from jupyter / markdown)

```
# json_req = '[["County_name","State_name","Tot_Population_CEN_2010","LAND_AREA","Females_CEN_2010","st
# ["Alameda County","California","2937","2.657","1476","06","001","400100"]]'
```

```
# ?fromJSON
# https://cran.r-project.org/web/packages/jsonlite/vignettes/json-apis.html

# NEEDS quotes around final url
# url_fin = url_fin_more

# print(url_fin)

data_pdb_raw = json_req %>%
# url_fin %>%
# noquote() %>%
fromJSON()

dim(data_pdb_raw)
```

```
## [1] 2348    8
```

```
# json 'null' is correctly encoded as r matrix 'NA'

summary(data_pdb_raw)
```

```
##                   V1                 V2             V3             V4
##  County_name       :   1   California:2347   0      :  12   0.126  :  17
##  Los Angeles County:2347   State_name:   1   2645   :   4   0.125  :  10
##                                              3110   :   4   0.188  :  10
##                                              3942   :   4   0.251  :  10
##                                              4074   :   4   0.124  :   9
##                                              (Other):2319   (Other):2291
##                                              NA's   :   1   NA's   :   1
##        V5            V6             V7              V8
##  0      :  13   06   :2347   037   :2347   101110 :   1
##  2034   :   7   state:   1   county:   1   101122 :   1
##  2273   :   6                              101210 :   1
##  2284   :   6                              101220 :   1
##  1510   :   5                              101300 :   1
##  (Other):2310                              101400 :   1
##  NA's   :   1                              (Other):2342
```

```
# r matrix shows 1 NA, which agrees with single null in json source

sum(is.na(data_pdb_raw))
```

```
## [1] 3
```

```
which(is.na(data_pdb_raw), arr.ind = TRUE)
```

```
##      row col
## [1,] 304   3
## [2,] 304   4
## [3,] 304   5
```

```
head(data_pdb_raw)
```

```
##      [,1]                 [,2]         [,3]
## [1,] "County_name"        "State_name" "Tot_Population_CEN_2010"
## [2,] "Los Angeles County" "California" "4731"
## [3,] "Los Angeles County" "California" "3664"
```

```
## [4,] "Los Angeles County" "California" "5990"
## [5,] "Los Angeles County" "California" "3363"
## [6,] "Los Angeles County" "California" "4199"
##      [,4]        [,5]               [,6]    [,7]     [,8]
## [1,] "LAND_AREA" "Females_CEN_2010" "state" "county" "tract"
## [2,] "0.442"     "2352"             "06"    "037"    "101110"
## [3,] "1.021"     "1869"             "06"    "037"    "101122"
## [4,] "0.251"     "2953"             "06"    "037"    "101210"
## [5,] "0.27"      "1702"             "06"    "037"    "101220"
## [6,] "0.996"     "2183"             "06"    "037"    "101300"
```

```
data_pdb_raw[unique(which(is.na(data_pdb_raw), arr.ind = TRUE)[,1]),]
```

```
## [1] "Los Angeles County" "California"         NA
## [4] NA                   NA                   "06"
## [7] "037"                "137000"
```

```
dim(data_pdb_raw)
```

```
## [1] 2348    8
```

```
# NA in row 1475
data_pdb_raw[1475,]
```

```
## [1] "Los Angeles County" "California"         "7190"
## [4] "3.775"              "3643"               "06"
## [7] "037"                "501600"
```

```
# data_pdb_raw = data_pdb_raw[-1475,]


data_pdb_raw[unique(which(is.na(data_pdb_raw), arr.ind = TRUE)[,1]),]
```

```
## [1] "Los Angeles County" "California"         NA
## [4] NA                   NA                   "06"
## [7] "037"                "137000"
```

```
data_pdb_raw = data_pdb_raw[-unique(which(is.na(data_pdb_raw), arr.ind = TRUE)[,1]),]


sum(is.na(data_pdb_raw))
```

```
## [1] 0
```

```
dim(data_pdb_raw)
```

```
## [1] 2347    8
```

```
# <8b> is a jupyter idiom for '...' shortening output
head(data_pdb_raw)
```

```
##      [,1]                [,2]         [,3]
## [1,] "County_name"       "State_name" "Tot_Population_CEN_2010"
## [2,] "Los Angeles County" "California" "4731"
## [3,] "Los Angeles County" "California" "3664"
## [4,] "Los Angeles County" "California" "5990"
## [5,] "Los Angeles County" "California" "3363"
## [6,] "Los Angeles County" "California" "4199"
##      [,4]        [,5]               [,6]    [,7]     [,8]
```

```
## [1,] "LAND_AREA" "Females_CEN_2010" "state" "county" "tract"
## [2,] "0.442"     "2352"             "06"    "037"    "101110"
## [3,] "1.021"     "1869"             "06"    "037"    "101122"
## [4,] "0.251"     "2953"             "06"    "037"    "101210"
## [5,] "0.27"      "1702"             "06"    "037"    "101220"
## [6,] "0.996"     "2183"             "06"    "037"    "101300"
```

Hurdle: Immediate Matrix without Header (Variable Names)

BUT, It IS there in row 1. EVERYTHING is currently read as 'character' strings of text stored in a tabular MATRIX

I view character strings as a huge benefit. Keeping everything as characters (at least initially) allows for consistent behavior when you merge, filter, apply logic to your data. It Prevents you from shooting yourself in the foot (like adding two strings, say "State Code" + "County Code"). This is Tomatoe = Potatoe. Later on, we explicitly convert intended numeric numbers into numeric data (Tomatoe = Tomatoe).

```
class(data_pdb_raw)
```

```
## [1] "matrix"
```

```
dim(data_pdb_raw)
```

```
## [1] 2347    8
```

```
str(data_pdb_raw)
```

```
##  chr [1:2347, 1:8] "County_name" "Los Angeles County" ...
```

```
colnames(data_pdb_raw)
```

```
## NULL
```

```
rownames(data_pdb_raw)
```

```
## NULL
```

Hurdle: Humans recognize that first row represents variable names, Need Computer to recognize it too (as column names)

```
# first row of matrix - humans understand as the intended variable names
data_pdb_raw[1,]
```

```
## [1] "County_name"             "State_name"
## [3] "Tot_Population_CEN_2010" "LAND_AREA"
## [5] "Females_CEN_2010"        "state"
## [7] "county"                  "tract"
```

```
# column names (header) of matrix - computer understands as empty
colnames(data_pdb_raw)
```

```
## NULL
```

```
# make the assignment: column names of matrix = first row of matrix
colnames(data_pdb_raw) = data_pdb_raw[1,]

# assignment reflected but first row still present (redundant)
data_pdb_raw %>% head()
```

```
##      County_name          State_name   Tot_Population_CEN_2010
## [1,] "County_name"        "State_name" "Tot_Population_CEN_2010"
## [2,] "Los Angeles County" "California" "4731"
## [3,] "Los Angeles County" "California" "3664"
## [4,] "Los Angeles County" "California" "5990"
## [5,] "Los Angeles County" "California" "3363"
## [6,] "Los Angeles County" "California" "4199"
##      LAND_AREA   Females_CEN_2010   state   county   tract
## [1,] "LAND_AREA" "Females_CEN_2010" "state" "county" "tract"
## [2,] "0.442"     "2352"             "06"    "037"    "101110"
## [3,] "1.021"     "1869"             "06"    "037"    "101122"
## [4,] "0.251"     "2953"             "06"    "037"    "101210"
## [5,] "0.27"      "1702"             "06"    "037"    "101220"
## [6,] "0.996"     "2183"             "06"    "037"    "101300"
```

```
# data_pdb_mat: a new object distinguishing it from _raw source
# throw away first row

data_pdb_mat = data_pdb_raw[-1,]

head(data_pdb_mat)
```

```
##      County_name          State_name   Tot_Population_CEN_2010 LAND_AREA
## [1,] "Los Angeles County" "California" "4731"                  "0.442"
## [2,] "Los Angeles County" "California" "3664"                  "1.021"
## [3,] "Los Angeles County" "California" "5990"                  "0.251"
## [4,] "Los Angeles County" "California" "3363"                  "0.27"
## [5,] "Los Angeles County" "California" "4199"                  "0.996"
## [6,] "Los Angeles County" "California" "3903"                  "2.436"
##      Females_CEN_2010 state county tract
## [1,] "2352"           "06"  "037"  "101110"
## [2,] "1869"           "06"  "037"  "101122"
## [3,] "2953"           "06"  "037"  "101210"
## [4,] "1702"           "06"  "037"  "101220"
## [5,] "2183"           "06"  "037"  "101300"
## [6,] "1948"           "06"  "037"  "101400"
```

Hurdle: Matrix to Dataframe, but [R] default converts characters, we want to keep everything as Fundamental 'Character' string

options(stringsAsFactors = FALSE)

```
# set options
```

```
str(data_pdb_mat)
```

```
##  chr [1:2346, 1:8] "Los Angeles County" "Los Angeles County" ...
##  - attr(*, "dimnames")=List of 2
##   ..$ : NULL
##   ..$ : chr [1:8] "County_name" "State_name" "Tot_Population_CEN_2010" "LAND_AREA" ...
```

```
# NOTE: default of [R]: automaticaly converting 'character' to 'factor'
# this is an artifact leftover from oldtime convenience

data_pdb_mat %>% as.data.frame() %>% str()
```

```
## 'data.frame':    2346 obs. of  8 variables:
##  $ County_name            : Factor w/ 1 level "Los Angeles County": 1 1 1 1 1 1 1 1 1 1 ...
##  $ State_name             : Factor w/ 1 level "California": 1 1 1 1 1 1 1 1 1 1 ...
##  $ Tot_Population_CEN_2010: Factor w/ 1888 levels "0","1","1029",..: 1206 709 1626 557 969 819 56 704
##  $ LAND_AREA              : Factor w/ 1078 levels "0","0.027","0.031",..: 374 755 188 207 740 964 389
##  $ Females_CEN_2010       : Factor w/ 1572 levels "0","1","101",..: 863 527 1196 404 747 577 1547 540
##  $ state                  : Factor w/ 1 level "06": 1 1 1 1 1 1 1 1 1 1 ...
##  $ county                 : Factor w/ 1 level "037": 1 1 1 1 1 1 1 1 1 1 ...
##  $ tract                  : Factor w/ 2346 levels "101110","101122",..: 1 2 3 4 5 6 7 8 9 10 ...
```

```
# turn that default option OFF
options(stringsAsFactors = FALSE)

data_pdb_mat %>% as.data.frame() %>% str()
```

```
## 'data.frame':    2346 obs. of  8 variables:
##  $ County_name            : chr  "Los Angeles County" "Los Angeles County" "Los Angeles County" "Los
##  $ State_name             : chr  "California" "California" "California" "California" ...
##  $ Tot_Population_CEN_2010: chr  "4731" "3664" "5990" "3363" ...
##  $ LAND_AREA              : chr  "0.442" "1.021" "0.251" "0.27" ...
##  $ Females_CEN_2010       : chr  "2352" "1869" "2953" "1702" ...
##  $ state                  : chr  "06" "06" "06" "06" ...
##  $ county                 : chr  "037" "037" "037" "037" ...
##  $ tract                  : chr  "101110" "101122" "101210" "101220" ...
```

```
# store as data frame
data_pdb_df = data_pdb_mat %>% as.data.frame()
```

Hurdle: Humans recognize that 'Tot_Population_CEN_2010' represents a count, Need Computer to recognize it too (as 'Numeric' number)

```
data_pdb_df %>% str()
```

```
## 'data.frame':    2346 obs. of  8 variables:
##  $ County_name            : chr  "Los Angeles County" "Los Angeles County" "Los Angeles County" "Los
##  $ State_name             : chr  "California" "California" "California" "California" ...
##  $ Tot_Population_CEN_2010: chr  "4731" "3664" "5990" "3363" ...
##  $ LAND_AREA              : chr  "0.442" "1.021" "0.251" "0.27" ...
##  $ Females_CEN_2010       : chr  "2352" "1869" "2953" "1702" ...
##  $ state                  : chr  "06" "06" "06" "06" ...
##  $ county                 : chr  "037" "037" "037" "037" ...
##  $ tract                  : chr  "101110" "101122" "101210" "101220" ...
```

```
# want numeric numbers for
# Tot_Population_CEN_2010
# LAND_AREA
# Females_CEN_2010

# more data
data_pdb_df = data_pdb_df %>%
mutate(Tot_Population_CEN_2010=as.numeric(Tot_Population_CEN_2010),
       LAND_AREA=as.numeric(LAND_AREA)
      )
```

```
data_pdb_df %>%
str()
```

```
## 'data.frame':    2346 obs. of  8 variables:
##  $ County_name            : chr  "Los Angeles County" "Los Angeles County" "Los Angeles County" "Los
##  $ State_name             : chr  "California" "California" "California" "California" ...
##  $ Tot_Population_CEN_2010: num  4731 3664 5990 3363 4199 ...
##  $ LAND_AREA              : num  0.442 1.021 0.251 0.27 0.996 ...
##  $ Females_CEN_2010       : chr  "2352" "1869" "2953" "1702" ...
##  $ state                  : chr  "06" "06" "06" "06" ...
##  $ county                 : chr  "037" "037" "037" "037" ...
##  $ tract                  : chr  "101110" "101122" "101210" "101220" ...
```

```
data_pdb_df %>%
summary()
```

```
##  County_name         State_name        Tot_Population_CEN_2010
##  Length:2346        Length:2346        Min.   :    0
##  Class :character   Class :character   1st Qu.: 3203
##  Mode  :character   Mode  :character   Median : 4098
##                                        Mean   : 4185
##                                        3rd Qu.: 5162
##                                        Max.   :12544
##    LAND_AREA       Females_CEN_2010      state
##  Min.   :  0.000   Length:2346        Length:2346
##  1st Qu.:  0.229   Class :character   Class :character
##  Median :  0.392   Mode  :character   Mode  :character
##  Mean   :  1.730
##  3rd Qu.:  0.702
##  Max.   :397.252
##     county             tract
##  Length:2346        Length:2346
##  Class :character   Class :character
##  Mode  :character   Mode  :character
##
##
##
```

On Your Own: convert 'Females_CEN_2010' to a numeric number

```
# data_pdb_df = data_pdb_df %>%
# mutate(
#       )

# check structure
```

Clear Hurdle? Basic Assertions - Check some Snapple Facts

Can Population be Negative?

```
data_pdb_df %>%
filter(Tot_Population_CEN_2010 < 0)
```

```
## [1] County_name             State_name              Tot_Population_CEN_2010
## [4] LAND_AREA               Females_CEN_2010        state
## [7] county                  tract
## <0 rows> (or 0-length row.names)
```

```
data_pdb_df %>%
filter(Tot_Population_CEN_2010 < 0) %>%
nrow()
```

```
## [1] 0
```

Basic Assertion: How ManyTracts have Positive Population?

```
data_pdb_df %>%
filter(Tot_Population_CEN_2010 > 0) %>% nrow()
```

```
## [1] 2334
```

Basic Assertion: How Many Tracts have Positive OR Zero Population? (Should be all Tracts in LA County)

```
# apply filter where Total Population Count >=0

data_pdb_df %>%
filter((Tot_Population_CEN_2010 > 0) || (Tot_Population_CEN_2010 == 0)) %>%
nrow()
```

```
## [1] 2346
```

```
# number of rows in census returned dataset (raw)
# no filter
data_pdb_df %>% nrow()
```

```
## [1] 2346
```

On Your Own: How Many Tracts have Zero Population?

```
# data_pdb_df %>%
# filter() %>%
# nrow()
```

Basic Assertion: Does our sum of LA County population agree with Independent Source?

```
# is.na(data_pdb_df)
# summary(data_pdb_df)

data_pdb_df %>%
group_by(County_name) %>%
summarise(Tot_Pop_County = sum(Tot_Population_CEN_2010))
```

```
## # A tibble: 1 x 2
##          County_name Tot_Pop_County
##                <chr>          <dbl>
```

```
## 1 Los Angeles County        9818605
```

```
summary(data_pdb_df)
```

```
##  County_name         State_name        Tot_Population_CEN_2010
##  Length:2346        Length:2346        Min.   :    0
##  Class :character   Class :character   1st Qu.: 3203
##  Mode  :character   Mode  :character   Median : 4098
##                                        Mean   : 4185
##                                        3rd Qu.: 5162
##                                        Max.   :12544
##    LAND_AREA       Females_CEN_2010      state
##  Min.   :  0.000   Length:2346        Length:2346
##  1st Qu.:  0.229   Class :character   Class :character
##  Median :  0.392   Mode  :character   Mode  :character
##  Mean   :  1.730
##  3rd Qu.:  0.702
##  Max.   :397.252
##     county             tract
##  Length:2346        Length:2346
##  Class :character   Class :character
##  Mode  :character   Mode  :character
##
##
##
```

https://www.google.com/search?q=how+many+people+in+la+county&ie=utf-8&oe=utf-8

Mental Exercise: Reflect on how the software steps make reading in data a transparent process

Answer in the jupyter markdown block below

3. Wrangle - Data into Analysis Ready Tabular Form

https://www.rstudio.com/wp-content/uploads/2015/02/data-wrangling-cheatsheet.pdf

Start with Tidy data, then Wrangle to Coordinate

The API returned census data is already in "Tidy" format (each row is an entity, each column is a measure)

Example: Tidy

Name	Year	Age	Sex
Arnold Schwarzenegger	2030	34	M
Arnold Schwarzenegger	2040	44	M
Sofia Vergara	2030	30	F
Sofia Vergara	2040	40	F

Example: Coordinate

Name	Year	Variable Name	Variable Value
Arnold Schwarzenegger	2030	Age	34
Arnold Schwarzenegger	2030	Sex	M
Arnold Schwarzenegger	2040	Age	44
Arnold Schwarzenegger	2040	Sex	M
Sofia Vergara	2030	Age	30
Sofia Vergara	2030	Sex	F
Sofia Vergara	2040	Age	40
Sofia Vergara	2040	Sex	F

Create Primary Key / Entity ID by gluing raw entity variables via 'tidyr' and unite()

```
# library(tidyr)

data_pdb_df %>%
head()
```

```
##          County_name State_name Tot_Population_CEN_2010 LAND_AREA
## 1 Los Angeles County California                     4731     0.442
## 2 Los Angeles County California                     3664     1.021
## 3 Los Angeles County California                     5990     0.251
## 4 Los Angeles County California                     3363     0.270
## 5 Los Angeles County California                     4199     0.996
## 6 Los Angeles County California                     3903     2.436
##   Females_CEN_2010 state county  tract
## 1             2352    06    037 101110
## 2             1869    06    037 101122
## 3             2953    06    037 101210
## 4             1702    06    037 101220
## 5             2183    06    037 101300
## 6             1948    06    037 101400
```

```
# [State,County,Tract] are the three variables, when viewed together, define the entity id
# County_name, State_name, Tot_Population_CEN_2010, are the three variables that are measurements of th

# ?tidyr::unite() to glue together variables into single entity id

# primary key := create a single entity id, by uniting the three entity variables
data_pdb_df_pk = data_pdb_df %>%
unite("state_county_tract",state,county,tract)

data_pdb_df_pk %>% head()
```

```
##          County_name State_name Tot_Population_CEN_2010 LAND_AREA
## 1 Los Angeles County California                     4731     0.442
## 2 Los Angeles County California                     3664     1.021
## 3 Los Angeles County California                     5990     0.251
## 4 Los Angeles County California                     3363     0.270
## 5 Los Angeles County California                     4199     0.996
## 6 Los Angeles County California                     3903     2.436
##   Females_CEN_2010 state_county_tract
## 1             2352      06_037_101110
## 2             1869      06_037_101122
```

```
## 3             2953      06_037_101210
## 4             1702      06_037_101220
## 5             2183      06_037_101300
## 6             1948      06_037_101400
```

```
# note: unite() default behavior drops original entity variables, and glues in underscore "_"
```

On Your Own: separate() the 'state_county_tract' variable back into 3 separate variables for state, county, tract

```
# ?tidyr::separate
```

```
# data_pdb_df_pk %>%
```

Many ways to Rome: Create Primary Key / Entity ID

```
# Alternative way of creating entity id by gluing entity variables
# using mutate(paste0())

data_pdb_df %>%
mutate(state_county_tract_v2 = paste0(state,county,tract)) %>%
head()
```

```
##          County_name State_name Tot_Population_CEN_2010 LAND_AREA
## 1 Los Angeles County California                    4731     0.442
## 2 Los Angeles County California                    3664     1.021
## 3 Los Angeles County California                    5990     0.251
## 4 Los Angeles County California                    3363     0.270
## 5 Los Angeles County California                    4199     0.996
## 6 Los Angeles County California                    3903     2.436
##   Females_CEN_2010 state county  tract state_county_tract_v2
## 1             2352    06    037 101110           06037101110
## 2             1869    06    037 101122           06037101122
## 3             2953    06    037 101210           06037101210
## 4             1702    06    037 101220           06037101220
## 5             2183    06    037 101300           06037101300
## 6             1948    06    037 101400           06037101400
```

Tidy to Coordinate via 'tidyr' and gather()

```
# Except for our entity id "state_county_tract"
# gather all columns (several measures) into two seperate columns [variable name,variable value]

data_pdb_df_pk %>%
gather(var_name,var_val,-state_county_tract) %>%
head()
```

```
##   state_county_tract    var_name            var_val
## 1      06_037_101110 County_name Los Angeles County
## 2      06_037_101122 County_name Los Angeles County
## 3      06_037_101210 County_name Los Angeles County
## 4      06_037_101220 County_name Los Angeles County
## 5      06_037_101300 County_name Los Angeles County
## 6      06_037_101400 County_name Los Angeles County
```

```
data_pdb_df_pk %>%
gather(var_name,var_val,-state_county_tract) %>%
tail()
```

```
##       state_county_tract         var_name var_val
## 11725      06_037_980030 Females_CEN_2010       0
## 11726      06_037_980031 Females_CEN_2010      94
## 11727      06_037_980033 Females_CEN_2010      12
## 11728      06_037_990100 Females_CEN_2010       0
## 11729      06_037_990200 Females_CEN_2010       0
## 11730      06_037_990300 Females_CEN_2010       0
```

```
data_pdb_df_coord = data_pdb_df_pk %>%
gather(var_name,var_val,-state_county_tract)
```

Many ways to Rome - Tidy to Coordinate via 'reshape' and melt()

NOTE: 'reshape' more powerful, sometimes overkill. Allows more control of "melting" and "casting" into various tabular shapes

```
# library(reshape)

# Many ways to Rome: alternative method
# ?reshape2::melt() to coordinate format

# data_pdb_df_pk %>%
# reshape::melt("state_county_tract") %>%
# head()

# data_pdb_df_pk %>%
# reshape::melt("state_county_tract") %>%
# tail()
```

Mental Exercise: Reflect on how the wrangled data structures can help transparency

Answer in the jupyter markdown block below

4. Analyze - Plot, Summary, Model

Plot a Graphic

```
data_pdb_df_pk %>% head()
```

```
##          County_name State_name Tot_Population_CEN_2010 LAND_AREA
## 1 Los Angeles County California                     4731     0.442
## 2 Los Angeles County California                     3664     1.021
## 3 Los Angeles County California                     5990     0.251
## 4 Los Angeles County California                     3363     0.270
## 5 Los Angeles County California                     4199     0.996
## 6 Los Angeles County California                     3903     2.436
```

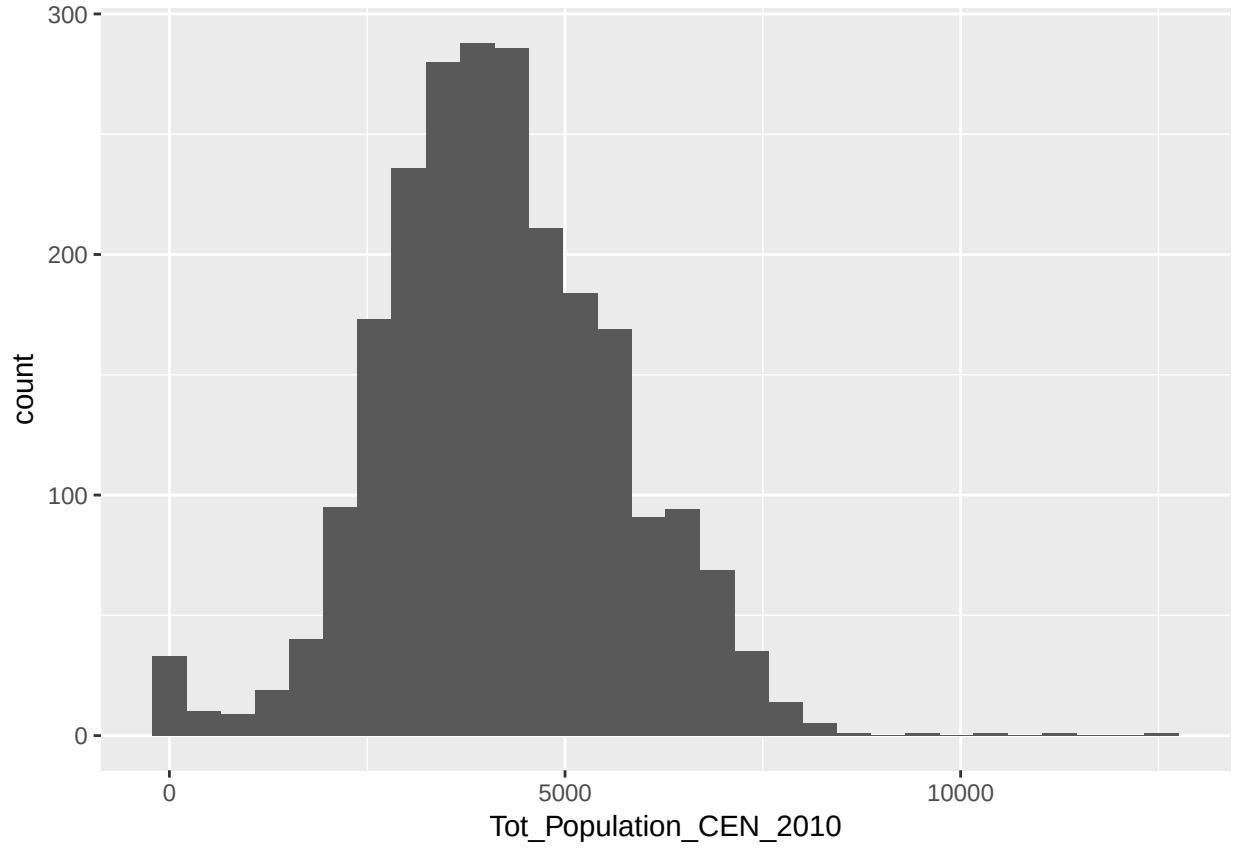
```
##   Females_CEN_2010 state_county_tract
## 1             2352      06_037_101110
## 2             1869      06_037_101122
## 3             2953      06_037_101210
## 4             1702      06_037_101220
## 5             2183      06_037_101300
## 6             1948      06_037_101400
```

```
# library(ggplot2)

# convert
## Relevel the cars by mpg
## this allows the plot to sort from most to least

# histogram of population count (each entity is a tract)
qplot(data=data_pdb_df_pk,Tot_Population_CEN_2010,geom="histogram")
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```




```
# p = ggplot(data_pdb_df_pk, aes(x=reorder(state_county_tract,-Tot_Population_CEN_2010), y=Tot_Populati
#     geom_point(stat="identity")
```

On Your Own: Request Tract Size Variable, Make scatterplot of Population by Tract Size

http://api.census.gov/data/2015/pdb/tract/variables/LAND_AREA.json

```
# ggplot(data_pdb_df_pk, aes(x=(LAND_AREA),y=(Tot_Population_CEN_2010))) +
# geom_point()
#
#
# ggplot(data_pdb_df_pk, aes(x=log(LAND_AREA),y=log(Tot_Population_CEN_2010))) +
# geom_point()
```

Summarize a Table

Assume A Table is Our Desired Analysis

County Level Count of Males and Females, sorted in decreasing order by Total Population Count

```
str(data_pdb_df_pk)

data_pdb_df_pk %>%
mutate(Num_Males_2010 = Tot_Population_CEN_2010 - Females_CEN_2010) %>%
group_by(State_name,County_name) %>%
summarise(Num_M_2010_Cou=sum(Num_Males_2010),
          Num_F_2010_Cou=sum(Females_CEN_2010),
           Num_All_2010_Cou=sum(Tot_Population_CEN_2010)
          ) %>%
arrange(desc(Num_All_2010_Cou))

tab_sex_county = data_pdb_df_pk %>%
mutate(Num_Males_2010 = Tot_Population_CEN_2010 - Females_CEN_2010) %>%
group_by(State_name,County_name) %>%
summarise(Num_M_2010_Cou=sum(Num_Males_2010),
          Num_F_2010_Cou=sum(Females_CEN_2010),
           Num_All_2010_Cou=sum(Tot_Population_CEN_2010)
          ) %>%
arrange(desc(Num_All_2010_Cou))
```

Assertion Check: How Many Counties in California?

https://www.google.com/search?q=how+many+counties+in+california&ie=utf-8&oe=utf-8

Fit a Regression Model

Regress Population Count on Land Area

```
names(data_pdb_df_pk)
```

```
## [1] "County_name"             "State_name"
## [3] "Tot_Population_CEN_2010" "LAND_AREA"
## [5] "Females_CEN_2010"        "state_county_tract"
```

```
data_pdb_df_pk %>%
head()
```

```
##          County_name State_name Tot_Population_CEN_2010 LAND_AREA
## 1 Los Angeles County California                    4731     0.442
## 2 Los Angeles County California                    3664     1.021
```

```
## 3 Los Angeles County California                    5990     0.251
## 4 Los Angeles County California                    3363     0.270
## 5 Los Angeles County California                    4199     0.996
## 6 Los Angeles County California                    3903     2.436
##   Females_CEN_2010 state_county_tract
## 1             2352      06_037_101110
## 2             1869      06_037_101122
## 3             2953      06_037_101210
## 4             1702      06_037_101220
## 5             2183      06_037_101300
## 6             1948      06_037_101400
```

```
# intercept allowed to be estimated when land area is 0
lm(Tot_Population_CEN_2010 ~ LAND_AREA,data=data_pdb_df_pk) %>%
summary()
```

```
##
## Call:
## lm(formula = Tot_Population_CEN_2010 ~ LAND_AREA, data = data_pdb_df_pk)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -4204.4  -995.8  -101.8   972.2  8395.5
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 4204.400     31.065 135.343  < 2e-16 ***
## LAND_AREA    -11.069      2.349  -4.712  2.6e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1492 on 2344 degrees of freedom
## Multiple R-squared:  0.009382,   Adjusted R-squared:  0.00896
## F-statistic:  22.2 on 1 and 2344 DF,  p-value: 2.6e-06
```

```
# force intercept to be 0
# 0 population when 0 land area
lm(Tot_Population_CEN_2010 ~ -1 + LAND_AREA, data=data_pdb_df_pk) %>%
summary()
```

```
##
## Call:
## lm(formula = Tot_Population_CEN_2010 ~ -1 + LAND_AREA, data = data_pdb_df_pk)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -11239   3176   4075   5116  12390
##
## Coefficients:
##           Estimate Std. Error t value Pr(>|t|)
## LAND_AREA   30.522      6.913   4.415 1.06e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4428 on 2345 degrees of freedom
## Multiple R-squared:  0.008243,   Adjusted R-squared:  0.00782
```

```
## F-statistic: 19.49 on 1 and 2345 DF,  p-value: 1.057e-05
```

```
# log-outcome and log-predictor, estimate the intercept
lm(log(Tot_Population_CEN_2010 + 0.01) ~ 1 + log(LAND_AREA + 0.01), data=data_pdb_df_pk) %>%
summary()
```

```
##
## Call:
## lm(formula = log(Tot_Population_CEN_2010 + 0.01) ~ 1 + log(LAND_AREA +
##     0.01), data = data_pdb_df_pk)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -12.9279  -0.1239   0.1322   0.3727   1.3463
##
## Coefficients:
##                         Estimate Std. Error t value Pr(>|t|)
## (Intercept)              8.15111    0.02829 288.133   <2e-16 ***
## log(LAND_AREA + 0.01) -0.03727    0.02150  -1.733   0.0832 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.1 on 2344 degrees of freedom
## Multiple R-squared:  0.00128,    Adjusted R-squared:  0.0008541
## F-statistic: 3.005 on 1 and 2344 DF,  p-value: 0.08316
```

Mental Exercise: Reflect on how you got to this "final" analysis stage. Compare the amount of code in 'Analyze Step 4' with Steps 1-3.

Answer in the jupyter markdown block below

0b. ORGANIZE!

Organization is a fundamental part of Transparency, hence step 0. Further, Organization is crucial during handoff migrations.

Transfer your wrangled data to a new environment

Output: 'data_pdb_df_pk'

In .csv format (Universal), Output Table to Computer File

Note: your_output_dir = '/home/foo' needs to represent your output directory

```
getwd()
```

```
# your_output_dir = 'Z:\\projects\\workshop_ccpr_stat\\workshop_data_workflow\\data_proc\\'

your_output_dir = '/home/jovyan/work/test/'

# ?write.csv
```

```
# write.csv(tab_sex_county,paste0(your_output_dir,'tab_sex_county.csv'))

# drop row number

write.csv(data_pdb_df_pk,paste0(your_output_dir,'data_pdb_df_pk.csv'),row.names=FALSE)
```

```
# list.files('/home/jovyan/work/test/')
list.files(your_output_dir)
```

In .dta format (Stata), Output Table to Computer File

```
# library(haven)

# ?write_dta
# write_dta(data_pdb_df_pk, paste0(your_output_dir,'data_pdb_df_pk_haven.dta'))
# ?read_dta

library(foreign)
# ?read.dta

write.dta(data_pdb_df_pk,paste0(your_output_dir,'data_pdb_df_pk_foreign.dta'))
```

```
list.files('/home/jovyan/work/test/')
```

Transfer your software versions. What versions are you using!?

Archive your software versions!

`?sessionInfo()`

type in the function yourself in the Code box below

```
sessionInfo()
```

```
## R version 3.4.1 (2017-06-30)
## Platform: x86_64-pc-linux-gnu (64-bit)
## Running under: Debian GNU/Linux 9 (stretch)
##
## Matrix products: default
## BLAS/LAPACK: /usr/lib/libopenblasp-r0.2.19.so
##
## locale:
##  [1] LC_CTYPE=en_US.UTF-8       LC_NUMERIC=C
##  [3] LC_TIME=en_US.UTF-8        LC_COLLATE=en_US.UTF-8
##  [5] LC_MONETARY=en_US.UTF-8    LC_MESSAGES=C
##  [7] LC_PAPER=en_US.UTF-8       LC_NAME=C
##  [9] LC_ADDRESS=C               LC_TELEPHONE=C
## [11] LC_MEASUREMENT=en_US.UTF-8 LC_IDENTIFICATION=C
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods   base
##
## other attached packages:
```

```
## [1] bindrcpp_0.2   foreign_0.8-69 ggplot2_2.2.1  tidyr_0.7.1
## [5] jsonlite_1.5   httr_1.3.1     dplyr_0.7.4
##
## loaded via a namespace (and not attached):
##  [1] Rcpp_0.12.13     knitr_1.17       bindr_0.1        magrittr_1.5
##  [5] tidyselect_0.2.0 munsell_0.4.3    colorspace_1.3-2 R6_2.2.2
##  [9] rlang_0.1.2      plyr_1.8.4       stringr_1.2.0    tools_3.4.1
## [13] grid_3.4.1       gtable_0.2.0     htmltools_0.3.6  lazyeval_0.2.0
## [17] yaml_2.1.14      assertthat_0.2.0 rprojroot_1.2    digest_0.6.12
## [21] tibble_1.3.4     purrr_0.2.3      curl_3.0         glue_1.1.1
## [25] evaluate_0.10.1  rmarkdown_1.6    labeling_0.3     stringi_1.1.5
## [29] compiler_3.4.1   scales_0.5.0     backports_1.1.1  pkgconfig_2.0.1
```

Software Versions... big whoop... So What? How do I make use of this info?

```
# library(devtools)
# devtools::install_version("tidyr", version = "0.3.1", repos = "http://cran.us.r-project.org")
```

Bundle the packages, export, and then unbundle in another environment

Even easier than manually installing specific version 1-by-1

https://rstudio.github.io/packrat/

```
library(packrat)

# ?packrat::bundle
packrat::bundle()

# ?packrat::unbundle
bundle_path = "/foo_path/foo.tar.gz"
packrat::unbundle(bundle=bundle_path)
```

Transfer your Jupyter File (from tmpnb.org) to your personal computer

Download as a .ipynb file or an .r

Appendix

Resources

- Jupyter http://jupyter.org/
- Rstudio https://www.rstudio.com/
- The Elements of Data Analytic Style https://leanpub.com/datastyle
- Tidy Data http://www.jstatsoft.org/v59/i10
- Open Science Framework https://osf.io/

[R] to Stata Interface

https://github.com/EconometricsBySimulation/RStata/wiki/Dictionary:-Stata-to-R

Need a STATA copy?

http://www.ccpr.ucla.edu/CCPRWebsite/services/computing

Need a STATA hint?

http://www.ats.ucla.edu/stat/stata/modules/

Fit a regression model in Stata

read in data

use "Z:\projects\workshop_ccpr_stat\workshop_data_workflow\data_proc\data_pdb_df_pk_foreign.dta", clear

estimate intercept

reg Tot_Population_CEN_2010 LAND_AREA

force intercept to be 0

reg Tot_Population_CEN_2010 LAND_AREA, nocon

log-outcome and log-predictor, estimate the intercept

generate log_Tot_Pop = log(Tot_Population_CEN_2010+0.01)
generate log_Land_Area = log(LAND_AREA+0.01)

reg log_Tot_Pop log_Land_Area