ds421 final project

xrw

5/3/2018

#Contents

This is a final project PDF document for DS421 stitched together from other experiments in this rpo.

Some major goals were: - Get satellite data/imagery for village and county names. - Poke around household income data - Poke around land use change for a few Taobao villages.

Section A

First we'll take a look at household income data from CHIP, an{rd geocode the counties based off a csv of "official administrative codes".

We'll also poke at the data a bit, looking at changes over time.

Section B

##CHIP

CHIP (China Household Income Project) is put out by the CIID Beijing as a longitudinal survey. It's been happening since 1988 and includes all kinds of juicy stuff including land use.

Load up necessary libraries. Some data is in `.dta` which is Stata file.

```
library(tidyr)
library(tidyverse)
```

```
## Warning: package 'tibble' was built under R version 3.4.3
```

```
## Warning: package 'stringr' was built under R version 3.4.3
```

```
library(dplyr)
library(foreign)
library(reticulate)
```

```
## Warning: package 'reticulate' was built under R version 3.4.4
```

```
library(haven)
library(ggmap)
library(sf)
```

```
## Warning: package 'sf' was built under R version 3.4.3
```

```
chips_rur_1988 <- read_dta('data/1988/09836-0002-Data.dta')
chips_rur_1995 <- read_tsv('data/1995/DS0002/03012-0002-Data.tsv')
```

```
## Warning in rbind(names(probs), probs_f): number of columns of result is not
## a multiple of vector length (arg 1)
```

```
## Warning: 198 parsing failures.
## row # A tibble: 5 x 5 col     row col     expected               actual file
## ... ................. ... ..........................................................................
## See problems(...) for more details.
```

```
chips_rur_2002<- read_tsv('data/2002/DS0006/21741-0006-Data.tsv')
```

```
## Warning in rbind(names(probs), probs_f): number of columns of result is not
## a multiple of vector length (arg 1)
```

```
## Warning: 244 parsing failures.
## row # A tibble: 5 x 5 col     row col   expected               actual           file
## ... ................. ... ..................................................................
## See problems(...) for more details.
```

```
chips_rur_2007abc <- read_dta('data/2007 (2008)/RHS_w1_abc.dta')
chips_rur_2007d <- read_dta('data/2007 (2008)/RHS_w1_d.dta')
chips_rur_2007e1 <- read_dta('data/2007 (2008)/RHS_w1_e1.dta')
chips_rur_2007e2 <- read_dta('data/2007 (2008)/RHS_w1_e2.dta')
chips_rur_2007e3 <- read_dta('data/2007 (2008)/RHS_w1_e3.dta')
chips_rur_2007e4 <- read_dta('data/2007 (2008)/RHS_w1_e4.dta')
chips_rur_2007hhiexp <- read_dta('data/2007 (2008)/CHIP2007_income_and_expenditure_20150408.dta')
chips_rur_2008abc <- read_dta('data/2008 (2009)/RHS_w2_abc.dta')
chips_rur_2008d <- read_dta('data/2008 (2009)/RHS_w2_d.dta')
chips_rur_2008e <- read_dta('data/2008 (2009)/RHS_w2_e.dta')
chips_rur_2008f <- read_dta('data/2008 (2009)/RHS_w2_f.dta')
chips_rur_2008hgsg <- read_dta('data/2008 (2009)/RHS_w2_hgsg.dta')
chips_rur_2008hijk <- read_dta('data/2008 (2009)/RHS_w2_hijk.dta')
chips_rur_2008vill <- read_dta('data/2008 (2009)/RHS_w2_vill.dta')
chips_rur_2013 <- read_dta('data/2013/CHIP2013_rural_household_f_income_asset.dta')


name_vill_id_2007 <- read_dta('data/2007 (2008)/name_id_and_village_id_20151010.dta')
```

Table of columns used:

Year	Net household income	Land cultivated	Number of rooms in House	Fixed production assets	Total household exp on production
1988	HNET88	LAT	HHO	VHPFP	EFP88
1995	B602	B801	B1001	B804_1	B7130
2002	na	na	na	na	na
2007	income_net	na	na	na	na
2009	na	H01	na	K01	na
2013	F01_1	L01_1	na	F07_1 + F07_2	F02_1

```
# Filter out some data from 1988 because there's missing values. They got rid of missing values in late

chips_rur_1988_filt <- chips_rur_1988 %>% filter(HNET88 != 99999999, LAT != 999.9, HHO != 99, VHPFP != 9

base::mean(chips_rur_1988_filt$HNET88)
```

```
## [1] 2739.51
```

```
base::mean(chips_rur_1995$B602)
```

```
## [1] 6812.06
```

```
base::mean(chips_rur_2007hhiexp$income_net)
```

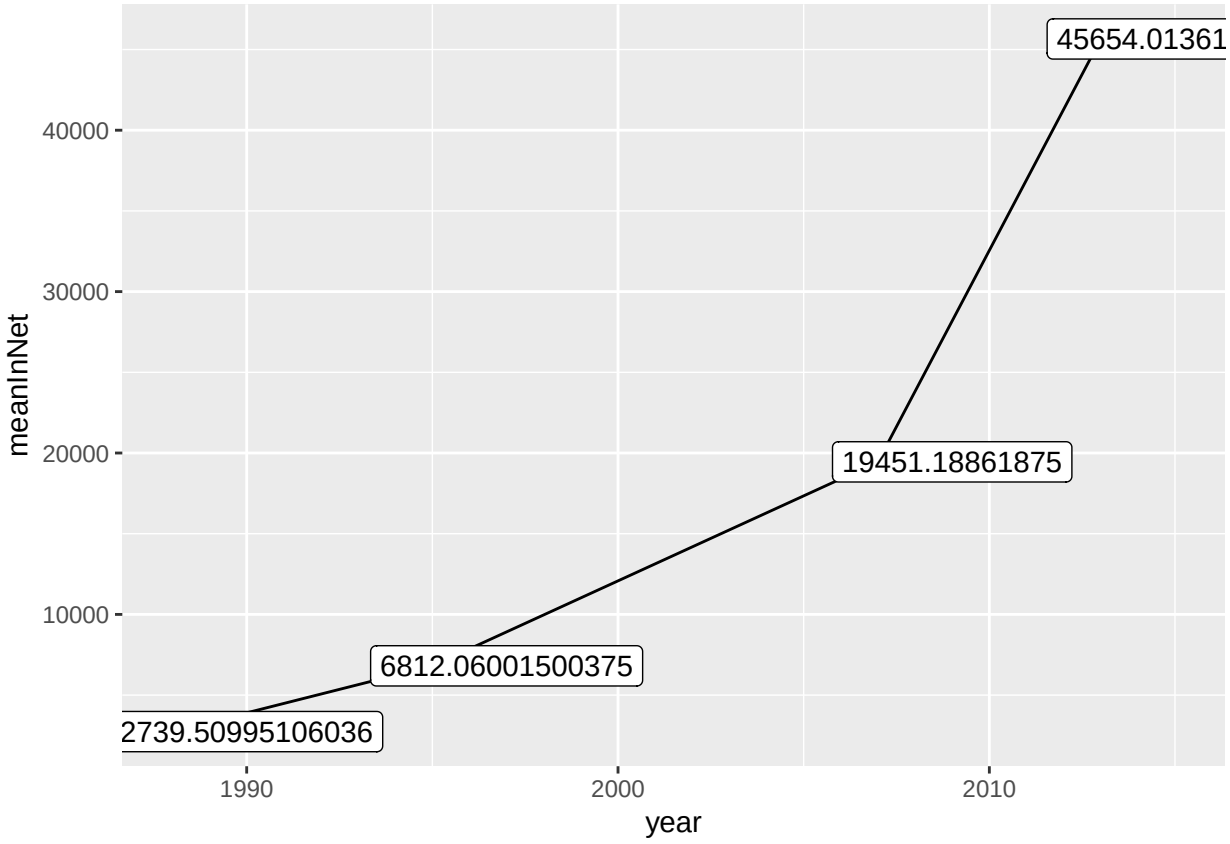
```
## [1] 19451.19
```

```
base::mean(chips_rur_2013$f01_1, na.rm=TRUE)
```

```
## [1] 45654.01
```

```
meanNetIncome <- new_tibble(list(year = c(1988,1995,2007,2013),
                     meanInNet = c(base::mean(chips_rur_1988_filt$HNET88), base::mean(chips_rur_1995$B6

ggplot(meanNetIncome, aes(year, meanInNet)) +
  geom_line() +
  geom_point() +
  geom_label(label=meanNetIncome$meanInNet, nudge_x = 2, nudge_y = 1)
```




```
chips_rur_1988_filt <- chips_rur_1988_filt %>%
  mutate(., PROVCOUNTY = paste0(PROVINCE, COUNTY))



#A1 is county and city code
chips_rur_1995 %>%
  mutate(., PROVCOUNTY = paste0(PROVINCE, COUNTY))
```

```
## # A tibble: 7,998 x 262
##        A1  B101  B401  B402  B403  B404  B405  B406  B407 B407A  B408
##     <int> <int> <int> <int> <int> <int> <int> <int> <int> <int> <int>
```

```
##  1 110221   101     1     1     2     2     2     2     2     0     1
##  2 110221   102     1     1     2     2     2     2     2     0     1
##  3 110221   103     1     1     2     2     2     2     2     0     1
##  4 110221   104     1     1     2     2     2     2     2     0     1
##  5 110221   105     1     1     2     2     2     2     2     0     1
##  6 110221   106     1     1     2     2     2     2     2     0     1
##  7 110221   107     1     1     2     2     2     2     2     0     1
##  8 110221   108     1     1     2     2     2     2     2     0     1
##  9 110221   109     1     1     2     2     2     2     2     0     1
## 10 110221   110     1     1     2     2     2     2     2     0     1
## # ... with 7,988 more rows, and 251 more variables: B409 <int>,
## #   B410 <int>, B411 <int>, B412 <int>, B412A <int>, B412B <int>,
## #   B412C <int>, B412D <int>, B413 <int>, B414 <int>, B501_1 <int>,
## #   B501_2 <int>, B501_3 <int>, B501A_1 <int>, B501A_2 <int>,
## #   B501A_3 <int>, B501B_1 <int>, B501B_2 <int>, B501B_3 <int>,
## #   B501C_1 <int>, B501C_2 <int>, B501C_3 <int>, B501D_1 <int>,
## #   B501D_2 <int>, B501D_3 <int>, B501E_1 <int>, B501E_2 <int>,
## #   B501E_3 <int>, B501F_1 <int>, B501F_2 <int>, B501F_3 <int>,
## #   B502_1 <int>, B502_2 <int>, B502_3 <int>, B502A_1 <int>,
## #   B502A_2 <int>, B502A_3 <int>, B502B_1 <int>, B502B_2 <int>,
## #   B502B_3 <int>, B502C_1 <int>, B502C_2 <int>, B502C_3 <int>,
## #   B502D_1 <int>, B502D_2 <int>, B502D_3 <int>, B502E_1 <int>,
## #   B502E_2 <int>, B502E_3 <int>, B502F_1 <int>, B502F_2 <int>,
## #   B502F_3 <int>, B502G_1 <int>, B502G_2 <int>, B502G_3 <int>,
## #   B502H_1 <int>, B502H_2 <int>, B502H_3 <int>, B503 <int>, B504 <int>,
## #   B504A <int>, B504B <int>, B504C <int>, B505 <int>, B506 <int>,
## #   B506A <int>, B506B <int>, B507 <int>, B508 <int>, B509 <int>,
## #   B510 <int>, B511_1 <int>, B511_2 <int>, B511A_1 <int>, B511A_2 <int>,
## #   B511B_1 <int>, B511B_2 <int>, B511C_1 <int>, B511C_2 <int>,
## #   B511D_1 <int>, B511D_2 <int>, B600 <int>, B601 <int>, B602 <int>,
## #   B700 <int>, B700A <int>, B701 <int>, B702 <int>, B703 <int>,
## #   B703A <int>, B703B <int>, B704 <int>, B705 <int>, B706 <int>,
## #   B707 <int>, B708 <int>, B708A <int>, B708B <int>, B708C <int>,
## #   B709 <int>, ...
```

##County level variations

```
chips_rur_1988_filt %>%
  group_by(PROVCOUNTY) %>%
  summarise(mean=base::mean(HNET88))
```

```
## # A tibble: 334 x 2
##    PROVCOUNTY  mean
##    <chr>      <dbl>
##  1 11222      5204.
##  2 11223      2503.
##  3 11228      3403.
##  4 12111      5210.
##  5 12120      4803.
##  6 12222      4145.
##  7 12224      3269.
##  8 13121      2248.
##  9 132122     1888.
## 10 13221      3166.
## # ... with 324 more rows
```

```r
chips_rur_1995 %>%
  group_by(COUNTY) %>%
  summarise(mean=base::mean(B602))
```

```
## # A tibble: 91 x 2
##    COUNTY   mean
##     <int>  <dbl>
##  1    111 24959.
##  2    121  6359.
##  3    124  8402.
##  4    125  2406.
##  5    130  5735.
##  6    220  9591.
##  7    221  9173.
##  8    222  3419.
##  9    223  7903.
## 10    225 14394.
## # ... with 81 more rows
```

```r
chips_rur_2007hhiexp %>%
  group_by(name_id) %>%
  summarise(mean=base::mean(income_net))
```

```
## # A tibble: 8,000 x 2
##       name_id   mean
##         <dbl>  <dbl>
##  1 130181001. 27923.
##  2 130181002. 44426.
##  3 130181003. 17771.
##  4 130181004. 22702.
##  5 130181005. 13504.
##  6 130181006. 22906.
##  7 130181007. 35729.
##  8 130181008. 15200.
##  9 130181009. 24412.
## 10 130181010. 11323.
## # ... with 7,990 more rows
```

```r
chips_rur_2013 %>%
  group_by(coun) %>%
  summarise(mean=base::mean(f01_1))
```

```
## # A tibble: 200 x 2
##       coun   mean
##      <dbl>  <dbl>
##  1 110106. 49088.
##  2 110112. 53410.
##  3 110114. 37207.
##  4 110117. 54127.
##  5 110229. 47183.
##  6 140106. 52508.
##  7 140181. 53273.
##  8 140225. 29111.
##  9 140321. 34723.
## 10 140603. 31185.
```

```
## # ... with 190 more rows
```

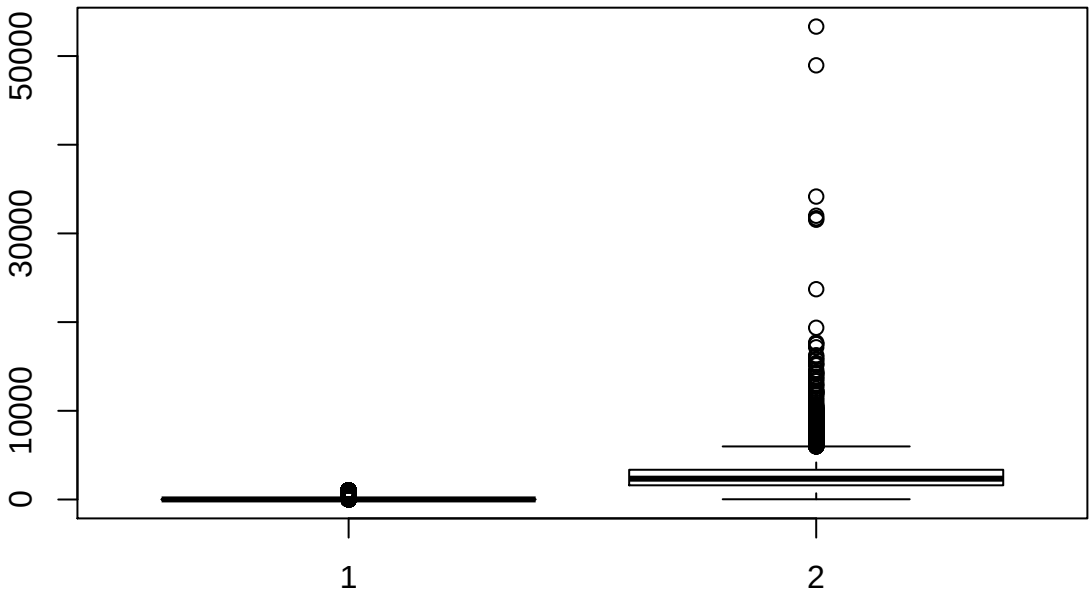
```
head(chips_rur_2013)
```

```
## # A tibble: 6 x 37
##   hhcode       coun  f01_1  f01_2  f01_3  f02_1 f02_2    f03  f03_1  f03_2
##   <chr>       <dbl>  <dbl>  <dbl>  <dbl>  <dbl> <dbl>  <dbl>  <dbl>  <dbl>
## 1 110106530~ 1.10e5 18598. 18000. 16000. 30660.    0.     0.     0.     0.
## 2 110106530~ 1.10e5 26700. 26700. 26700. 21873.    0.  4830.  4830.     0.
## 3 110106530~ 1.10e5 33462. 33462. 33462. 26695.    0.  6767.  6767.     0.
## 4 110106531~ 1.10e5 38000. 38000. 38000.  9544.    0. 28456. 28456.     0.
## 5 110106610~ 1.10e5 25904. 22000. 20000. 14696.    0.  3000.  3000.     0.
## 6 110106610~ 1.10e5 45844. 27844. 27000. 52216.    0. 33000.  3000. 20000.
## # ... with 27 more variables: f03_3 <dbl>, f03_4 <dbl>, f03_5 <dbl>,
## #   f03_6 <dbl>, f03_7 <dbl>, f03_8 <dbl>, f03_9 <dbl>, f03_10 <dbl>,
## #   f03_11 <dbl>, f03_12 <dbl>, f04 <dbl>, f05 <dbl>, f05_1 <dbl>,
## #   f05_2 <dbl>, f05_3 <dbl>, f05_4 <dbl>, f05_5 <dbl>, f05_6 <dbl>,
## #   f06_1 <dbl>, f06_2 <dbl>, f06_3 <dbl>, f06_4 <dbl>, f06_5 <dbl>,
## #   f06_6 <dbl>, f07_1 <dbl>, f07_2 <dbl>, f08 <dbl>
```

##dice off last 3 digits from `name id` of chips_rur_2007hhiexp

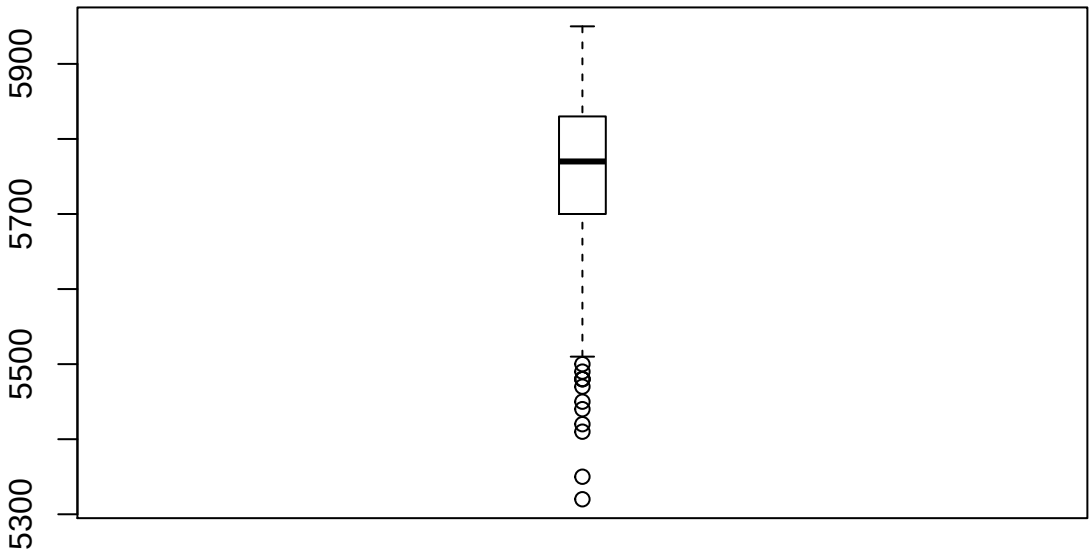
(the first set)

```
outlier_values_1988 <- boxplot.stats(chips_rur_1988_filt$HNET88)$out  # outlier values.
outlier_values_1995 <- boxplot.stats(chips_rur_1995$B602)$out  # outlier values.
outlier_values_2007 <- boxplot.stats(chips_rur_2007hhiexp$income_net)$out  # outlier values.
outlier_values_2013 <- boxplot.stats(chips_rur_2013$f01_1)$out  # outlier values.
boxplot(chips_rur_1988_filt$LAT, chips_rur_1988_filt$HNET88)
```




```
url <- "http://rstatistics.net/wp-content/uploads/2015/09/ozone.csv"
# alternate source:  https://raw.githubusercontent.com/selva86/datasets/master/ozone.csv
inputData <- read.csv(url)  # import data

outlier_values <- boxplot.stats(inputData$pressure_height)$out  # outlier values.
boxplot(inputData$pressure_height, main="Pressure Height", boxwex=0.1)
```




```
#mtext(paste("Outliers: ", paste(outlier_values, collapse=", ")), cex=0.6)
```

Geocode county addresses based off the county codes

Data from https://raw.githubusercontent.com/modood/Administrative-divisions-of-China/, basically there's a county code and name for each county. They are not geocoded however (by geocoded I mean "center" of county)

#Add long lat columns to the county code table

```
counties_main <- read_csv('https://raw.githubusercontent.com/modood/Administrative-divisions-of-China/ma

counties_main
```

```
## # A tibble: 2,856 x 4
##      code name     cityCode provinceCode
##     <int> <chr>       <int>        <int>
##  1 110101          1101           11
##  2 110102          1101           11
##  3 110105          1101           11
##  4 110106          1101           11
##  5 110107        1101           11
##  6 110108          1101           11
##  7 110109        1101           11
##  8 110111          1101           11
##  9 110112          1101           11
## 10 110113          1101           11
## # ... with 2,846 more rows
```

```
counties_main$lat <- 'NA'
counties_main$long <- 'NA'

city_main <- read_csv('https://raw.githubusercontent.com/modood/Administrative-divisions-of-China/maste
province_main <- read_csv('https://raw.githubusercontent.com/modood/Administrative-divisions-of-China/ma
```

```
prov_county_main <- left_join(counties_main, province_main,
                              by=c('provinceCode'='code'))

prov_county_main <- prov_county_main %>%
    rename('cityName' = name.y,
           'countyName' = name.x) %>%
    mutate(., geocodeAdd = paste0(cityName,countyName))


prov_county_main
```

```
## # A tibble: 2,856 x 8
##      code countyName cityCode provinceCode lat   long  cityName geocodeAdd
##     <int> <chr>         <int>        <int> <chr> <chr> <chr>    <chr>
##  1 110101            1101           11 NA    NA            ~
##  2 110102            1101           11 NA    NA            ~
##  3 110105            1101           11 NA    NA            ~
##  4 110106            1101           11 NA    NA            ~
##  5 110107          1101           11 NA    NA            ~
##  6 110108            1101           11 NA    NA            ~
##  7 110109          1101           11 NA    NA            ~
##  8 110111            1101           11 NA    NA            ~
##  9 110112            1101           11 NA    NA            ~
## 10 110113            1101           11 NA    NA            ~
## # ... with 2,846 more rows
```

Set an API key for ggmap so we don't go over the limit

```
ggmap_credentials()
register_google(key='insertyourkeyhere')
library(ggmap)
ggmap_credentials()
```

Divide the table into two so as to preempt going over the free geocoding limit

```
# pcoun_codes_1 <- filter(prov_county_main, code < 400000)
# pcoun_codes_2 <- filter(prov_county_main, code >= 400000)
```

```
# infile <- 'pcoun_2'
# data <-pcoun_codes_2
#
# addresses = data$geocodeAdd
#
# #define a function that will process googles server responses for us.
# getGeoDetails <- function(address){
#    #use the gecode function to query google servers
#    geo_reply = geocode(address, output='all', messaging=TRUE, override_limit=TRUE)
#    #now extract the bits that we need from the returned list
#
#    answer <- data.frame(lat=NA, long=NA, accuracy=NA, formatted_address=NA, address_type=NA, status=N
```

```
#    answer$status <- geo_reply$status
#
#    #if we are over the query limit - want to pause for an hour
#    while(geo_reply$status == "OVER_QUERY_LIMIT"){
#        print("OVER QUERY LIMIT - Pausing for 1 hour at:")
#        time <- Sys.time()
#        print(as.character(time))
#        Sys.sleep(60*60)
#        geo_reply = geocode(address, output='all', messaging=TRUE, override_limit=TRUE)
#        answer$status <- geo_reply$status
#    }
#
#    #return Na's if we didn't get a match:
#    if (geo_reply$status != "OK"){
#        return(answer)
#    }
#    #else, extract what we need from the Google server reply into a dataframe:
#    answer$lat <- geo_reply$results[[1]]$geometry$location$lat
#    answer$long <- geo_reply$results[[1]]$geometry$location$lng
#    if (length(geo_reply$results[[1]]$types) > 0){
#        answer$accuracy <- geo_reply$results[[1]]$types[[1]]
#    }
#    answer$address_type <- paste(geo_reply$results[[1]]$types, collapse=',')
#    answer$formatted_address <- geo_reply$results[[1]]$formatted_address
#
#    return(answer)
# }
#
# #initialise a dataframe to hold the results
# geocoded <- data.frame()
# # find out where to start in the address list (if the script was interrupted before):
# startindex <- 1
# #if a temp file exists - load it up and count the rows!
# tempfilename <- paste0(infile, '_temp_geocoded.rds')
# if (file.exists(tempfilename)){
#        print("Found temp file - resuming from index:")
#        geocoded <- readRDS(tempfilename)
#        startindex <- nrow(geocoded)+1
#        print(startindex)
# }
#
# # Start the geocoding process - address by address. geocode() function takes care of query speed limi
# for (ii in seq(startindex, length(addresses))) {
#    print(paste("Working on index", ii, "of", length(addresses)))
#    #query the google geocoder - this will pause here if we are over the limit.
#
#    result = getGeoDetails(addresses[ii])
#    print(result$status)
#    result$index <- ii
#    #append the answer to the results file.
#    geocoded <- rbind(geocoded, result)
#    #save temporary results as we are going along
#    saveRDS(geocoded, tempfilename)
```

```
# }
```

```
# geocodedTable <- data.frame(matrix(ncol = 3, nrow = 1516))
# #now we add the latitude and longitude to the main data
#
# geocodedTable$status <- geocoded$status
# geocodedTable$formatted_address <- geocoded$formatted_address
# geocodedTable$index <- geocoded$index
# geocodedTable$lat <- geocoded$lat
# geocodedTable$long <- geocoded$long
# geocodedTable$accuracy <- geocoded$accuracy
#
#
#
# #finally write it all to the output files
# saveRDS(data, paste0("../data/", infile ,"_geocoded.rds"))
# write.table(geocodedTable, file=paste0("", infile ,"_geocoded.csv"), sep=",", row.names=FALSE)
```

Now there's two tables, put them together vertically

```
a <- read_csv('data/pcoun_1_geocoded.csv')
b <- read_csv('data/pcoun_2_geocoded.csv')
geocoded_areas <- bind_rows(a,b)

prov_county_main$lat <- geocoded_areas$lat
prov_county_main$long <- geocoded_areas$long

prov_county_main$engAdd <- geocoded_areas$formatted_address
prov_county_main
```

```
## # A tibble: 2,856 x 9
##      code countyName cityCode provinceCode   lat  long cityName geocodeAdd
##     <int> <chr>         <int>        <int> <dbl> <dbl> <chr>    <chr>
##  1 110101            1101           11  39.9  116.         ~
##  2 110102            1101           11  39.9  116.         ~
##  3 110105            1101           11  39.9  116.         ~
##  4 110106            1101           11  39.9  116.         ~
##  5 110107          1101           11  39.9  116.         ~
##  6 110108            1101           11  40.0  116.         ~
##  7 110109          1101           11  39.9  116.         ~
##  8 110111            1101           11  39.7  116.         ~
##  9 110112            1101           11  39.9  117.         ~
## 10 110113            1101           11  40.1  117.         ~
## # ... with 2,846 more rows, and 1 more variable: engAdd <chr>
```

Taobao villages

Geocoding taobao villages

There are 1312 taobao villages as of 2017. This is under the google geocoding api limit, yay.

Testing out the geocoding response, put the province and village together (column + column, separate by comma)