Import income insufficiency data

Importing income insufficiency data

```r
library(tidyverse)
library(readxl)
library(tabulizer)

source('../income_ins_functions.R')
```

Consumer Expenditure Survey data

All Consumer Expenditure Survey (CES) data was gathered from the BLS's CES website. The following expenses were collected from CES data: health insurance (by household size and age), transportation, apparel and services, housekeeping supplies personal care products and services, reading, and misc.

The one screen data source application was used (https://data.bls.gov/PDQWeb/cx) for all CES expenses except for health insurance by age. From the one screen data source page, we selected the following items from boxes:

- For all items, 'Expenditures' was selected from 'Select One Category'.
- The selections in the 'Select one or more items' box depend on the item selected in the 'Subcategory' box. For each of the following subcategories, the corresponding 'Select one or more items' lines were selected:
  - Healthcare: health insurance;
  - Transportation: Vehicle purchases: Cars and trucks (used); Other vehicle expenses; Gasoline, other fuels, and motor oil; and Public and other transportation;
  - Apparel and services: Apparel and services
  - Housing: Housekeeping supplies;
  - Personal care products and services: Personal care products and services;
  - Reading: Reading; and
  - Miscellaneous expenditures: Miscellaneous expenditures.
- In the 'Select one or more demographics' box, size of consumer unit and region of residence were selected.
- Finally, the following items were selected from 'Select one or more characteristics: one person in consumer unit, two people in consumer unit, three people in consumer unit, four people in consumer unit, five or more people in consumer unit, all consumer units (LB1101), and region of residence: south.
- We then clicked 'Add to selection' after selections were made, followed by 'Get data'.
- From the next page we clicked 'More formatting options'.
- To create the raw data, we selected the following options:
  - Specify year range: 2006 to most recent year;
  - Select view of the data > Multiseries table;
  - Output type > Text > tab delimited; and
  - click 'Retrieve data'.
- This takes us to a page with the raw data for all years and a given expense. We copied the text of the data into a .txt file.
- To extract column headers, we used the following option:
  - Specify year range: 2006 to most recent year;
  - Select view of the data > column format;
  - Output type > Text > tab delimited; and
  - click 'Retrieve data'.

- Like with the raw data, we then copy and pasted all output into a .txt file.

We used cross-tabulated tables for health insurance expenses by age and consumer unit size (https://www.bls.gov/cex/csxcross.htm). From this page we selected the excel file for each year under size of consumer unit by age of reference person > One person.

Import and clean CES data (minus health insurance by age and consumer unit size)

All data except for health insurance by size of consumer unit and age contain a common format. Therefore, we can import and clean these dataset with a single function and for loop, and combine these datasets into one dataset.

```r
# create vectors of file names for raw data and headers
# we will loop through the file names, importing and cleaning data

# path to folder containing ces data
ces_path <- 'raw_data/ces/'

# all raw data files end in raw_data.txt; therefore, we can identiy all such files in folder
data_files <- list.files(ces_path, pattern = 'raw_data.txt') %>%
  # sort alphabetically so that raw data order matches header order
  sort()
# header files contain 'header'
header_files <- list.files(ces_path, pattern = 'header') %>%
  sort()

# initialize dataframe to contain all ces data
df_ces <- data.frame()

# loop through each data file, import and clean data, and add to dataframe containing ces data
for (i in seq_along(data_files)) {
  print(i)
  # import ces expense file
  df_ces_single <- ff_clean_ces(paste0(ces_path, data_files[i]),
                                paste0(ces_path, header_files[i]))
  # add to data frame containing all ces expenses
  df_ces <- bind_rows(df_ces, df_ces_single)
}

# we want to replace string describing consumer unit size with integer
# create replacement values; must use character to replace other characters
unit_size_int <- as.character(1:length(unique(df_ces$Characteristics)))
# ad descriptions as names; these are the values we want to replace
names(unit_size_int) <- unique(df_ces$Characteristics)

df_ces <- df_ces %>%
  select(Year, region_ratio, Characteristics, Item, Subcategory, Expense) %>%
  # rename column names to lover case
  rename_all(funs(tolower(.))) %>%
  rename(consumer_unit_size = characteristics) %>%
  # replace strings of consumer unit size with integer
  mutate(consumer_unit_size = str_replace_all(consumer_unit_size, unit_size_int),
         consumer_unit_size = as.integer(consumer_unit_size)) %>%
  # some of the columns, but not all, have start and end years
  # extract these years and place in start_age and end_age columns
  mutate(start_age = ifelse(str_detect(item, 'under 2'), 0,
                            ifelse(str_detect(item, '2 to 15'), 2,
                                   ifelse(str_detect(item, '16 and over'), 16,
                                                   # make age very old if no conditions are met
                                                   150))),
         end_age = ifelse(str_detect(item, 'under 2'), 1,
                          ifelse(str_detect(item, '2 to 15'), 15,
                                                   150))) %>%
  # come items have gender categories
  # extract gender and place in its own column,
  # with items not having gender labeled as 0
  mutate(gender = ifelse(str_detect(item, ' Men|Boys'), 1,
                         ifelse(str_detect(item, 'Women|Girls'), 2, 0)))
```

Import and clean CES data for health insurance by age and consumer unit size

For health insurance by age and consumer unit size, each year is in a different file and all files share a common format. The year is represented in the final two digits of the file name. The block below loop through each file, importing and cleaning along the way.

```r
# get list of all files
health_age_files <- list.files(paste0(ces_path, 'health_ins_age'))

# initialize dataframe to store all health insurance by age data
ces_health <- data.frame()

# loop through each file
for (file_name in health_age_files) {

  # extract year from filename
  single_year <- str_match(file_name, '_([0-9][0-9]).')[2] %>%
    # convert to integer
    as.integer() %>%
    # year is only two integers (example: 06); so add 2000 to convert to 2006
    sum(2000)

  # import data
  health_age <- read_excel(paste0(ces_path, 'health_ins_age/', file_name), skip = 2) %>%
    # only select items column and two age columns
    select(Item, starts_with('65-74'), starts_with('75 years')) %>%
    # trim whitespace in all columns
    mutate_all(funs(str_trim(., side = 'both'))) %>%
    # only keep row showing health insurance expenses
    filter(Item == 'Health insurance') %>%
    # add year
    mutate(year = single_year)

  # change column names to ensure they are the same for every dataset
  colnames(health_age) <- c('Item', '65_74', '75_plus', 'Year')

  # add singlge year to dataframe containing all years
  ces_health <- bind_rows(ces_health, health_age)
}

# convert fron wide (each age group is in its own row)
# to long (each age group is in a different row)
ces_health <- gather(ces_health, 'age_group', 'expense', `65_74`:`75_plus`) %>%
  # change age group names to number representing start age
  mutate(start_age = str_replace_all(age_group, '65_74', '65')) %>%
  mutate(start_age = str_replace_all(start_age, '75_plus', '75')) %>%
  # create end age
  mutate(end_age = ifelse(start_age == 65, 74, 150)) %>%
  # convert expense and ages to integer
  mutate(expense = as.integer(expense),
         start_age = as.integer(start_age),
         end_age = as.integer(end_age)) %>%
  select(-age_group) %>%
  # convert column to lowercase
  rename_all(funs(tolower(.))) %>%
  # add subcategory to match other ces dataset
  mutate(subcategory = 'Healthcare') %>%
  # this dataset is only used for individuals of the given age
  # therefore, the consumer unit size will be one
  mutate(consumer_unit_size = 1,
         # add gender to match primary ces dataset
         gender = 0)
```

The cross tab of health insurance by consumer unit and age does not contain regional information. So, we will apply the regional ratio from the health insurance by one consumer unit data.

```r
# get ratio for health insurance by one consumer unit
ratio_one <- df_ces %>%
  filter(consumer_unit_size == 1,
         item == 'Health insurance') %>%
  # we only need year and ratio, which will be merged with the health ins by consumer unit and age df
  select(year, region_ratio)

# add ratio to health insurance by age and consumer unit dataframe
ces_health <- left_join(ces_health, ratio_one, by = c('year'='year')) %>%
  mutate(expense = round( expense * region_ratio, 0 )) %>%
  select(-region_ratio)
```

Two CES files are now cleaned and ready for export:

1. A file containing all CES data except health insurance by age and consumer unit size. This includes the following:
   - Health insurance by consumer unit,
   - Transportation,
   - Apparel and services,
   - Housekeeping supplies,
   - Personal care products,
   - Reading, and
   - Miscellaneous.
2. Health insurance by age and consumer unit size.

These two files can be merged into one and written out.

```r
# merge ces files and write out
df_ces %>%
  select(-region_ratio) %>%
  bind_rows(ces_health) %>%
  write_csv('cleaned_data/ces.csv')
```

Medical Expenditure Panel Survey data

Healthcare expenditures beyond insurance were gathered from the Medical Expenditure Panel Survey (MEPS). We used the MPESNet household component query tool (https://www.meps.ahrq.gov/mepsweb/data_stats/MEPSnetHC/startup) to collect the data. The following variables were used:

- Geographic variables > Region,
- Demographic variables > Age, and
- Utilization, expenditure and SOP variables > All types of service > TOTAL AMT PAID BY SELF/FAMILY.

Under the Descriptive Statistics tab we selected 'Minimums, Maximums, Sums, Means, and Medians'. Then, TOTSLF was selected for the variable statistic and the median and mean were checked. Final, we created a cross-tab of age and region. Clicking the 'Show Statistics' button reveals medical expenditures by age and region. The table labeled Standard Errors was copy and pasted into an Excel file, and the year was added as an additional column. Subsequent years were added to the same excel file.

The following code block imports and cleans this data.

```r
# import MEPS data
meps <- read_csv('raw_data/meps_raw.csv') %>%
  # only keep needed columns
  select(age, region, year, mean) %>%
  # remove rows where age is total
  filter(age != 'TOTAL',
         # only keep south region
         region == '3 SOUTH') %>%
  # no longer need region variable, since it was only needed for filtering
  select(-region) %>%
  # only keep first year of age range,
  # this results in only keeping the start age year, which is what is used in other expenses
  mutate(start_age = str_extract(age, '[0-9]*')) %>%
  select(-age)
```

2015 is the most recent year for MEPS data. 2016 and 2017 were estimated by using 2015 values and adjusting them for inflation.

```r
# create table of adjustment factors
inf_adj <- ff_inflation_adj_table(2015) %>%
  # don't need cpi
  select(-cpi)

# create dataset of 2015
meps_inflation <- meps %>%
  filter(year == 2015)

# add this dataset to itself as additiona lrows
# this creates the dataset twice, one for 2016 and one for 2017
meps_inflation <- meps_inflation %>%
  bind_rows(meps_inflation) %>%
  # add 2016 and 2017: this adds 2016 6 times and 2017 6 times
  mutate(year = c(rep(2016, nrow(.)/2),
                  rep(2017, nrow(.)/2))) %>%
  # add inflation adjustment factor
  left_join(inf_adj, by = 'year') %>%
  # divide by inflation adjustment factor to convert 2015 price to 2016 and 2017
  mutate(mean = mean / adj_factor) %>%
  select(-adj_factor)

# add 2016 and 2017 inflation adjusted estimates to data frame containing 2006-2015
meps <- bind_rows(meps_inflation, meps) %>%
  mutate(mean = round(mean, 0)) %>%
  # add description variable
  mutate(item = 'Medical expenses') %>%
  # change names to match CES data format
  rename(expense =  mean) %>%
  # change column order to match CES
  select(item, year, start_age, expense)

# write out data
# write_csv(meps, 'cleaned_data/meps.csv')
```

Fair Market Rent (FMR)

Housing expenses were gathered from HUD's Fair Market Rents (https://www.huduser.gov/portal/datasets/fmr.html). Yearly data was downloaded through a url. Each yearly dataset contains estimated rents by bedroom size for all counties in the United States.

```r
# vector of urls for each yearly dataset
url <- c( 'https://www.huduser.gov/portal/datasets/fmr/fmr2017/FY2017-4050-County-Level_Data.xlsx',
'https://www.huduser.gov/portal/datasets/fmr/fmr2016f/FY2016F_4050_Final.xlsx',
'https://www.huduser.gov/portal/datasets/fmr/fmr2015f/FY2015_4050_RevFinal.xls',
'https://www.huduser.gov/portal/datasets/fmr/fmr2014f/FY2014_4050_RevFinal.xls',
'https://www.huduser.gov/portal/datasets/fmr/fmr2013f/FY2013_4050_Final.xls',
'https://www.huduser.gov/portal/datasets/fmr/fmr2012f/FY2012_4050_Final.xls',
'https://www.huduser.gov/portal/datasets/fmr/fmr2011f/FY2011_4050_Final.xls',
'https://www.huduser.gov/portal/datasets/fmr/fmr2010f/FY2010_4050_Final_PostRDDs.xls',
'https://www.huduser.gov/portal/datasets/fmr/fmr2009r/FY2009_4050_Rev_Final.xls',
'https://www.huduser.gov/portal/datasets/fmr/fmr2008r/FMR_county_fy2008r_rdds.xls',
'https://www.huduser.gov/portal/datasets/fmr/fmr2007f/FY2007F_County_Town.xls',
'https://www.huduser.gov/portal/datasets/fmr/fmr2006r/FY2006_County_Town.xls')

# itereate through each url, downloading data and placing into a list

# initialize list to store a dataframe for each year
fmr_list <- list()

for (yr in seq_along(url)) {
  str_extract(url[yr], 'FY.*|FMR.*')

  # extract year from url
  single_year <- as.integer(str_match(url[yr], 'fmr([0-9]{4})')[2])
  # extract filename from url
  destfile <- str_extract(url[yr], 'FY.*|FMR.*')

  curl::curl_download(url[yr], destfile)

  # import yearly dataset
  df <- read_excel(destfile) %>%
    # only selected needed rows
    # we are not using 0 bedroom apartments (fmr0)
    # need populations to create weighted average of rents for entire NC
    select('state_alpha', 'countyname', 'fmr1', 'fmr2', 'fmr3', 'fmr4', starts_with('pop20')) %>%
    # fitler for NC counties
    filter(state_alpha == 'NC') %>%
    # add year
    mutate(year = single_year)

  # add dataset to list
  fmr_list[[yr]] <- df
}

# bind all yearly datasets into a single dataset
fmr <- bind_rows(fmr_list) %>%
  # remove state; all counties are in NC so state is irrelevant
  select(-state_alpha)
```

The block above imported rent data for all NC counties. To create an NC average we need to weight the county rents by county population.

```r
# there are two population columns: pop2010 and pop2000. Combine into one column.
# We can do this by summing the two columns and ignoring NA values
fmr$pop <- rowSums(fmr[, c('pop2000', 'pop2010')], na.rm = TRUE)

# remove old popualtion columns and
fmr <- select(fmr, -pop2000, -pop2010)
```

We need an additional dataset with weighted average rents for NC.

```r
### create NC dataset with rents weighted by population

# create dataset that is the yearly NC population;
# this will be used to calculate each county's percentage of the population
nc_pop <- fmr %>%
  group_by(year) %>%
  summarize(nc_pop = sum(pop))


fmr_nc <- fmr %>%
  # create an additional column in the fmr datset showing yearly NC population
  left_join(nc_pop, by='year') %>%
  # calculate each county' percentage of the NC population
  mutate(perc_pop = pop / nc_pop)

# write out dataset with county populations and precentage of populatio nfor use with other expenses
fmr_nc %>%
  select(countyname, year, pop, perc_pop) %>%
  write_csv('county_perc_pop.csv')

fmr_nc <- fmr_nc %>%
  # multiply each rent value by the county's percentage of the population
  mutate_at(c('fmr1', 'fmr2', 'fmr3', 'fmr4'),
            funs(. * perc_pop)) %>%
  # for each year, sum the values of each rent column
  group_by(year) %>%
  summarize_at(c('fmr1', 'fmr2', 'fmr3', 'fmr4'), sum)

# add the state rent dataframe to the county dataframe
fmr <- fmr_nc %>%
  # add a column to state df called countyname to match county df
  mutate(countyname = 'NC') %>%
  # bind state and county dataframes
  bind_rows(fmr) %>%
  # remove word 'county' from county names
  mutate(countyname = str_replace_all(countyname, ' County', '')) %>%
  # round all rent values to a whole number
  mutate_at(c('fmr1', 'fmr2', 'fmr3', 'fmr4'),
            funs(round(., 0))) %>%
  select(-pop)

# write out fmr dataset
#write_csv(fmr, 'cleaned_data/fmr.csv')
```

Food Costs

Food costs come from the USDA's low cost food plan (https://www.cnpp.usda.gov/USDAFoodPlansCostofFood/reports). This data is trapped in pdf files and we manually entered into a csv file. We then multiplies the values in the csv file by .93 to adjust prices to the South region. The code below minimally cleans the manually entered data.

```
food <- read_delim("raw_data/food_costs.csv",
    "\t", escape_double = FALSE, trim_ws = TRUE) %>%
  # convert from wide (each year is a column) to long (each row is a different year)
  gather('year', 'estimate', `2017`:`2006`) %>%
  mutate(estimate = as.integer(estimate),
         # change age labeling to match other datasets
         Age = str_replace_all(Age, '-', ' to ')) %>%
  mutate(Age = str_replace_all(Age, '[+]', ' plus')) %>%
  # rename variables to match other datasets
  rename(age_group = Age, gender = Gender)

# write out dataset
#write_csv(food, 'cleaned_data/food.csv')
```

Child Care Expenses

Child care expenses were taken from Child Care Aware's annual High Cost of Child Care report for each year. The 2007 state-level report that reflects 2006 data did not contain any North Carolina data, so 2007 rates were inflation adjusted to 2006. The data was manually entered into a csv file from the reports, so no cleaning is necessary.

Old Child Care Expenses

Note: This sections reflects child care expenses from the NC market rate study. It was not used for 2017 income insifficiency calculations.

The North Carolina Division of Child Development and Early Education produces child care market rates for each North Carolina county (https://ncchildcare.ncdhhs.gov/Home/DCDEE-Sections/Subsidy-Services/Market-Rates). These rates are the same as those found in the North Carolina Child Care Market Rate Study (https://ncchildcare.ncdhhs.gov/Portals/0/documents/pdf/M/market_rate_survey_report_2015.pdf). The rates are in pdf files and were copy and pasted into csv files.

North Carolina state-wide averages were calculated by created a weighted average of county rates by population.

Child care costs are available for the following years: 2018, 2016, 2015, 2007 and 2006. 2017 data is listed, but it is not in text readable format. Additionally, there is a sharp increase in costs between 2016 and 2017; and this increase continues through 2018. Therefore, 2017 rates are the average of the inflation adjusted 2016 and 2018 rates..

2008 to 2014 rates were calculated by inflation adjusting the rates of the nearest year. 2008 to 2010 use 2007 rates, while 2011 to 2014 use 2015 rates.

```
# # 2006 data and 2007 and 2015 data are in two separate files.
# # They have the same format, except their columns are in different orders.
# # these can be imported and bound by rows
# child_a <- read_excel('raw_data/child_care/child_06.xlsx') %>%
#    bind_rows(read_excel('raw_data/child_care/child_07_15.xlsx')) %>%
#   # currently each age group is in a different column
#   # convert to long form (each age group is a different row
#   gather('age_group', 'estimate', infant_Sone:school_Sfive)
#
# # 2016 and 2018 pasted in an odd format that needs cleaning
# child_b <- read_excel('raw_data/child_care/child_16_18.xlsx')
#
# # remove county names (this is any row with a letter) and convert to vector
# counties <- child_b %>%
#   # only keep rows that are the county name
#   filter(age_group == 'county') %>%
#   # convert to vector
#   .[[1]]
#
# # remove county names from child_b and create new row with previously created
# # vector of counties that displays the county name
# child_b <- child_b %>%
#   # remove county names
#   filter(age_group != 'county') %>%
#   # add vector of counties as column
#   mutate(county = rep(counties, each = nrow(.)/length(counties))) %>%
#   # there is a column for each year, convert to long form with years on different rows
#   gather('year', 'estimate', sixteen:eighteen) %>%
#   # replace sixteen with 2016 and eighteen with 2018
#   # must first replace with character so some rows and not character and others integer
#   mutate(year = str_replace_all(year, 'sixteen', '2016')) %>%
#   mutate(year = str_replace_all(year, 'eighteen', '2018')) %>%
#   mutate(year = as.integer(year),
#          estimate = as.integer(estimate))
#
# # create 2017 rates by averaging 2016 and 2018
# # these two years are in same file,
# # so group by age and county, and take mean
# # bind onto 2016 and 2018 dataset after creating
# child_b <- child_b %>%
#   group_by(age_group, county) %>%
#   summarize(estimate = round(mean(estimate), 0)) %>%
#   mutate(year = 2017,
#          estimate = as.integer(estimate)) %>%
#   bind_rows(child_b, .)
#
# # we only need four star estimates
# stars <- c('infant_Sfour', 'one_Sfour', 'two_Sfour', 'three_Sfour', 'school_Sfour')
#
# # we will change age group values to integers representing start age
# start_age <- c(0, 1, 2, 3, 6)
#
# # bind child_a and child_b
# child <- bind_rows(child_a, child_b) %>%
#   # only keep four stars
#   filter(age_group %in% stars) %>%
#   # replace age group with integer representing start age
#   mutate(start_age = ifelse(age_group == !!stars[1], !!start_age[1],
#                      ifelse(age_group == !!stars[2], !!start_age[2],
#                        ifelse(age_group == !!stars[3], !!start_age[3],
#                         ifelse(age_group == !!stars[4], !!start_age[4],
#                          ifelse(age_group == !!stars[5], !!start_age[5], 7)))))) %>%
#   select(-age_group)
#
# # we need to calculate rates for missing years based on inflation adjusted rates in the nearest year
# # this requires the same manipulations for all inflation adjustments,
# # so a function will be used to
#
# inf_adj_child_care <- function(child_care_costs, year_create, year_inflate) {
#
#   # inputs:
#   #   child_costs: dataframe of child care costs
#   #   year_create: the year we want to create a dataset for
#   #   year_adjust: the year's data we will use and inflation adjust
#   #                to create the missing year's data
#
#   child_care_costs %>%
#     filter(year == !!year_inflate) %>%
#     ff_inflation_adjust(., wages_col = estimate, year_adjust = year_create, error = FALSE) %>%
#     mutate(year = !!year_create) %>%
#     select(-estimate) %>%
#     rename(estimate = estimate_adj)
# }
#
```

```
# # calculate 2008 to 2010 using 2007 rates
# eight_ten <- data.frame()
# for (i in 2008:2010) {
#   eight_ten <- inf_adj_child_care(child, i, 2007) %>%
#     bind_rows(eight_ten, .)
# }
#
# # calculate 2011 to 2014 using 2015 rates
# eleven_fourteen <- data.frame()
# for (i in 2011:2014) {
#   eleven_fourteen <- inf_adj_child_care(child, i, 2015) %>%
#     bind_rows(eleven_fourteen, .)
# }
#
# # combine these years with all other years
# child <- child %>%
#   bind_rows(., list(eight_ten, eleven_fourteen)) %>%
#   arrange(year, start_age, county)
#
# ### next: create weighted average for NC
#
# # import county population percentages
# county_pop <- read_csv('county_perc_pop.csv') %>%
#   mutate(countyname = str_replace_all(countyname, ' County', '')) %>%
#   select(-pop)
#
# # bind county percentages
# # we do not have population percentage data for 2018
# child <- child %>%
#   left_join(., county_pop, by = c('year', 'county' = 'countyname')) %>%
#   # 2018 will have NAs in percentage, we can drop this year
#   drop_na() %>%
#   # create each county's weighted child care costs
#   mutate(weighted_costs = estimate * perc_pop) %>%
#   # sum these weighted costs by year to get state amounts
#   group_by(year, start_age) %>%
#   summarize(estimate = as.integer( sum(weighted_costs) )) %>%
#   mutate(county = 'NC') %>%
#   ungroup() %>%
#   # bind NC costs to county costs
#   bind_rows(child, .) %>%
#   # convert monthly amounts to year
#   mutate(estimate = estimate * 12)
#
# # save as R object because csvfile will not load properly
# #saveRDS(child, 'cleaned_data/child_care.Rda')
```