Process Data

Chris Hoover et al

4/23/2021

Generate population seeds

The next step is to generate the household and person seed dataframes. These "seeds" contain individual level characteristics of the target population that will be used in iterative proportional fitting along with target population characteristics to generate the population. To generate seeds, we need to incorporate our desired household and person characteristics and the survey weights to correct for over/under representation in the PUMS sample. We'll start with the household seed data, and the first step is to determine which of our chosen variables are at the household level

```
# Determine which variables are household level
SF_vars <- colnames(SF_pums)
hh_vars <- SF_vars[which(SF_vars %in% (pums_vars %>% filter(level == "housing") %>% pull(var_code)))]

hh_vars
```

```
## [1] "WGTP"   "HINCP"  "NP"     "ADJINC" "HHT"
```

We know that the "WGTP" variable is used for weighting, so our only household level variables are income and household size (number of persons, NP). We'll first combine the household income and inflation adjustment variables into a single household income variable. We also want to avoid stratifying the population too much, else we'll end up with unstable proportions to generate our population, so we'll condense the household income variable into categories as well. We'll use fairly arbitrary cutoffs of <$50k, $50k-$100k, and >$100k, but you could also use e.g. local poverty rates to better inform this categorization for your specific area. In this case, you might also want to check the distribution of household sizes (NP variable) and consider adding an upper cutoff (e.g. group all households with >=7 individuals). Lastly, we'll get our proportions for the seed dataset. For now, we won't include anyone living in group quarters, so we'll remove anyone with the "GQ" tag in their serial number

```
SF_hh_seed <- SF_pums %>%
  # Ignore individuals in group quarters for now
  filter(!grepl("GQ", SERIALNO)) %>%
  # Restrict to our household variables
  dplyr::select(c("SERIALNO", "SPORDER", "WGTP", "PWGTP", "PUMA", all_of(hh_vars))) %>%
  # Create household income adjusted for inflation variable and then categorize
  mutate(
    HHINCADJ  = HINCP*as.numeric(ADJINC),
    HHINC_CAT = as.factor(case_when(HHINCADJ < 50000 ~ 1,
                                    HHINCADJ >= 50000 & HHINCADJ < 100000 ~ 2,
                                    HHINCADJ >=100000 ~ 3)),
    NP        = if_else(NP >= 7, 7, NP), # any households with >7 people grouped together to match acs
    HHT       = if_else(HHT %in% c("1", "2", "3"), 1, 2)     # Family for 1,2,3 ; GQ/other/Non-family f
    ) %>%
  # Only one observation per household
  group_by(SERIALNO) %>%
```

```
  summarise(across(.cols = everything(),
                   .fns = first)) %>%
  # Cleanup
  dplyr::select(SERIALNO, PUMA, HHINC_CAT, HHT, NP, WGTP) %>%
  rename("hhincome" = HHINC_CAT,
         "hhsize" = NP,
         "hhtype" = HHT,
         "weight" = WGTP)



head(SF_hh_seed)
```

```
## # A tibble: 6 x 6
##   SERIALNO      PUMA  hhincome hhtype hhsize weight
##   <chr>         <chr> <fct>     <dbl>  <dbl>  <dbl>
## 1 2015000000528 07503 2             2      1     24
## 2 2015000000665 07503 3             1      3     12
## 3 2015000001345 07507 3             1      6     16
## 4 2015000001590 07507 2             2      1     41
## 5 2015000003135 07501 1             2      1      0
## 6 2015000004680 07505 3             2      1     14
```

Now we'll go through the same process to get the person seeds data frame. Similar to household income and size variables, any variables that are continuous, have many levels, or have levels with small frequencies will likely need to be condensed in order to produce stable weights for the iterative proportional fitting algorithm. Below we condense age into decadal increments, but also maintain the school grade variable, `SCHG`, in order to place school-aged children into classes.

```
# Determine which variables are person level as those leftover from household vars
p_vars <- SF_vars[which(!SF_vars %in% hh_vars)]

p_vars
```

```
##  [1] "SERIALNO" "SPORDER"  "PWGTP"    "AGEP"     "PUMA"     "ST"
##  [7] "SCHG"     "SEX"      "HISP"     "OCCP"     "RAC1P"
```

```
SF_p_seed <- SF_pums %>%
  # Ignore individuals in group quarters for now
  filter(!grepl("GQ", SERIALNO)) %>%
  # Restrict to our person variables
  dplyr::select(unique(c("SERIALNO", "SPORDER", "WGTP", "PWGTP", "PUMA", all_of(p_vars)))) %>%
  mutate(
    # Condense age into categories
    age_cat  = cut(AGEP, breaks = c(0,10,20,30,40,50,60,70,80,Inf), labels = FALSE, right = FALSE, inclu
    # Condense American Indian and Alaska Native categories to match acs reporting of aggregate totals
    race = if_else(RAC1P %in% c("3", "4", "5"), "3", RAC1P),
    # make hispanic category numeric
    hispanic = if_else(HISP == "01", 0, 1)
    ) %>%
  # Cleanup
  dplyr::select(SERIALNO, PUMA, SCHG, SEX, OCCP, age_cat, race, hispanic, PWGTP) %>%
  rename("grade"      = SCHG,
         "sex"        = SEX,
         "occupation" = OCCP,
```

```
         "weight"     = PWGTP) %>%
  # Merge with lookup table which relates acs and occp codes then add code 9 for unemployed/retired/in
  left_join(acs_occp_lookup %>% dplyr::select(Code, occ_group),
            by = c("occupation" = "Code")) %>%
  mutate(occ_group = if_else(occupation == "0009", 99, occ_group)) %>%
  filter(occ_group != 55) # Leave out military for now due to lack of target data

head(SF_p_seed)
```

```
## # A tibble: 6 x 10
##   SERIALNO  PUMA  grade sex   occupation age_cat race  hispanic weight occ_group
##   <chr>     <chr> <chr> <chr> <chr>        <int> <chr>    <dbl>  <dbl>     <dbl>
## 1 20150000~ 07507 bb    1     4700             5 6            0     16        41
## 2 20150000~ 07507 bb    2     0009             8 6            0     20        99
## 3 20150000~ 07507 bb    1     4220             5 6            0     17        37
## 4 20150000~ 07507 bb    2     3602             5 6            0     19        31
## 5 20150000~ 07507 bb    1     4020             3 6            0     19        35
## 6 20150000~ 07507 bb    1     0009             2 6            0     17        99
```

```
# household size -----------------------
acs_hhsize_tgt <- acs_hhsize_type %>%
  filter(county_fips == fips_use) %>%
  mutate(hhsize = parse_number(variable)) %>%
  group_by(GEOID, hhsize) %>%
  summarise(estimate = sum(estimate)) %>%
  ungroup() %>%
  pivot_wider(names_from = hhsize, values_from = estimate) %>%
  rename("geo_tract" = GEOID)
```

```
## `summarise()` has grouped output by 'GEOID'. You can override using the `.groups` argument.
```

```
# household type -----------------------
acs_hhtype_tgt <- acs_hhsize_type %>%
  filter(county_fips == fips_use) %>%
  mutate(hhtype = if_else(grepl("fam", variable), 1, 2)) %>%
  group_by(GEOID, hhtype) %>%
  summarise(estimate = sum(estimate)) %>%
  ungroup() %>%
  pivot_wider(names_from = hhtype, values_from = estimate) %>%
  rename("geo_tract" = GEOID)
```

```
## `summarise()` has grouped output by 'GEOID'. You can override using the `.groups` argument.
```

```
# household income ---------------
acs_hhincome_tgt <- acs_hh_income %>%
  filter(county_fips == fips_use) %>%
  left_join(acs_hhincome_lookup, by = c("variable" = "name")) %>%
  filter(!is.na(label)) %>%
  # match income groups to seed data
  mutate(hhincome = as.factor(case_when(income_max < 50000 ~ 1,
                                        income_min >= 50000 & income_max < 100000 ~2,
                                        income_min >= 100000 ~3))) %>%
  group_by(GEOID, hhincome) %>%
  summarise(estimate = sum(estimate)) %>%
  ungroup() %>%
  pivot_wider(names_from = hhincome, values_from = estimate) %>%
```

```
  rename("geo_tract" = GEOID)
```

```
## `summarise()` has grouped output by 'GEOID'. You can override using the `.groups` argument.
```

```
# Place household targets into list
SF_hh_tgt          <- list()
SF_hh_tgt$hhincome <- acs_hhincome_tgt
SF_hh_tgt$hhtype   <- acs_hhtype_tgt
SF_hh_tgt$hhsize   <- acs_hhsize_tgt
```

```
# Age -------------
acs_age_tgt <- acs_age_sex %>%
  filter(county_fips == fips_use) %>%
  # merge with acs_sex_by_age lookup table to get age categories and sex associated with each variable
  left_join(acs_sex_by_age_lookup, by = c("variable" = "name")) %>%
  filter(!variable %in% c("B01001_001", "B01001_002","B01001_026")) %>%  # Remove total aggregate, aggr
  # match decile age groups to seed data
  mutate(age_cat = case_when(age_max < 10 ~ 1,
                             age_min >= 10 & age_max < 20 ~2,
                             age_min >= 20 & age_max < 30 ~3,
                             age_min >= 30 & age_max < 40 ~4,
                             age_min >= 40 & age_max < 50 ~5,
                             age_min >= 50 & age_max < 60 ~6,
                             age_min >= 60 & age_max < 70 ~7,
                             age_min >= 70 & age_max < 80 ~8,
                             age_min >= 80 ~9)) %>%
  group_by(GEOID,age_cat) %>%
  summarise(estimate = sum(estimate)) %>%
  ungroup() %>%
  pivot_wider(names_from = age_cat, values_from = estimate) %>%
  rename("geo_tract" = GEOID)
```

```
## `summarise()` has grouped output by 'GEOID'. You can override using the `.groups` argument.
```

```
# Sex ---------------------
acs_sex_tgt <- acs_age_sex %>%
  filter(county_fips == fips_use) %>%
  # merge with acs_sex_by_age lookup table to get age categories and sex associated with each variable
  left_join(acs_sex_by_age, by = c("variable" = "name")) %>%
  filter(!variable %in% c("B01001_001", "B01001_002","B01001_026")) %>%  # Remove total aggregate, aggr
  group_by(GEOID,sex_num) %>%
  summarise(estimate = sum(estimate)) %>%
  ungroup() %>%
  pivot_wider(names_from = sex_num, values_from = estimate) %>%
  rename("geo_tract" = GEOID)
```

```
## `summarise()` has grouped output by 'GEOID'. You can override using the `.groups` argument.
```

```
# Race ---------------------
acs_race_tgt <- acs_race %>%
  filter(county_fips == fips_use) %>%
  # merge with lookup table to get race categories and binary ethnicity associated with each variable
  left_join(acs_race_eth_lookup, by = c("variable" = "name")) %>%
  filter(!variable %in% c("B03002_001", "B03002_002", "B03002_012")) %>% # Remove aggregate totals vari
  # recode race variable to match pums seed data
  mutate(race = case_when(grepl("White", race) ~ 1,
```

```
                          grepl("Black", race) ~ 2,
                          grepl("American Indian", race) ~ 3,
                          grepl("Asian", race) ~ 6,
                          grepl("Native Hawaiian", race) ~ 7,
                          grepl("Some other", race) ~ 8,
                          grepl("Two or more", race) ~ 9)) %>%
  group_by(GEOID,race) %>%
  summarise(estimate = sum(estimate)) %>%
  ungroup() %>%
  pivot_wider(names_from = race, values_from = estimate) %>%
  rename("geo_tract" = GEOID)
```

```
## `summarise()` has grouped output by 'GEOID'. You can override using the `.groups` argument.
```

```
# Ethnicity ----------------------
acs_eth_tgt <- acs_race %>%
  filter(county_fips == fips_use) %>%
  # merge with lookup table to get race categories and binary ethnicity associated with each variable
  left_join(acs_race_eth_lookup, by = c("variable" = "name")) %>%
  filter(!variable %in% c("B03002_001", "B03002_002", "B03002_012")) %>% # Remove aggregate totals vari
  group_by(GEOID,hispanic) %>%
  summarise(estimate = sum(estimate)) %>%
  ungroup() %>%
  pivot_wider(names_from = hispanic, values_from = estimate) %>%
  rename("geo_tract" = GEOID)
```

```
## `summarise()` has grouped output by 'GEOID'. You can override using the `.groups` argument.
```

```
# School grade -----------------
acs_grade_tgt <- acs_grade %>%
  filter(county_fips == fips_use) %>%
  left_join(acs_grade_lookup, by = c("variable" = "name")) %>%
  filter(!is.na(grade)) %>%  # filter out aggregate totals
  group_by(GEOID, grade) %>%
  summarise(estimate = sum(estimate)) %>%
  ungroup() %>%
  pivot_wider(names_from = grade, values_from = estimate) %>%
  rename("geo_tract" = GEOID)
```

```
## `summarise()` has grouped output by 'GEOID'. You can override using the `.groups` argument.
```

```
# Occupation ---------------
# Filter out higher level aggregations of occupational classes to only get lowest level classes
lowest_acs_occp_vars <- acs_occp_lookup %>%
  filter(str_length(Code2) < 10, !is.na(acs_var)) %>%
  pull(acs_var)

acs_occup_tgt_init <- acs_occup %>%
  filter(county_fips == fips_use,
         variable %in% lowest_acs_occp_vars) %>%
  # Merge with lookup table in order to match with seed data
  left_join(acs_occp_lookup, by = c("variable" = "acs_var")) %>%
  group_by(GEOID, occ_group) %>%
  summarise(estimate = sum(estimate)) %>%
  ungroup() %>%
  rename("geo_tract" = GEOID)
```

```
## `summarise()` has grouped output by 'GEOID'. You can override using the `.groups` argument.
```

```
# No unemployed counts in acs data, so need to determine number of people not working by subtracting wo
pop_totals <- acs_sex_tgt %>% mutate(estimate = `1`+`2`) # Total population
wrk_totals <- acs_occup_tgt_init %>% group_by(geo_tract) %>% summarise(estimate = sum(estimate)) #Total

nwrk_pop <- pop_totals %>% left_join(wrk_totals, by = c("geo_tract")) %>% #Total non-working population
  mutate(occ_group = 99,
         estimate = estimate.x - estimate.y) %>%
  dplyr::select(geo_tract, occ_group, estimate)

#Check to make sure population classified into working groups is equal to total population
sum(acs_occup_tgt_init %>% group_by(geo_tract) %>% summarise(tpop = sum(estimate)) %>% left_join(pop_to
```

```
## Joining, by = "geo_tract"
```

```
## [1] FALSE
```

```
# Pivot wider for final

acs_occup_tgt <- rbind(acs_occup_tgt_init, nwrk_pop) %>%
  group_by(geo_tract, occ_group) %>%
  summarise(estimate = sum(estimate)) %>%
  ungroup() %>%
  pivot_wider(names_from = occ_group, values_from = estimate)
```

```
## `summarise()` has grouped output by 'geo_tract'. You can override using the `.groups` argument.
```

```
# Place person targets into list
SF_p_tgt             <- list()
SF_p_tgt$grade       <- acs_grade_tgt
SF_p_tgt$sex         <- acs_sex_tgt
SF_p_tgt$occ_group   <- acs_occup_tgt
SF_p_tgt$age_cat     <- acs_age_tgt
SF_p_tgt$race        <- acs_race_tgt
SF_p_tgt$hispanic    <- acs_eth_tgt
```

```
synth_cts <- unique(wrk_totals$geo_tract[which(wrk_totals$estimate > 0)])

save(list = c("SF_p_seed", "SF_hh_seed", "SF_p_tgt", "SF_hh_tgt", "synth_cts"),
     file = here::here("Tutorial/data/processed_data.Rdata"))
```