Robustness Checks

Sarah Eckhardt

2025-02-04

0. Introduction

SIPP is representative at the national level, not sub-nationally. Given the sample size, 2+ dimensional cross-tabs may not be robust. Ideally, we would generate cuts by:

1. urban urban/rural vs industry
2. urban/rural vs states
3. generation
4. urban/rural by generation

In order to ascertain whether if any of these cuts are feasible, we conduct a series of robustness checks.

Robustness checks:

a. Simple sample size checks:

As a general rule, sub-groups should have no fewer than 30 obs (un-weighted).

b. Estimate sampling variability:

Compute standard errors for key estimates within each sub-group using survey weights. Large standard errors indicate a lack of reliability.

c. Design Effects

Check the design effect to see if clustering or stratification increases variance. High DEFF (>2) suggests that a larger sample may be needed for stability.

d. Margin of Error

Calculate confidence intervals for subgroup estimates. Wide confidence intervals indicate non-robustness.

1. Set-up

```r
# remove dependencies
rm(list = ls())

# load packages
library(dplyr)
library(survey)
library(batman)
library(ggplot2)

# set user path
project_directories <- list(
  "sarah" = "/Users/sarah/Documents/GitHub/Retirement-Analysis-Urban-Rural",
  "jiaxinhe" = "/Users/jiaxinhe/Documents/projects/Retirement-Analysis-Urban-Rural"
)

current_user <- Sys.info()[["user"]]
if (!current_user %in% names(project_directories)) {
  stop("Root folder for current user is not defined.")
}


# set project paths
project_path = project_directories[[current_user]]
data_path = file.path(project_path, "Data")
output_path = file.path(project_path, "Output")


# Read in wrangled SIPP data (see 1. wrangle SIPP 2023.R)
load(paste(output_path, "SIPP_2023_WRANGLED.RData", sep = "/"))

  # convert access, participation, and matching to bools
sipp_2023 = sipp_2023 %>%
  mutate(ANY_RETIREMENT_ACCESS_bool = case_when(
          ANY_RETIREMENT_ACCESS == "Yes" ~ 1,
          ANY_RETIREMENT_ACCESS == "No" ~ 0
        ),
        PARTICIPATING_bool = case_when(
          PARTICIPATING == "Yes" ~ 1,
          PARTICIPATING == "No" ~ 0
        ),
        MATCHING_bool = case_when(
          MATCHING == "Yes" ~ 1,
          MATCHING == "No" ~ 0
        ))
```

2. Robustness checks for urban/rural by major industry group

```r

 metro_ind_access = sipp_2023 %>% count(METRO_STATUS, INDUSTRY_BROAD,
                                          ANY_RETIREMENT_ACCESS)  %>%
    filter(METRO_STATUS != "Not identified") %>% filter(!is.na(METRO_STATUS))

  # hist(metro_ind_access$n, main = "metro X industry X access", xlab = "n")

  # how many are under 30?
  count(metro_ind_access %>% filter(n<30))
```

2.a Sample size check

```
##    n
## 1 14
```

```
  # what do we exclude?
  metro_ind_access_nu30 = metro_ind_access %>% filter(n<30)
  unique(metro_ind_access_nu30$INDUSTRY_BROAD)
```

```
## [1] "Public Administration"
## [2] "Agriculture, Forestry, Fishing, and Hunting, and Mining"
## [3] "Arts, Entertainment, and Recreation, and Accommodation and Food Services"
## [4] "Finance and Insurance, and Real Estate and Rental and Leasing"
## [5] "Information"
## [6] "Other Services, Except Public Administration"
## [7] "Wholesale Trade"
```

```
  excl = unique(metro_ind_access_nu30$INDUSTRY_BROAD)

  # if we exclude unreliable industries, what does this leave us with?
  metro_ind_access_n30 = metro_ind_access %>%
    filter(!(INDUSTRY_BROAD %in% excl))

  unique(metro_ind_access_n30$INDUSTRY_BROAD)
```

```
## [1] "Construction"
## [2] "Educational Services, and Health Care and Social Assistance"
## [3] "Manufacturing"
## [4] "Retail Trade"
## [5] "Transportation and Warehousing, and Utilities"
## [6] "and Waste Management Services"
```

```
  incl =  unique(metro_ind_access_n30$INDUSTRY_BROAD)
```

2.b Sample size check without urban/rural slice Since half of the industries would have too small of a sample size when divided along metro/non-metro lines, we should conduct industry-level analysis without the geographical dimension.

```
ind_access = sipp_2023 %>% filter(METRO_STATUS != "Not identified") %>%
  filter(!is.na(METRO_STATUS)) %>%
  count(INDUSTRY_BROAD, ANY_RETIREMENT_ACCESS)

# how many are under 30?
count(ind_access %>% filter(n<30))
```

```
##   n
## 1 2
```

```r
# what do we exclude?
ind_access_nu30 = ind_access %>% filter(n<30)
unique(ind_access_nu30$INDUSTRY_BROAD)
```

```
## [1] "Public Administration"
```

```r
excl = unique(ind_access_nu30$INDUSTRY_BROAD)

# if we exclude unreliable industries, what does this leave us with?
ind_access_n30 = ind_access %>%
  filter(!(INDUSTRY_BROAD %in% excl))

unique(ind_access_n30$INDUSTRY_BROAD)
```

```
##  [1] "Agriculture, Forestry, Fishing, and Hunting, and Mining"
##  [2] "Arts, Entertainment, and Recreation, and Accommodation and Food Services"
##  [3] "Construction"
##  [4] "Educational Services, and Health Care and Social Assistance"
##  [5] "Finance and Insurance, and Real Estate and Rental and Leasing"
##  [6] "Information"
##  [7] "Manufacturing"
##  [8] "Other Services, Except Public Administration"
##  [9] "Retail Trade"
## [10] "Transportation and Warehousing, and Utilities"
## [11] "Wholesale Trade"
## [12] "and Waste Management Services"
```

```r
incl =  unique(ind_access_n30$INDUSTRY_BROAD)
```

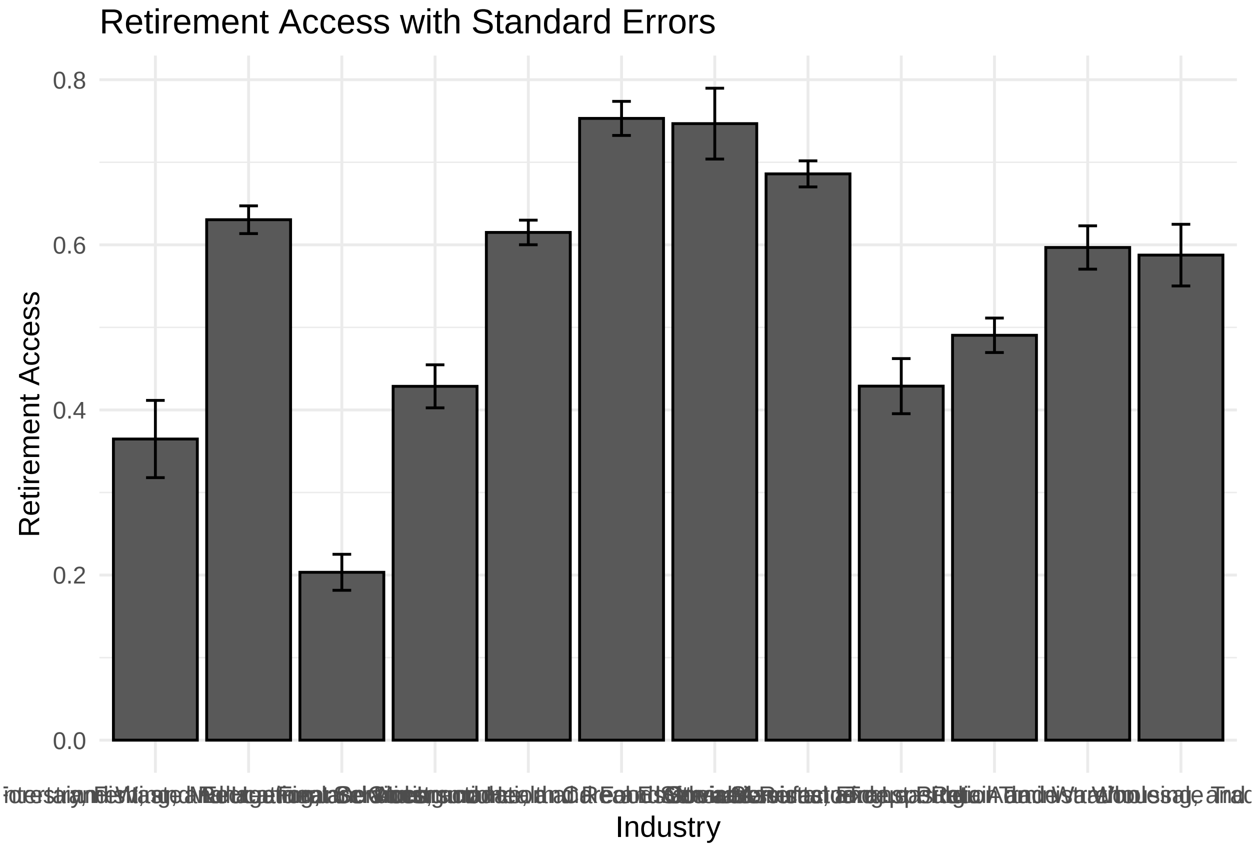
We exclude public administration based on this test

```r


# if we exclude these industries, is variance too large?
sipp_2023_excl_ind = sipp_2023 %>%
  filter(INDUSTRY_BROAD %in% incl) %>%
  filter(METRO_STATUS != "Not identified")

design <- svydesign(ids = ~0, weights = ~WPFINWGT, data = sipp_2023_excl_ind)

# mean, se of retirement access generally.
svymean(~ANY_RETIREMENT_ACCESS_bool, design)
```

2.c sampling variability

```
##                               mean     SE
## ANY_RETIREMENT_ACCESS_bool 0.58035 0.0067
```

```r
# means for sub groups
res = svyby(~ANY_RETIREMENT_ACCESS_bool, by = ~ INDUSTRY_BROAD, design, svymean)

# plot

ggplot(res, aes(x = INDUSTRY_BROAD, y = ANY_RETIREMENT_ACCESS_bool)) +
  geom_bar(stat = "identity", position = position_dodge(), color = "black") +
  geom_errorbar(aes(ymin = ANY_RETIREMENT_ACCESS_bool - se, ymax = ANY_RETIREMENT_ACCESS_bool + se),
                width = 0.2, position = position_dodge(0.9)) +
  labs(title = "Retirement Access with Standard Errors",
       x = "Industry",
       y = "Retirement Access",
       axis.ticks.x  = element_blank()) +
  theme_minimal()
```



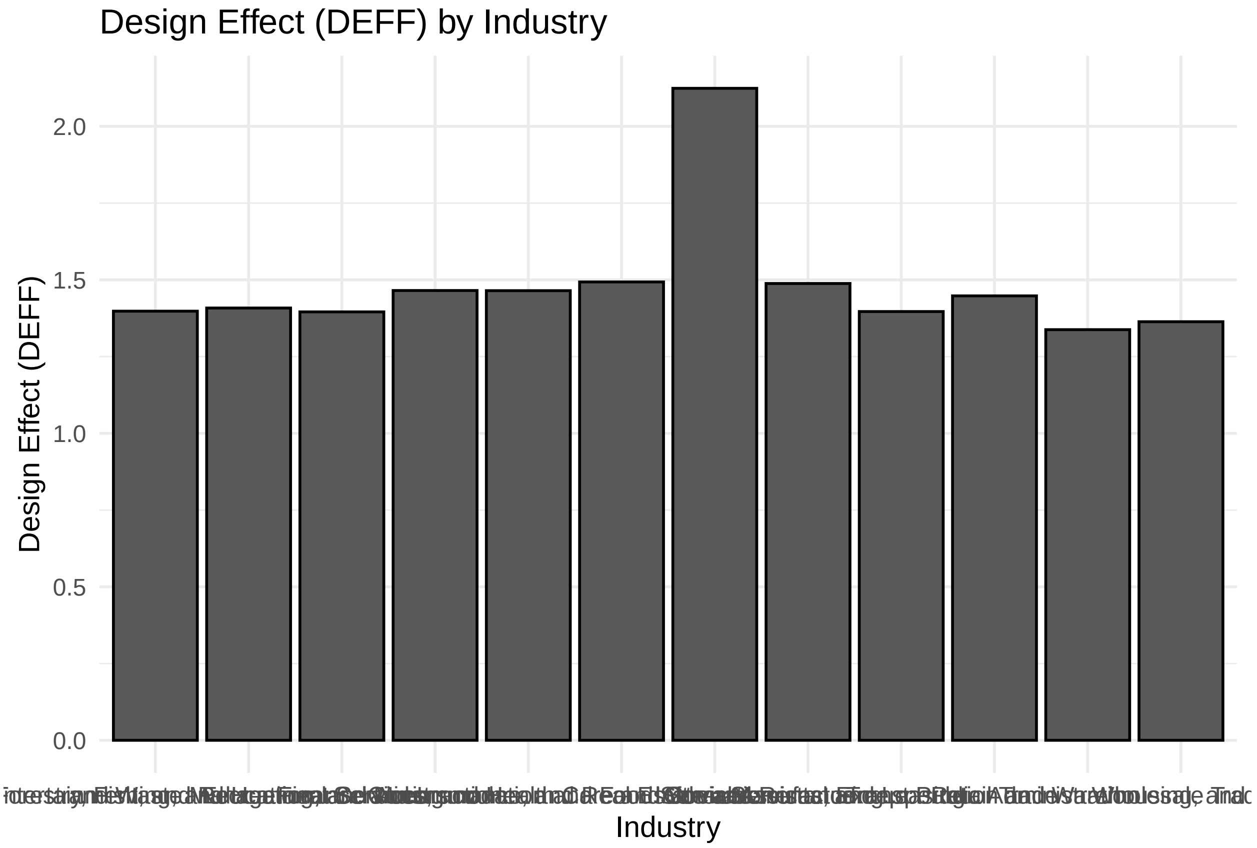
```r
# this looks ok, and fits w/ intuition
```

```r


# design effects --- does clustering or stratification increase variance?
# on the subsection of >30 n.
  # DEFF ~ 1 similar to random sample
  # > 1 higher variance w/ clustering
  # >2 high clustering, less precise
  # < 1 stratification improves precision

# consider: re-run svydesign construction to exclude Information

svymean(~ANY_RETIREMENT_ACCESS_bool, design, deff = TRUE)
```

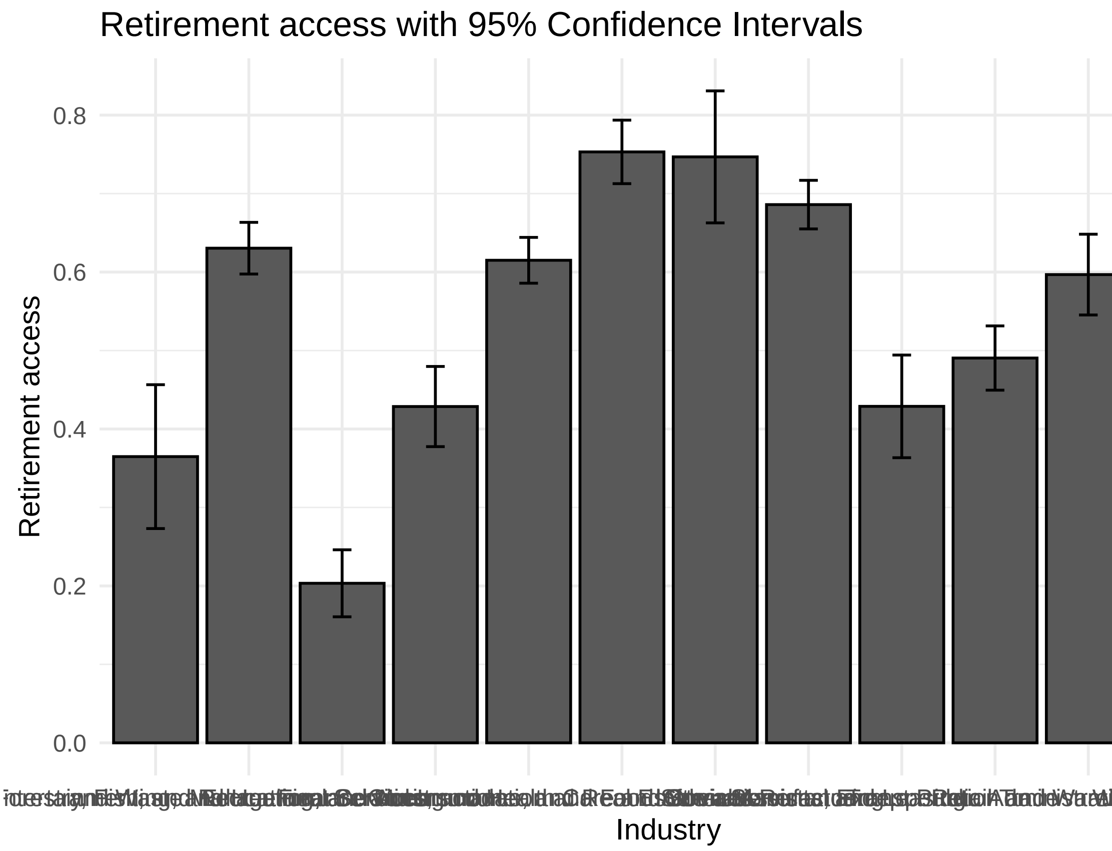
2.c design effects

```
##                                     mean         SE  DEff
## ANY_RETIREMENT_ACCESS_bool 0.5803544 0.0067184 1.461
```

```r
# all subgroups
res2 = svyby(~ANY_RETIREMENT_ACCESS_bool, by = ~INDUSTRY_BROAD, design, svymean, deff = TRUE)

# Exclude DEFF > 2, which is Information
excl <- c(excl, res2 %>% filter(res2$DEff.ANY_RETIREMENT_ACCESS_bool > 2) %>%
            .$INDUSTRY_BROAD)
incl <- setdiff(incl, excl)

# plot DEFF results
ggplot(res2, aes(x = INDUSTRY_BROAD, y = DEff.ANY_RETIREMENT_ACCESS_bool)) +
  geom_bar(stat = "identity", position = position_dodge(), color = "black") +
  labs(title = "Design Effect (DEFF) by Industry",
       x = "Industry",
       y = "Design Effect (DEFF)") +
  theme_minimal()
```



```
res2 = res2 %>%
  mutate(
    lower_CI = ANY_RETIREMENT_ACCESS_bool - 1.96 * se,
    upper_CI = ANY_RETIREMENT_ACCESS_bool + 1.96 * se
  )


ggplot(res2, aes(x = INDUSTRY_BROAD, y = ANY_RETIREMENT_ACCESS_bool)) +
  geom_bar(stat = "identity", position = position_dodge(), color = "black") +
  geom_errorbar(aes(ymin = lower_CI, ymax = upper_CI),
                width = 0.2, position = position_dodge(0.9)) +
  labs(title = "Retirement access with 95% Confidence Intervals",
       x = "Industry",
       y = "Retirement access") +
  theme_minimal()
```



2.d confidence intervals

```
# looks fine, no abnormal values
confint(svymean(~ANY_RETIREMENT_ACCESS_bool, design))
```

```
##                                2.5 %    97.5 %
## ANY_RETIREMENT_ACCESS_bool 0.5671866 0.5935223
```

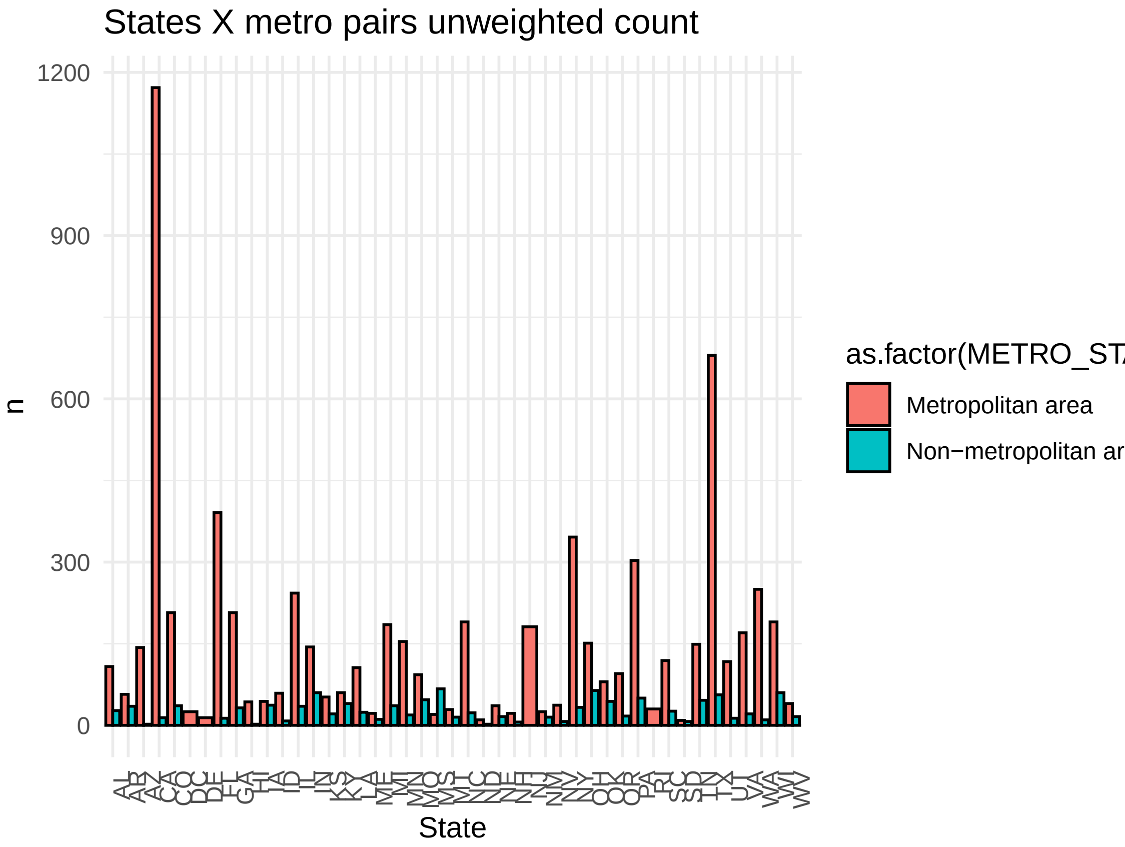
```r
# final list that passed robustness checks
print(incl)
```

```
##  [1] "Agriculture, Forestry, Fishing, and Hunting, and Mining"
##  [2] "Arts, Entertainment, and Recreation, and Accommodation and Food Services"
##  [3] "Construction"
##  [4] "Educational Services, and Health Care and Social Assistance"
##  [5] "Finance and Insurance, and Real Estate and Rental and Leasing"
##  [6] "Manufacturing"
##  [7] "Other Services, Except Public Administration"
##  [8] "Retail Trade"
##  [9] "Transportation and Warehousing, and Utilities"
## [10] "Wholesale Trade"
## [11] "and Waste Management Services"
```

```r
#output the final list
save(incl, file = file.path(output_path, "industry_robust.RData"))
```

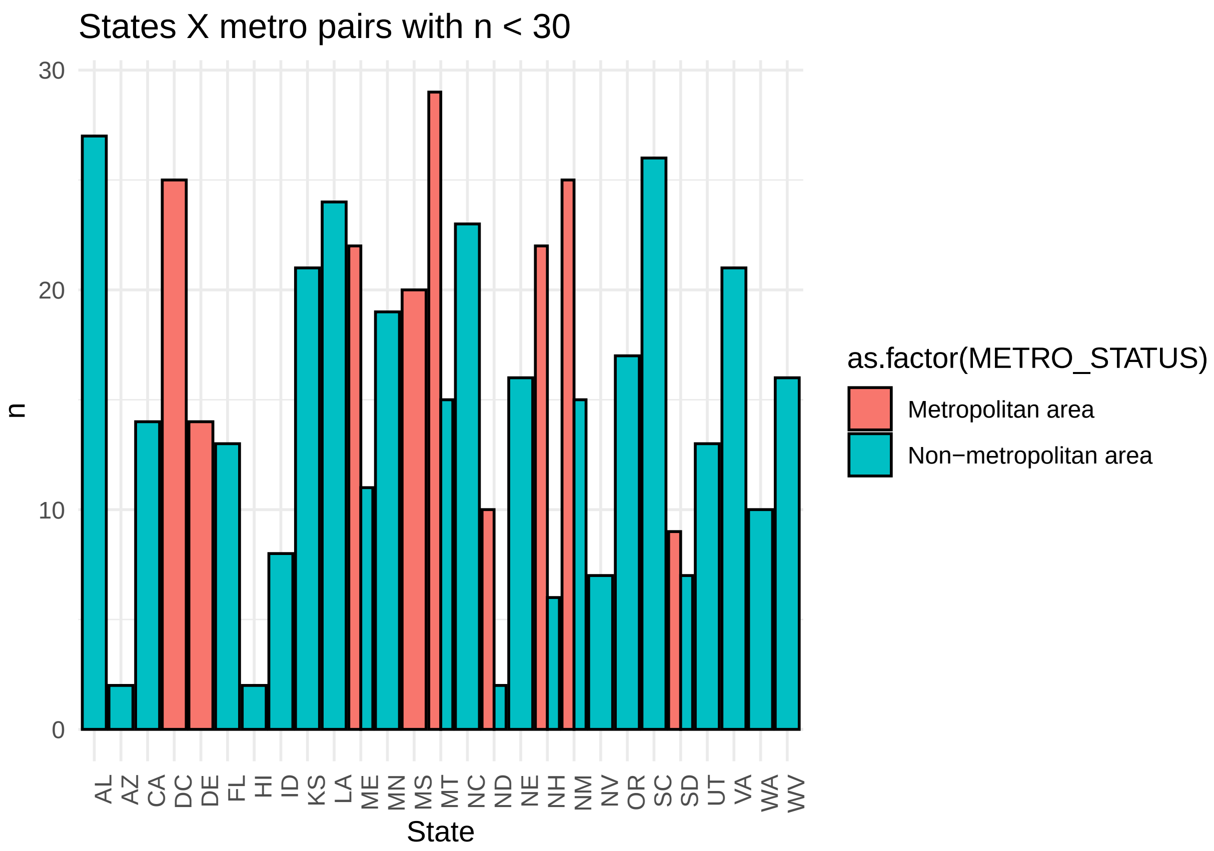
3. Robustness checks for States X Metro/non-metro

```r
# generate a crosswalk of state abbrs, state fips
state_fips_lookup <- data.frame(
  state_fips = c(1, 2, 4, 5, 6, 8, 9, 10, 11, 12, 13, 15, 16, 17, 18, 19,
                 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33,
                 34, 35, 36, 37, 38, 39, 40, 41, 42, 44, 45, 46, 47, 48,
                 49, 50, 51, 53, 54, 55, 56),
  state_abbr = c("AL", "AK", "AZ", "AR", "CA", "CO", "CT", "DE", "DC", "FL", "GA",
                 "HI", "ID", "IL", "IN", "IA", "KS", "KY", "LA", "ME", "MD", "MA",
                 "MI", "MN", "MS", "MO", "MT", "NE", "NV", "NH", "NJ", "NM", "NY",
                 "NC", "ND", "OH", "OK", "OR", "PA", "RI", "SC", "SD", "TN", "TX",
                 "UT", "VT", "VA", "WA", "WV", "WI", "WY")
)


sipp_2023 = sipp_2023 %>% left_join(state_fips_lookup, by = c("TST_INTV" = "state_fips"))


# basic state count.
sipp_2023 %>%
  count(state_abbr, METRO_STATUS) %>%
  filter(!is.na(METRO_STATUS)) %>% filter(METRO_STATUS != "Not identified") %>%
  ggplot(aes(x = state_abbr, y = n, fill = as.factor(METRO_STATUS))) +
    geom_bar(stat = "identity", position = "dodge", color = "black") + theme_minimal() +
  theme(axis.text.x = element_text(angle = 90, hjust = 1)) +
  labs(title = "States X metro pairs unweighted count",
       x = "State", y = "n")
```



3.a Sample size check

```
# how many are n<30 ?

sipp_2023 %>%
  count(state_abbr, METRO_STATUS) %>%
    filter(!is.na(METRO_STATUS)) %>% filter(METRO_STATUS != "Not identified") %>%
    filter(n<30) %>%
    ggplot(aes(x = state_abbr, y = n, fill = as.factor(METRO_STATUS))) +
    geom_bar(stat = "identity", position = "dodge", color = "black") + theme_minimal() +
  theme(axis.text.x = element_text(angle = 90, hjust = 1)) +
  labs(title = "States X metro pairs with n < 30",
       x = "State", y = "n")
```



```
# how many states does this impact?
st_met_n_under30 = sipp_2023 %>%
  count(state_abbr, METRO_STATUS) %>%
  filter(!is.na(METRO_STATUS)) %>%
  filter(METRO_STATUS != "Not identified") %>%
  filter(n<30)

length(unique(st_met_n_under30$state_abbr))
```

```
## [1] 27
```

```
# > 50% of the sample has states with either a metro or a non-metro sample of n < 30.
# this is not a viable sub group.
```

4. Robustness checks for SIPP vs SCF retirement account sizes

```
# Read in wrangled SCF data (see 1. wrangle SCF 2022.R)
load(paste(output_path, "SCF_2022_WRANGLED.RData", sep = "/"))

# calculated weighted average retirement account sizes for SCF and SIPP
avg_ret <- scf_2022 %>%
  mutate(across(RETIREMENT_ACCT_VAL, ~.x * WGT, .names="weighted_{.col}")) %>%
  summarise(across(weighted_RETIREMENT_ACCT_VAL, ~ sum(.x)/sum(WGT))) %>%
  rename(SCF_AVG_RETIREMENT_ACCOUNT = weighted_RETIREMENT_ACCT_VAL) %>%
  mutate(
    SIPP_AVG_RETIREMENT_ACCOUNT = sipp_2023 %>%
      mutate(across(TVAL_RET, ~.x * WPFINWGT, .names = "weighted_{.col}")) %>%
      summarise(across(weighted_TVAL_RET, ~sum(.x)/sum(WPFINWGT))) %>%
      .$weighted_TVAL_RET
  )

avg_ret # SIPP reports nearly $10,000 more retirment savings on average than SCF, not counting the infl
```

```
## # A tibble: 1 x 2
##   SCF_AVG_RETIREMENT_ACCOUNT SIPP_AVG_RETIREMENT_ACCOUNT
##                        <dbl>                       <dbl>
## 1                     99466.                     108963.
```

```
# two-sided t-test for weight-normalized account values
weighted_scf_ret <- scf_2022 %>%
  mutate(across(RETIREMENT_ACCT_VAL,~.x*WGT,.names="weighted_{.col}")) %>%
  mutate(across(weighted_RETIREMENT_ACCT_VAL, ~.x/sum(WGT))) %>%
  select("weighted_RETIREMENT_ACCT_VAL")

weighted_sipp_ret <- sipp_2023 %>%
  mutate(across(TVAL_RET,~.x*WPFINWGT,.names="weighted_{.col}")) %>%
  mutate(across(weighted_TVAL_RET, ~.x/sum(WPFINWGT))) %>%
  select("weighted_TVAL_RET")

t.test(weighted_scf_ret, weighted_sipp_ret)
```

```
##
##  Welch Two Sample t-test
##
## data:  weighted_scf_ret and weighted_sipp_ret
## t = -6.6998, df = 12724, p-value = 2.175e-11
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -4.530466 -2.479557
## sample estimates:
## mean of x mean of y
##  9.483789 12.988801
```

```
# the p-value is close to 0 and the 95% CI is [-5.23,-3.21], hence SIPP has larger mean account size th
```