Homework 1: SCF

Name

2025-11-08

Contents

1 Weights, Implications, and Replicate weights **1**

2 Application to population groups (can use packages!): **6**

1 Weights, Implications, and Replicate weights

(a) Step by Step calculation of mean household wealth (networth) and standard errors by hand (i.e. without survey package in R):

```
# (i) Unweighted classic
net <- scf$networth
valid <- !is.na(net)
n_obs <- sum(valid)
mean_unw <- mean(net, na.rm = TRUE)
se_unw <- sd(net, na.rm = TRUE) / sqrt(n_obs)
cat(sprintf('Unweighted: mean = %g, se = %g\n', mean_unw, se_unw))
```

```
## Unweighted: mean = 1.99564e+07, se = 726839
```

```
# (ii) Weighted naive
w <- scf$wgt
# weighted mean
mean_wtd <- sum(scf$networth * w, na.rm = TRUE) / sum(w[!is.na(scf$networth)])
# naive weighted SE: sqrt( sum( (w/sum(w))^2 * (x - mean_wtd)^2 ) )
den <- sum(w[!is.na(scf$networth)])
se_wtd <- sqrt( sum( (w[!is.na(scf$networth)]/den)^2 *
                ( (scf$networth[!is.na(scf$networth)] - mean_wtd)^2 ) ) )
cat(sprintf('Weighted (naive): mean = %g, se = %g\n', mean_wtd, se_wtd))
```

```
## Weighted (naive): mean = 1.05946e+06, se = 15011.7
```

```r
# (iii) Rubin (sampling weights for within)
M <- length(unique(scf$id_imp))
# compute theta (weighted mean) and within variance for each implicate
imp_stats <- scf %>%
  group_by(id_imp) %>%
  summarise(
    theta = sum(networth * w, na.rm = TRUE) / sum(w[!is.na(networth)]),
    # within-imputation variance
    var_within = {
      idx <- !is.na(networth)
      wloc <- w[idx]
      xloc <- networth[idx]
      denom <- sum(wloc)
      # compute weighted variance in the "probability weights" sense
      # variance of weighted mean approx: sum( (w/denom)^2 * (x - theta)^2 )
      theta_loc <- sum(xloc * wloc) / denom
      sum((wloc/denom)^2 * (xloc - theta_loc)^2)
    },
    .groups = 'drop'
  )

theta_bar <- mean(imp_stats$theta)
# average within-imputation variance
W <- mean(imp_stats$var_within)
# between-imputation variance
B <- (1/(M-1)) * sum( (imp_stats$theta - theta_bar)^2 )
Tvar <- W + (1 + 1/M) * B
se_rubin_sampling <- sqrt(Tvar)

#cat(sprintf('Implicates M = %d\n', M))
#cat(sprintf('Theta per implicate: %s\n',
#            paste(round(imp_stats$theta,3), collapse = ', ')))
cat(sprintf('Rubin: theta_bar = %g, SE = %g\n', theta_bar, se_rubin_sampling))
```

```
## Rubin: theta_bar = 1.95471e+07, SE = 932432
```

```r
# (iv) Replicate weights (implicate 1)
# Prepare replicate weights table: NA -> 0
rw <- scf_rw_2022_raw %>% mutate(across(everything(),
                                        ~ ifelse(is.na(.), 0, .)))

# detect replicate patterns: wt1b* and mm*
wt_cols <- grep('^wt1b', names(rw), value = TRUE)
mm_cols <- grep('^mm', names(rw), value = TRUE)
R <- min(length(wt_cols), length(mm_cols))
if(R < 1) stop('No replicate weight columns found')
```

```r
cat(sprintf('Detected %d replicate columns (using first %d)\n', R, R))
```

```
## Detected 999 replicate columns (using first 999)
```

```r
# build combined replicate weights wgt1...wgtR and keep yy1
rw_comb <- rw %>% arrange(yy1)
for(k in seq_len(R)){
  wcol <- paste0('wgt', k)
  rw_comb[[wcol]] <- rw_comb[[wt_cols[k]]] * rw_comb[[mm_cols[k]]]
}
rep_cols <- paste0('wgt', seq_len(R))
rw_comb <- rw_comb %>% select(yy1, all_of(rep_cols))

# merge replicate weights to summary file by yy1
scf_full <- left_join(scf, rw_comb, by = 'yy1')

# use first implicate
first_imp <- min(scf_full$id_imp, na.rm = TRUE)
scf_imp1 <- scf_full %>% filter(id_imp == first_imp)
if(nrow(scf_imp1) == 0) stop('No rows for first implicate')

# point estimate for implicate 1 (using main sampling weight wgt)
theta_imp1 <- sum(scf_imp1$networth * scf_imp1$wgt,
                  na.rm = TRUE) / sum(scf_imp1$wgt[!is.na(scf_imp1$networth)])

# compute replicate estimates theta_r for r in 1:R
theta_r <- numeric(R)
for(k in seq_len(R)){
  rwk <- scf_imp1[[paste0('wgt',k)]]
  denom <- sum(rwk[!is.na(scf_imp1$networth)])
  if(denom == 0) {
    theta_r[k] <- NA
  } else {
    theta_r[k] <- sum(scf_imp1$networth * rwk, na.rm = TRUE) / denom
  }
}
valid_theta_r <- !is.na(theta_r)
var_rep_imp1 <- mean( (theta_r[valid_theta_r] - theta_imp1)^2 )
se_rep_imp1 <- sqrt(var_rep_imp1)
cat(sprintf('Implicate %d: theta = %g, replicate-based SE = %g (R=%d used)\n',
            first_imp, theta_imp1, se_rep_imp1, sum(valid_theta_r)))
```

```
## Implicate 1: theta = 1.04182e+06, replicate-based SE = 78594.8 (R=998 used)
```

```r
# (v) Rubin + replicates (combined)

# For each implicate: compute theta_m and replicate-based variance
imp_theta <- scf_full %>% group_by(id_imp) %>%
  summarise(theta_m = sum(networth * wgt, na.rm = TRUE) / sum(
    wgt[!is.na(networth)]),
    .groups='drop')

replicate_variance_for_implicate <- function(df_imp, R, rep_prefix = 'wgt'){
  theta_hat <- sum(df_imp$networth * df_imp$wgt, na.rm = TRUE) / sum(
    df_imp$wgt[!is.na(df_imp$networth)])
  thetas_r <- numeric(R)
  for(k in seq_len(R)){
    rwk <- df_imp[[paste0(rep_prefix,k)]]
    denom <- sum(rwk[!is.na(df_imp$networth)])
    if(denom == 0) {
      thetas_r[k] <- NA
    } else {
      thetas_r[k] <- sum(df_imp$networth * rwk, na.rm = TRUE) / denom
    }
  }
  thetas_r <- thetas_r[!is.na(thetas_r)]
  if(length(thetas_r) == 0) return(NA_real_)
  mean((thetas_r - theta_hat)^2)
}

var_m_vec <- numeric(M)
for(m in seq_len(M)){
  dfm <- scf_full %>% filter(id_imp == m)
  var_m_vec[m] <- replicate_variance_for_implicate(dfm, R)
}

# average within-imputation variance from replicates
W_rep <- mean(var_m_vec, na.rm = TRUE)
B_imp <- (1/(M-1)) * sum( (imp_theta$theta_m - mean(imp_theta$theta_m))^2 )
Tvar_combined <- W_rep + (1 + 1/M) * B_imp
se_combined <- sqrt(Tvar_combined)

cat(sprintf('Combined: theta_bar = %g, SE = %g\n',
            mean(imp_theta$theta_m), se_combined))
```

```
## Combined: theta_bar = 1.05946e+06, SE = 97097.2
```

Method	Estimate (mean networth)	SE
(i) Unweighted classic	19,956,403	726,839.3

Method	Estimate (mean networth)	SE
(ii) Weighted naive	1,059,457	15,011.7
(iii) Rubin (sampling weights for within)	19,547,139	932,431.9
(iv) Replicate weights (implicate 1)	1,041,815	78,594.8
(v) Rubin + replicates (combined)	1,059,457	97,097.2

(b): Why do we use weights, why implicates, why replicate weights? How do the means and standard errors change between the steps and why?

- Sampling weights (`wgt`): adjust for unequal selection probabilities, nonresponse and post-stratification so that estimates generalize to the target population. Using weights changes the estimand from a sample-average to a population-average and typically reduces the influence of over-sampled subgroups (e.g., very wealthy households in SCF).
- Implicates (multiple imputation): the SCF provides M completed datasets to reflect uncertainty about imputed values (especially for wealth components). Compute your statistic in each implicate and combine with Rubin's rules: the overall point estimate is the average of implicate estimates; the total variance equals the average within-imputation variance plus (1 + 1/M) times the between-imputation variance. Ignoring implicates underestimates uncertainty.
- Replicate weights: capture sampling variance due to the complex survey design (strata, clustering, multi-stage selection). For each set of replicate weights recompute the estimator; the variability across replicate estimates (with the replication-specific scaling) gives a design-correct estimate of variance. Replicate-based SEs are generally larger than naive weighted SEs because they account for design effects.
- (i) Unweighted: Mean = sample average; SE = s / sqrt(n) (SRS formula). This treats the data as a simple random sample and typically overstates the influence of over-sampled groups (e.g., very wealthy households), so the unweighted mean is often larger than population-weighted estimates.
- (ii) Weighted naive: Mean = weighted population average using wgt; SE = naive weighted variance (treats weights as fixed). The point estimate usually moves (often down) relative to the unweighted mean because weights correct for unequal selection and post-stratification. The naive SE typically underestimates true sampling variability because it ignores the complex survey design (clustering/stratification) and any imputation uncertainty.
- (iii) Rubin (sampling-weights within): Point estimate = average of implicate-specific weighted means (close to the weighted mean if imputations are similar). SE = Rubin total variance combining average within-imputation variance and between-imputation variance. Compared with (ii) the SE increases if imputations differ, because Rubin's rules add between-imputation uncertainty to the within-imputation variance.
- (iv) Replicate weights (single implicate): Point estimate for the implicate uses the main weight (so similar to that implicate's weighted mean); SE = variance across replicate-weight estimates (design-based). This SE is typically larger than the naive weighted SE

because replicate-based variance captures the complex SCF sampling design rather than assuming independent observations.

- (v) Rubin + replicates (combined): Point estimate = average of implicate-specific replicate-based estimates (same estimand as (ii) aggregated over imputations). SE = average replicate-based within-imputation variance plus (1 + 1/M) times the between-imputation variance. This combined SE captures both design-based sampling variability and imputation uncertainty and is the most appropriate (and usually largest) SE among the steps.

(c): Now use the survey and mitools package to calculate the correct mean and standard error.

```
# Build a list of implicates
# each implicate is a dataframe with replicate weights merged
scf_list_full <- split(scf_full, factor(scf_full$id_imp))
scf_list_full <- lapply(scf_list_full, function(df) df %>% arrange(yy1))

# Prepare replicate weights matrix (rows by unique yy1)
# Use the combined replicate table we already built: rw_comb (yy1 + wgt1..wgtR)
repweights_mat <- rw_comb[ , -1]

# Create svrepdesign with the provided replicate weights and the imputation list
scf_design <- svrepdesign(weights = ~ wgt,
                          repweights = repweights_mat,
                          data = imputationList(scf_list_full),
                          scale = 1,
                          rscales = rep(1 / (R-1), R),
                          mse = FALSE,
                          type = "other",
                          combined.weights = TRUE)

# compute mean(networth) with survey functions across implicates
# then combine with mitools
svy_mean_list <- with(scf_design, svymean(~ networth))
mi_mean <- MIcombine(svy_mean_list)

cat(sprintf('Estimate = %g, SE = %g\n', coef(mi_mean), SE(mi_mean)))
```

```
## Estimate = 1.05946e+06, SE = 46959.6
```

2 Application to population groups (can use packages!):

(a) Calculate mean household wealth and standard error by gender of household head (hhsex) and by race (racecl5) (separately) and the weighted number of observations by group. Shortly explain what you find.

```r
# By hhsex
mi_hhsex_mean <- MIcombine(with(scf_design, svyby(~ networth,
                                                  ~ hhsex, svymean)))
#cat('\nMean(networth) by hhsex (survey+mitools):\n')
print(coef(mi_hhsex_mean))
```

```
##         1         2
## 1330436.5  358141.5
```

```r
#cat('SEs:\n')
print(SE(mi_hhsex_mean))
```

```
##       1       2
## 61388.5 35124.9
```

```r
# weighted population counts by hhsex
#use counts=5 per row so totals reflect population counts
mi_hhsex_pop <- MIcombine(with(scf_design, svyby(~ counts, ~ hhsex, svytotal)))
#cat('\nWeighted population counts (by hhsex):\n')
print(coef(mi_hhsex_pop))
```

```
##        1        2
## 94709757 36596633
```

Table below summarizes the results. There is a large gender wealth gap: male-headed households hold nearly four times more net worth on average. The standard errors are relatively small compared to the means, suggesting precise estimates. The population counts indicate that male-headed households are more numerous in the sample.

hhsex	Label	Mean_Networth	SE_Networth	Population_Count
1	Male	1330436	61388	94709757
2	Female	358142	35125	36596633

```r
# By racecl5
mi_race_mean <- MIcombine(with(scf_design, svyby(~ networth,
                                                 ~ racecl5, svymean)))
#cat('\nMean(networth) by racecl5 (survey+mitools):\n')
print(coef(mi_race_mean))
```

```
##         1         2         3         4         5
## 1361785.1  211511.7  227523.4 1810102.3  389473.8
```

```r
print(SE(mi_race_mean))
```

```
##         1         2         3         4         5
##  63203.01  39885.20  26517.11 319343.65  51448.57
```

```r
mi_race_pop <- MIcombine(with(scf_design, svyby(~ counts, ~ racecl5, svytotal)))
#cat('\nWeighted population counts (by racecl5):\n')
print(coef(mi_race_pop))
```

```
##        1        2        3        4        5
## 87725120 15118565 12284804  5181424 10996477
```

Table below summarizes the results by race. White and Asian households show much higher mean net worth than Black, Hispanic, or Other households in these SCF estimates. Asian households have the highest point estimate but also the largest SE (reflecting smaller sample size and more variability). Black and Hispanic households show substantially lower mean net worth with relatively smaller absolute SEs but large relative uncertainty compared with their point estimates. These differences reflect persistent wealth disparities and also larger uncertainty for smaller subgroups in the SCF public sample.

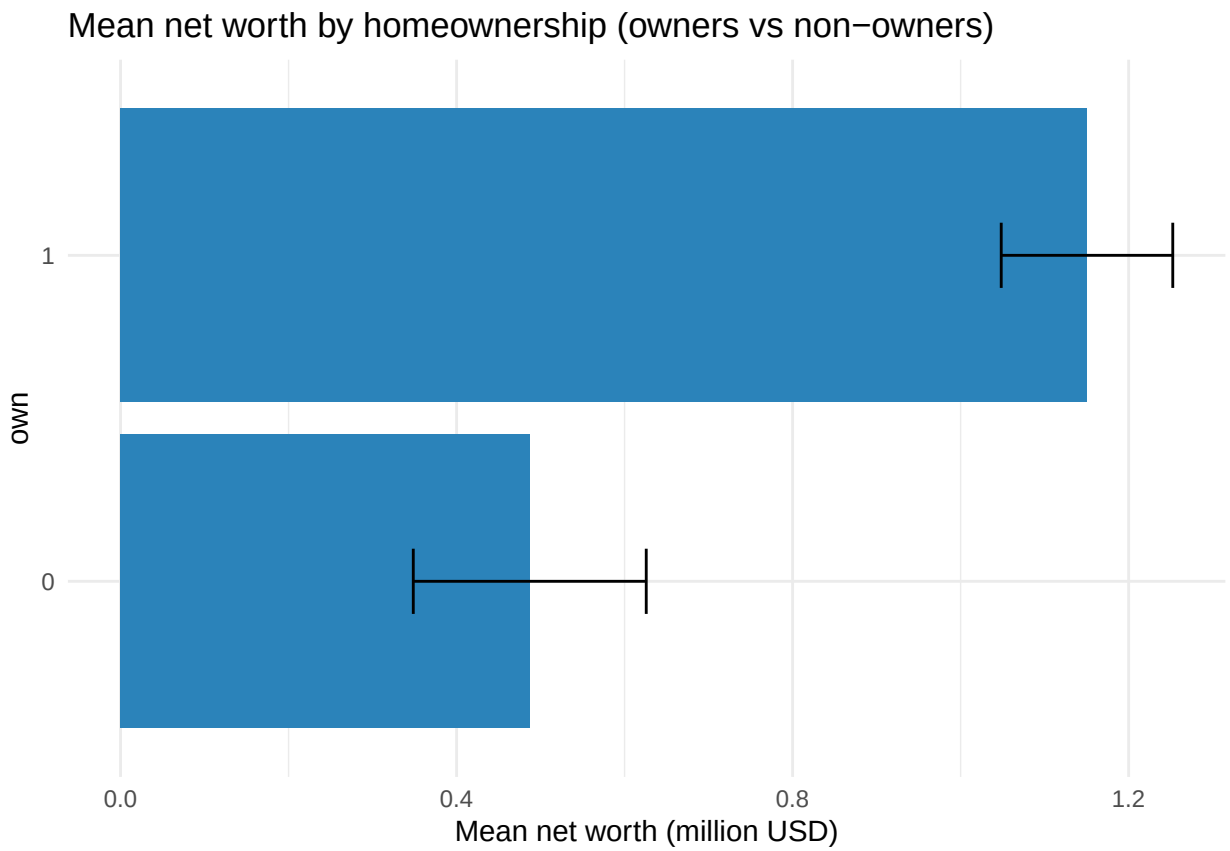
racecl5	Label	Mean_Networth	SE_Networth	Population_Count
1	White	1361785	63203	87725120
2	Black	211512	39885	15118565
3	Hispanic	227523	26517	12284804
4	Asian	1810102	319344	5181424
5	Other	389474	51449	10996477

(b) Choose another characteristic of households (which has not yet been used in class and makes sense in this context) and show mean household wealth and 95% confidence intervals in a bar chart by this characteristic and shortly explain what you find.

```r
# Compute mean(networth) and 95% CIs by a chosen household characteristic
# The script picks the first available variable from a candidate list

# Candidate grouping variables (try in this order)
candidates <- c('ownhome','homeown','own_primary','own', 'x30022', 'region',
                'age', 'agecl5', 'agecl', 'agegrp', 'educ', 'educyrs',
                'edcl', 'hhage')
# exclude variables already used
exclude <- c('hhsex','racecl5')

found_var <- NULL
for(var in candidates){
  if(var %in% names(scf) && !(var %in% exclude)){
```

```
    found_var <- var
    break
  }
}

if(is.null(found_var)){
  # fallback: pick the first non-NA categorical-like variable not excluded
  others <- setdiff(names(scf), c('networth','wgt','y1','yy1','id_imp',
                                  'wgt5','counts', exclude))
  # prefer factor/character variables
  for(var in others){
    if(is.character(scf[[var]]) || is.factor(scf[[var]])){
      found_var <- var; break
    }
  }
}

# Prepare replicate weights like in hw1_solutions.R
rw <- scf_rw_2022_raw %>% mutate(across(everything(), ~ ifelse(is.na(.), 0, .)))
wt_cols <- grep('^wt1b', names(rw), value = TRUE)
mm_cols <- grep('^mm', names(rw), value = TRUE)
R <- min(length(wt_cols), length(mm_cols))
if(R < 1) stop('No replicate weight columns found in replicate file')

rw_comb <- rw %>% arrange(yy1)
for(k in seq_len(R)){
  wcol <- paste0('wgt', k)
  rw_comb[[wcol]] <- rw_comb[[wt_cols[k]]] * rw_comb[[mm_cols[k]]]
}
rep_cols <- paste0('wgt', seq_len(R))
rw_comb <- rw_comb %>% select(yy1, all_of(rep_cols))

scf_full <- left_join(scf, rw_comb, by = 'yy1')

# If chosen var is numeric with many unique values, bin into quartiles
var_data <- scf_full[[found_var]]
group_var_name <- found_var
if(is.numeric(var_data) && length(unique(var_data[!is.na(var_data)])) > 10){
  # create 4 quantile bins
  scf_full[[paste0(found_var, '_grp')]] <-
    cut(var_data, breaks = unique(quantile(var_data,
                                           probs = seq(0,1,0.25),
                                           na.rm = TRUE)),
        include.lowest = TRUE)
  group_var_name <- paste0(found_var, '_grp')
}
```

```r
# build imputation list and replicate matrix
scf_list_full <- split(scf_full, factor(scf_full$id_imp))
scf_list_full <- lapply(scf_list_full, function(df) df %>% arrange(yy1))
repweights_mat <- rw_comb[ , -1]

scf_design <- svrepdesign(weights = ~ wgt,
                          repweights = repweights_mat,
                          data = imputationList(scf_list_full),
                          scale = 1,
                          rscales = rep(1 / (R-1), R),
                          mse = FALSE,
                          type = 'other',
                          combined.weights = TRUE)

# compute means by group and combine
formula_group <- as.formula(paste('~', group_var_name))
mi_group_mean <- MIcombine(with(scf_design, svyby(~ networth,
                                                  formula_group, svymean)))
est <- coef(mi_group_mean)
se  <- SE(mi_group_mean)

# build output data.frame
groups <- names(est)
df_out <- data.frame(group = groups, estimate = as.numeric(est),
                     se = as.numeric(se), stringsAsFactors = FALSE)
df_out$lower95 <- df_out$estimate - 1.96 * df_out$se
df_out$upper95 <- df_out$estimate + 1.96 * df_out$se

ggplot(df_out, aes(x = reorder(group, estimate), y = estimate/1e6)) +
  geom_col(fill = '#2b83ba') +
  geom_errorbar(aes(ymin = lower95/1e6, ymax = upper95/1e6), width = 0.2) +
  coord_flip() +
  labs(title = "Mean net worth by homeownership (owners vs non-owners)",
       x = found_var, y = 'Mean net worth (million USD)') +
  theme_minimal()
```



Owner households have a substantially higher mean net worth (~$1.15M) than non-owners (~$0.49M). The 95% CIs do not overlap, indicating a statistically meaningful difference in mean net worth between owners and non-owners in these SCF estimates. Note that homeowners both tend to have higher accumulated wealth and are oversampled in SCF designs; the reported SEs account for design and imputation uncertainty. Smaller subgroup sample sizes and the heavy-tailed wealth distribution mean these CI widths should be considered when comparing subgroups.

(c) Now choose another variable (should be inequality-related) that you want to compare by the characteristic in 2b. Again create a similar barchart.

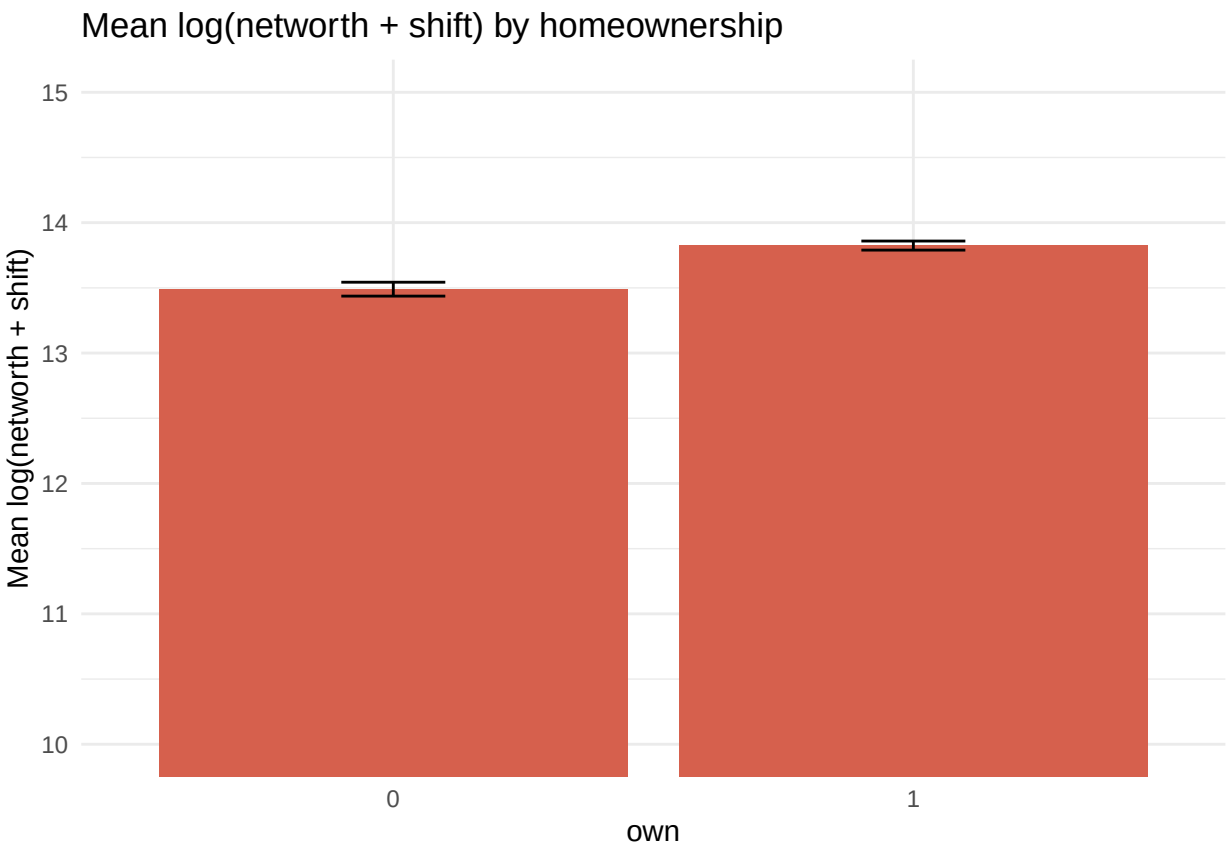
```
# Compute mean(log(networth)) by homeownership (own) using survey + mitools
# We'll compute log(networth) after shifting to positive values
min_nw <- min(scf$networth, na.rm = TRUE)
shift <- 0
if(min_nw <= 0) shift <- abs(min_nw) + 1
cat(sprintf('Shifting networth by %g to compute log(networth + shift)\n',
            shift))
```

```
## Shifting networth by 555501 to compute log(networth + shift)
```

```
scf <- scf %>% mutate(log_net = log(networth + shift))
```

```r
# Prepare replicate weights
rw <- scf_rw_2022_raw %>% mutate(across(everything(), ~ ifelse(is.na(.), 0, .)))
wt_cols <- grep('^wt1b', names(rw), value = TRUE)
mm_cols <- grep('^mm', names(rw), value = TRUE)
R <- min(length(wt_cols), length(mm_cols))
if(R < 1) stop('No replicate weight columns found')

rw_comb <- rw %>% arrange(yy1)
for(k in seq_len(R)){
  wcol <- paste0('wgt', k)
  rw_comb[[wcol]] <- rw_comb[[wt_cols[k]]] * rw_comb[[mm_cols[k]]]
}
rep_cols <- paste0('wgt', seq_len(R))
rw_comb <- rw_comb %>% select(yy1, all_of(rep_cols))

scf_full <- left_join(scf, rw_comb, by = 'yy1')

# Build imputation list
scf_list_full <- split(scf_full, factor(scf_full$id_imp))
scf_list_full <- lapply(scf_list_full, function(df) df %>% arrange(yy1))
repweights_mat <- rw_comb[ , -1]

scf_design <- svrepdesign(weights = ~ wgt,
                          repweights = repweights_mat,
                          data = imputationList(scf_list_full),
                          scale = 1,
                          rscales = rep(1 / (R-1), R),
                          mse = FALSE,
                          type = 'other',
                          combined.weights = TRUE)

# compute mean(log_net) by own
mi_log_mean <- MIcombine(with(scf_design, svyby(~ log_net, ~ own, svymean)))
est <- coef(mi_log_mean)
se <- SE(mi_log_mean)

groups <- names(est)
df_out <- data.frame(group = groups, estimate = as.numeric(est),
                     se = as.numeric(se), stringsAsFactors = FALSE)
df_out$lower95 <- df_out$estimate - 1.96 * df_out$se
df_out$upper95 <- df_out$estimate + 1.96 * df_out$se

# Plot mean log values (not exponentiated)
ggplot(df_out, aes(x = factor(group), y = estimate)) +
  geom_col(fill = '#d6604d') +
  geom_errorbar(aes(ymin = lower95, ymax = upper95), width = 0.2) +
  labs(title = 'Mean log(networth + shift) by homeownership',
```

```
      x = 'own', y = 'Mean log(networth + shift)') +
  # limit y-axis from 10 to 15
  coord_cartesian(ylim = c(10, 15)) +
  theme_minimal()
```



Owners have a higher mean log(networth + shift) than non-owners, consistent with the earlier level comparisons; the difference is meaningful and the confidence intervals are tight. Mean log differences compress extreme values and therefore give a sense of typical proportional differences rather than absolute-dollar gaps.