Problem Set 6

Due 11/10/2021

Following the material in survey weighting lecture, we will use the Census data from East Germany. In Canvas, download the survey ($n = 1000$, `svy_GDR.rds`) and the population frame ($N = 100{,}000$, `pop_GDR.rds`) it is drawn from. The poll is a biased sample of the population, so the goal is to make proper inferences about the population.

To recap, the survey has the following variables:

- `Y` is a synthetic binary outcome variable of interest (e.g. Yes to some public opinion question).
- `educ` is Education, which has three levels. "Abitur" are those who pass the final exam at the end of high school aiming for higher education. "Vocational" are vocational schools in the former GDR. The rest are coded as "No HighEd".
- `state` is the region/state in Germany.
- Other variables are other demographic variables and their binary versions.

The population frame is often not available in modern online surveys, but we will use it here to explore the accuracy of our methods.

- In this data, `D` is an indicator for Selection / Survey Inclusion. It is recorded in the frame. It is 1 if the item in the frame ended up in the survey, 0 otherwise.
- `ID` is the unique identifier to join the population and the survey.

```
svy <- read_rds(here("data/svy_GDR.rds"))
pop <- read_rds(here("data/pop_GDR.rds"))
```

Questions

1. Post-stratification weights: Compute post-stratification weights for the poll that will correct for the imbalance in the *joint* distribution of state and education in the survey data. Use the simple ratio estimator shown in lecture.

   Create a table with unweighted and weighted estimates of `Y` in each state (with standard errors), as well as a final column of the population mean of `Y`, and assess the accuracy of the survey estimates. That is, your table should look like:

   ```
   # weights
   pop_strat <- count(pop, state, educ, name = "N")
   svy_strat <- count(svy, state, educ)

   strat_weights <- svy_strat %>%
     left_join(pop_strat, by = c("state", "educ")) %>%
     mutate(w_strat = (N / sum(N)) / (n / sum(n)))

   svy_pstrat <- svy %>%
     left_join(strat_weights, by = c("state", "educ"))
   ```

```r
# estimates
Ys_raw <- svy_pstrat %>%
  group_by(state) %>%
  summarize(est = mean(Y), se = sqrt(sd(Y)/n()))

Ys_wt  <- svy_pstrat %>%
  group_by(state) %>%
  summarize(est = weighted.mean(Y, w_strat),
            se = sqrt(sd(Y)/ (n() / (sd(w_strat) + 1))))
Ys_pop <- pop %>%
  group_by(state) %>%
  summarize(Y = mean(Y))


# Combine in table
val_tbl <- left_join(Ys_raw, Ys_wt, suffix = c("_raw", "_wgt"), by = "state") %>%
  left_join(Ys_pop, by = "state")

# Format in nice table
val_tbl %>%
  kbl(col.names = c("State", "Est.", "SE", "Est.", "SE", "Pop."),
      digits = 2,
      booktabs = TRUE,
      centering = TRUE,
      position = "!h",
      caption = "Answer for Question 1",
      format = "latex") %>%
  add_header_above(c(" " = 1, "Unweighted" = 2, "Weighted" = 2, " "= 1)) %>%
  kable_styling(position = "center")
```

Table 1: Answer for Question 1

	Unweighted		Weighted		
State	Est.	SE	Est.	SE	Pop.
Brandenburg	0.17	0.04	0.19	0.05	0.24
Mecklenburg-Vorpommern	0.27	0.07	0.30	0.08	0.25
Saxony	0.21	0.04	0.22	0.04	0.24
Saxony-Anhalt	0.21	0.04	0.22	0.05	0.25
Thuringia	0.21	0.05	0.23	0.05	0.24

2. Estimate a propensity score for selection `D` from the frame. Try two different modeling specifications, one that is simple and one that is complex. You can use anything — logit, `CBPS` (Imai and Ratkovic 2012), random forest, etc.. — as long as the model produces estimates that are bounded between 0 and 1.

```r
fit_ps1 <- glm(D ~ educ, pop, family = binomial)
fit_ps2 <- glm(D ~ educ + age + marital + state, pop, family = binomial)

pop_pscore <- pop %>%
```

```
  mutate(w_ps1 = 1 / predict(fit_ps1, type = "response"),
         w_ps2 = 1 / predict(fit_ps2, type = "response"))

svy_pscore <- pop_pscore %>%
  filter(D == 1) %>%
  mutate(w_ps1 = w_ps1 / mean(w_ps1),
         w_ps2 = w_ps2 / mean(w_ps2))
```

Summarize your two propensity scores' and unweighted estimates' in performance in predicting `Y` in a table. In columns, report: (1) the prediction error (sample bias), (2) the Kish design effect (reduction in effective sample size due to increased variance) for each of the three models.

```
Ypop <- mean(pop$Y)

yhat_raw <- mean(svy_pscore$Y)
deff_raw <- 1

# bias
yhat_ps1 <- weighted.mean(svy_pscore$Y, svy_pscore$w_ps1)
yhat_ps2 <- weighted.mean(svy_pscore$Y, svy_pscore$w_ps2)

# D_eff
deff_ps1 <- 1 + sd(svy_pscore$w_ps1)
deff_ps2 <- 1 + sd(svy_pscore$w_ps2)

# summarize
tribble(
  ~Method,   ~Bias,           ~Deff,
  "Raw",     yhat_raw - Ypop, deff_raw,
  "Simple",  yhat_ps1 - Ypop, deff_ps1,
  "Complex", yhat_ps2 - Ypop, deff_ps2
) %>%
  kbl(format = "latex",
      digits = 2,
      caption = "Example Answer for Question 2",
      booktabs = TRUE, position ="h")  %>%
  kable_styling(position = "center")
```

Table 2: Example Answer for Question 2

Method	Bias	Deff
Raw	-0.04	1.00
Simple	-0.02	1.23
Complex	-0.03	1.34

3. In practice, a population frame is often unavailable, so estimating true propensity scores is out of the question. Calibration methods like entropy balancing (Hainmueller 2012) avoids estimating a propensity score and instead treats weighting as a constrained optimization problem with user-specified constraints. That is, it searches for a weight vector that minimizes a loss function (like variance of weights) while constraining the weighted proportion of covariates to equal the

user-specified target marginal distribution (like the education breakdown).

For example, `ebal` minimizes the entropy of the weights (roughly $\sum_i w_i \log(w_i)$) . This method derives from iterated proportional fitting (IPF) in the survey weighting literature, which has also been shown to minimize the entropy of weights (Ireland and Kullback 1968). IPF is often called "raking".

Compute rake weights that weights your survey to the education distribution and the state distribution in the population. You can use the `survey` package (Or other implementations, like `autumn`, by the UCLA political science department). As is often the case in practical examples, pretend you know nothing else about the population: you do not know the joint distribution of education and states, and you certainly do not have a individual-level dataset of the population.

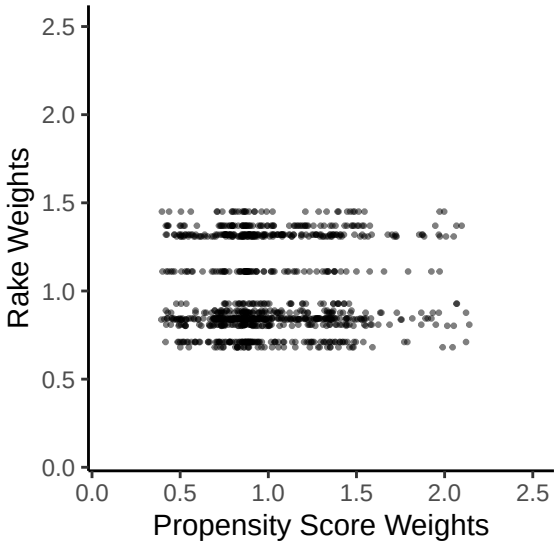
After verifying that your rake weighted marginals do in fact match the population marginals, present a clean scatterplot that compares your rake weights to your propensity score weights in the previous problem.

```
library(survey)

# setup
edu_margins   <- count(pop, educ, name = "Freq")
state_margins <- count(pop, state, name = "Freq")
svyd <- svydesign(id = ~1, data = svy)
```

```
## Warning in svydesign.default(id = ~1, data = svy): No weights or probabilities
## supplied, assuming equal probability
```

```
# rake weighting
svyd_rake <- rake(
  svyd,
  sample.margins = list(~ educ, ~ state),
  population.margins = list(edu_margins, state_margins))

# standardize
svy_rake <- svy_pscore %>%
  mutate(w_rake = 1/svyd_rake$prob,
         w_rake = w_rake / mean(w_rake))
```

```
ggplot(svy_rake, aes(w_ps2, w_rake)) +
  geom_point(size = 0.5, alpha = 0.5) +
  coord_equal(xlim = c(0.1, 2.5), ylim = c(0.1, 2.5)) +
  theme_classic() +
  labs(x = "Propensity Score Weights",
       y = "Rake Weights")
```



Coding suggestions and tips

Post-stratification

- You will not need any survey specific package for this step. You can use the `dplyr` function `count`, or the base-R tabular function `xtabs`, to get counts.
- The `count` function takes the argument `wt` to included weighted sums. The `xtabs` function takes weights on the left hand side. For example, `xtabs(w ~ X1 + X2, data)` will do a cross-tab of `X1` and `X2` by summing the respective `w`s.

CBPS

- Because you are interested in the population quantity and this is `D = 0` in the survey setting, you will want to estimate the Average Treatment on Control (ATC, or `ATT = 0` in their package) instead of the Average Treatment on Treated.
- CBPS will estimate a propensity score, so the weights are the inverse of that.

The `rake` function in `survey`

- See Doug's Sampling Lecture notes for an example of how to use the `survey` package. Although the `survey` package is probably the authoritative R package for survey statistics, its user-interface is rather specific.
- First, you'd need to register the survey dataframe as a survey design object. Because there are no pre-survey stratification in this example, you can do something like `svyd <- svydesign(id = ~1, data = svy)`
- For raking, the `rake` function will ask for a list of population margins where each item is a table of population counts. The count variable should be named `Freq`. A easy way to do this is in tidyverse: `edu_margins <- count(pop, educ, name = "Freq")`.