SURV625 Applied Sampling

2025-04-16

SM 625: Week 4 Sampling Project Notes

For each of the three variables that will be the focus of the final course project, the Department of Education would like to generate estimates of means and proportions having a coefficient of variation of no more than 0.05. Using the numbers provided to you in the description of the final project, compute estimates of the element variances for each variable. Given these estimates, compute the desired level of precision (the desired sampling variance) for each estimate that corresponds to the desired coefficient of variation.

Now, given the desired levels of precision for each estimate, compute estimates of the necessary sample sizes for each of these three estimates (assuming simple random sampling), ignoring the finite population correction. These will be starting points for the eventual two-stage cluster sample design.

We first build a table to store our results for each week's assignments.

- We also add the expected averages for each outcome variable.

```
# build dataframe with inputs
MI_school_samples <- tibble(
  Outcome = c("smoked_cig", "smoked_mj", "age_approached_to_smoke"),
  type = c("prop", "prop", "mean"),
  desire_cv = rep(.05, 3),
  expect_mean = c(.25, .15, 12),
)
  # calculate element
MI_school_samples |> kable()
```

Outcome	type	desire_cv	expect_mean
smoked_cig	prop	0.05	0.25
smoked_mj	prop	0.05	0.15
age_approached_to_smoke	mean	0.05	12.00

Our process is to:

- 1st, calculate the estimated element variance.
  - For a proportion, to get the element variance we use $\hat{p}(1-\hat{p})$.
  - For a mean, to get the element variance we simply just square the estimated standard deviation $v(\bar{y}) = \sigma^2$.
- 2nd, we calculate the estimated standard error as $se(\hat{p}) = CV \times \hat{p}$.
- 3rd, we compute the desired sampling variance as: $var(\hat{p}) = se(\hat{p})^2$, where $se(\hat{p}) = \sqrt{var(\hat{p})}$

```
MI_school_samples <- MI_school_samples |>
  mutate(
    # compute element variance
    var = if_else(type=="prop", # for proportions
                               expect_mean * (1 - expect_mean),
                               if_else(type=="mean",# for means
                                       1^2, NA)),
    # compute stand dev
    sd = sqrt(var),
    # compute standard error
    se = desire_cv * expect_mean,
    # compute desired sample variance
    V = se^2
    )

MI_school_samples |> select(-type) |> kable()
```

Outcome	desire_cv	expect_mean	var	sd	se	V
smoked_cig	0.05	0.25	0.1875	0.4330127	0.0125	0.0001563

Outcome	desire_cv	expect_mean	var	sd	se	V
smoked_mj	0.05	0.15	0.1275	0.3570714	0.0075	0.0000562
age_approached_to_smoke	0.05	12.00	1.0000	1.0000000	0.6000	0.3600000

We now estimate the desired sample sizes when we desire a CV =.05 as $n = \frac{s^2}{se^2}$

```
MI_school_samples <- MI_school_samples |>
  mutate(SRS_n = var / V)

MI_school_samples |> select(1, SRS_n) |> kable()
```

Outcome	SRS_n
smoked_cig	1200.000000
smoked_mj	2266.666667
age_approached_to_smoke	2.777778

SM 625: Week 5 Sampling Project Notes

For this week, we will consider the information available for stratified sampling of students. Eventually you are going to design a stratified cluster sample of students, where the clusters (or PSUs) are schools, but we aren't there yet.

Recall the regions of interest in the sampling project description:

```
school_frame <- read_xls(
  "~/work/d/SURV625project/data/MI_school_frame_head_counts.xls")
```

Region	County_ID
1	07, 31, 66
2	22, 27, 36, 55
3	02, 21, 52
4	17, 48, 49, 77
5	01, 04, 06, 16, 20, 26, 35, 60, 65, 68, 69, 71, 72
6	05, 10, 15, 18, 24, 28, 40, 43, 45, 51, 53, 57, 67, 83
7	03, 08, 11, 12, 13, 14, 34, 39, 41, 54, 59, 61, 62, 64, 70, 75, 80
8	09, 19, 23, 25, 29, 30, 33, 37, 38, 46, 47, 56, 73, 78, 81
9	32, 44, 50, 58, 63, 74, 76, 79, 82

As "State officials are interested in providing, if at all possible, separate estimates for each of nine education regions in the state, where the regions are defined by groups of counties", we will use these nine regions as strata.

Prepare a table that includes the:

- Overall population counts in each of these nine strata (the total count of students in the target population at each school is in the tot_all column on the sampling frame).
- Given these counts, once you have the working overall sample size (unknown for now and will be decided by your team next week), what is the proportionate allocation plan of that sample of students across these nine strata?

```
# we will use Region, County_ID, and tot_all

# region counts
strata_Prop_allocate <- school_frame |>
  group_by(Region) |>
```

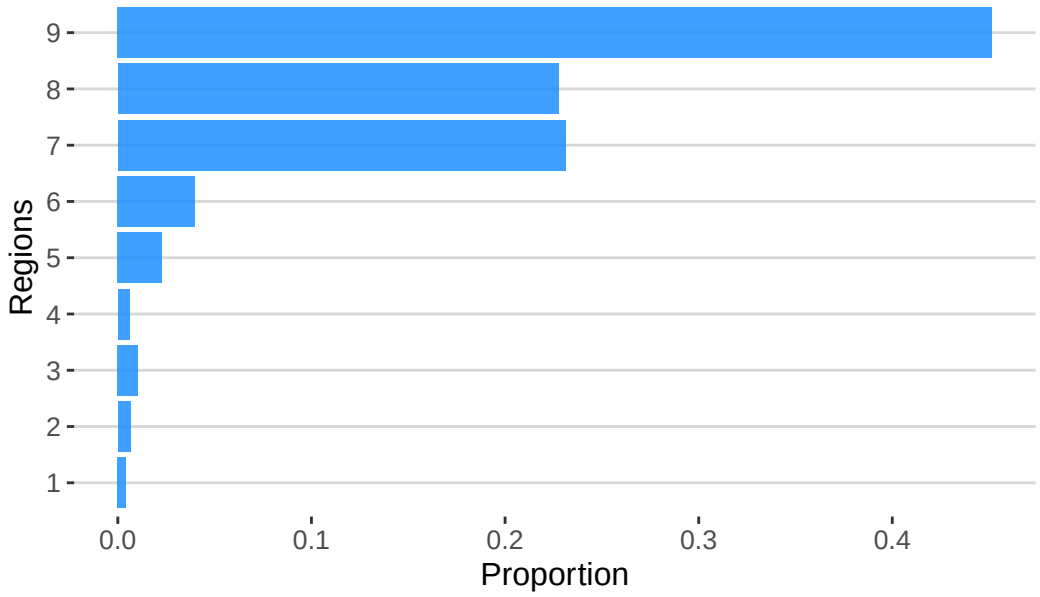
```
  reframe(M_h = sum(tot_all), # total of students in stratum
          N_h = n()) |> # total of schools in stratum
  mutate(prop_allocation = M_h/sum(M_h))

strata_Prop_allocate |>
  kable()
```

Region	M_h	N_h	prop_allocation
1	3561	20	0.0042896
2	5474	30	0.0065941
3	8631	33	0.0103971
4	4855	31	0.0058484
5	18907	80	0.0227757
6	33133	133	0.0399126
7	191992	644	0.2312772
8	188830	549	0.2274682
9	374755	923	0.4514370

```
# what is the proportionate allocation plan of that sample
## of students across these nine strata?
```

```
strata_Prop_allocate |>
  mutate(Region = factor(Region)) |>
  ggplot(aes(x=Region, y=prop_allocation)) +
  geom_col(position="dodge", fill="dodgerblue", alpha=.85) +
  coord_flip() +
  guides(fill=guide_legend(title="", reverse = TRUE)) +
  labs(
    title = "Figure 1. Proportionate Allocation Plan Across Nine Strata",
    x = "Regions",
    y = "Proportion"
  ) +
  theme_hc()
```

Figure 1. Proportionate Allocation Plan Across Nine Strata



SM 625: Week 6 Sampling Project Notes

From a previous study, you obtain estimates of the following design effects for each of these three estimates:

- proportion ever smoked one cigarette = 2.5;
- proportion ever smoked marijuana = 2.0; and
- mean age when first asked to smoke = 1.7.

This previous study featured a sample of size n = 7,500 students between the ages of 13 and 19, selected from a total of a = 150 clusters. Using this information, compute a synthetic estimate of roh for each of the three variables. These synthetic estimates of roh will be used to consider alternative cluster sample designs as you continue with your project work. Finally, budget and cost information is now available. The total budget for data collection for this project will be $500,000. The client and the data collection organization estimate that the data collection will cost $3,000 per primary stage cluster (school), and $50 per completed questionnaire within a cluster. We will use this cost information moving forward for optimal subsample size calculations.

We can estimate the sample ICC or roh from the given design effect estimate as:

$$\hat{roh} = \frac{deff - 1}{m - 1}$$

We now that the sample total is $nm = 7500$ and the sample number of cluster is $n = 150$, which we can take the mean cluster size as $m = nm/n = 7500/150 = 50$ and use it to calculate roh.

```
nm <- 7500
n <- 150
m <- nm / n

MI_school_samples <- MI_school_samples |>
  # add deff and roh to our table
  mutate(desire_deff = c(2.5, 2.0, 1.7),
         # compute roh
         roh = (desire_deff - 1) / (m - 1),
```

```
          roh = round(roh, 4)
          )

MI_school_samples |> select(Outcome, desire_deff, roh) |>  kable()
```

Outcome	desire_deff	roh
smoked_cig	2.5	0.0306
smoked_mj	2.0	0.0204
age_approached_to_smoke	1.7	0.0143

SM 625: Week 7 Sampling Project Notes

Recall that the client and the data collection organization estimated that the data collection would cost \$3,000 per primary stage cluster (school), and \$50 per completed questionnaire within a cluster. We will now use this information for optimum subsample size calculations. Recall that the total budget for data collection will be \$500,000.

Given this cost information and your estimates of roh for the three different variables of primary interest from last week, compute the optimum subsample size (and the corresponding optimal number of first stage clusters, given the total budget above) for each of the variables.

- We now have budget constraints and denote the cost per cluster as $c_n = \$3,000$ and cost per element as $c_m = \$50$, with a total budget constraint of $C = \$500,000$. Since we know there are $n = 150$ clusters and a total sample size of $7,500$ students.
- To compute the optimum m size we use the following equation:

$$m_{opt} = \sqrt{\frac{c_n}{c_m}\frac{1-roh}{roh}}$$

```
c_n = 3000 # cost per cluster
c_m = 50 # cost per element within cluster
C = 500000 # total budget

MI_school_samples <- MI_school_samples |>
  mutate(
    # compute optimum m size
    m_opt = sqrt( (c_n / c_m) * ( (1-roh)/roh) ),
    n_opt = C / (c_n + m_opt * c_m),
    # compute new deff
    deff_new = 1 + (m_opt-1) * roh,
     # compute total SSU
    total_nm = m_opt * n_opt)

MI_school_samples |> select(Outcome, m_opt, n_opt) |>
  kable()
```

Outcome	m_opt	n_opt
smoked_cig	43.59799	96.52697
smoked_mj	53.67659	87.96886
age_approached_to_smoke	64.31022	80.44391

How will you decide on a single overall optimum subsample size to use in your design?

- Above we estimated the new design effects which range from 2.3 to 1.9, which are almost in line with our desired design effects of 2.5, 1.7. Below we print the new design effects, optimum number of cluster and cluster size, total sample size `total_nm`for our projected $500,000 budget for all three outcome variables.
  - Finally, we compute the sampling cost as $n \times c_n + n \times m \times c_m$ which we defined these terms above.

```
MI_school_samples |>
  select(Outcome, m_opt, n_opt, deff_new, total_nm) |>
  mutate_at(c(2, 3, 5), floor) |>
  mutate(cost = (c_n * n_opt) + (c_m * n_opt * m_opt),
         cost = scales::dollar(cost),
         Outcome = ifelse(Outcome == "age_approached_to_smoke",
                          "age_smoke", Outcome),
             deff_new = round(deff_new, 4)) |>
  kable()
```

Outcome	m_opt	n_opt	deff_new	total_nm	cost
smoked_cig	43	96	2.3035	4208	$494,400
smoked_mj	53	87	2.0746	4721	$491,550
age_smoke	64	80	1.9053	5173	$496,000

Think about a comparison of alternative cluster sample designs: under a fixed cost constraint, how would we decide which design would be best? What will be your overall sample size (n) under this new optimum subsample size?

As you make progress in writing up what you have done so far, provide some discussion of the rationale for your choices in this regard.

Next, given this optimum subsample size and treating the values of roh as portable, compute the new expected DEFF for each estimate given the new design (this can be specific to each variable / estimate, given the different optimum subsample sizes). In addition, compute a new expected SRS variance for each variable under the new design, using the new "optimum" overall sample size (remember that you can treat the element variances for each variable estimated last week as portable). Finally, compute the new expected sampling variance for each estimate under this new cluster sample design. Are you still meeting the client's precision requirements?

- Given that we have three outcome variables, we also have three optimum number of clusters and cluster size estimates. That is, we can design and examine three options of different optimum number of clusters and cluster sizes.
- We will use the portable roh estimate and calculate new design effects, SRS variance, and complex design variance for each outcome variable.

```
map(seq(1,3), function(x){

  MI_school_samples |>
  select(Outcome, roh, m_opt, n_opt, total_nm) |>
  # we can print projected total cost for n=50
  mutate(m_opt = m_opt[x], # optimum m from first row
         n_opt = n_opt[x],
         total_nm = m_opt*n_opt,
         # calculate new deff
         deff_new = 1 + (m_opt-1) * roh,
         # recalcualte element variance
         var = c(.24, .1275, 9),
         # calcualte SRS variance
         var_srs =  var / (total_nm - 1),
         # calculate complex design variance
         var_crs = var_srs * deff_new,
         #m_opt = round(m_opt)
         ) |>
    select(-roh, -var)  |>
    mutate_at(2:3, floor) |>
    mutate(total_nm = m_opt*n_opt) |>
    mutate_at(5:7, round, 5) |>
  kable()

}) |>
  set_names(str_c(rep("Option ", 3), seq(1,3)))
```

$`Option 1`

Outcome	m_opt	n_opt	total_nm	deff_new	var_srs	var_crs
smoked_cig	43	96	4128	2.30350	0.00006	0.00013
smoked_mj	43	96	4128	1.86900	0.00003	0.00006
age_approached_to_smoke	43	96	4128	1.60915	0.00214	0.00344

$`Option 2`

Outcome	m_opt	n_opt	total_nm	deff_new	var_srs	var_crs
smoked_cig	53	87	4611	2.61190	0.00005	0.00013
smoked_mj	53	87	4611	2.07460	0.00003	0.00006
age_approached_to_smoke	53	87	4611	1.75328	0.00191	0.00334

$`Option 3`

Outcome	m_opt	n_opt	total_nm	deff_new	var_srs	var_crs
smoked_cig	64	80	5120	2.93729	0.00005	0.00014
smoked_mj	64	80	5120	2.29153	0.00002	0.00006
age_approached_to_smoke	64	80	5120	1.90534	0.00174	0.00332

We print standard error for the complex design with 95% confidence intervals, and we also flag whether the sampling variance from the clustering is equal or smaller than the desired sampling variance.

```
map(seq(1,3), function(x){

  MI_school_samples |>
  select(Outcome, expect_mean, V, roh, m_opt, n_opt, total_nm) |>
  # we can print projected total cost for n=50
  mutate(m_opt = m_opt[x], # optimum m from first row
         n_opt = n_opt[x],
         total_nm = m_opt*n_opt,
         # calculate new deff
         deff_new = 1 + (m_opt-1) * roh,
        # recalcualte element variance
         var = c(.24, .1275, 9),
         # calcualte SRS variance
         var_srs =  var / (total_nm - 1) ,
         # calculate complex design variance
         var_crs = var_srs * deff_new,
         # compute confidence intervals
         se = sqrt(var_crs),
         lower = expect_mean  - 1.96*se,
```

```
         upper = expect_mean + 1.96*se,
        # flag if var_crs is lower or = to desired sampling var
         var_ck = ifelse(var_crs <= V, "yes", "no")) |>
  select(Outcome, expect_mean, se, lower, upper, var_ck) |>
  mutate_at(3:5, round, 4) |>
  kable()

}) |>
  set_names(str_c(rep("Option ", 3), seq(1,3)))
```

$`Option 1`

Outcome	expect_mean	se	lower	upper	var_ck
smoked_cig	0.25	0.0115	0.2275	0.2725	yes
smoked_mj	0.15	0.0075	0.1352	0.1648	no
age_approached_to_smoke	12.00	0.0587	11.8850	12.1150	yes

$`Option 2`

Outcome	expect_mean	se	lower	upper	var_ck
smoked_cig	0.25	0.0115	0.2274	0.2726	yes
smoked_mj	0.15	0.0075	0.1353	0.1647	yes
age_approached_to_smoke	12.00	0.0578	11.8867	12.1133	yes

$`Option 3`

Outcome	expect_mean	se	lower	upper	var_ck
smoked_cig	0.25	0.0117	0.2271	0.2729	yes
smoked_mj	0.15	0.0075	0.1353	0.1647	no
age_approached_to_smoke	12.00	0.0576	11.8871	12.1129	yes

- Option 2 with a number of cluster of 87 and cluster size of 53 is the design we will choose given that the total sample size of 4,611 is within the allocated budget ($491,550).

We prefer this model because it stays close to the desired design effects we received from the customer. Additionally, the standard errors we estimate for this second option overall are the smallest resulting in tighter 95% confidence intervals for the expected estimates we were provided. This design in close to option 3, yet we prefer having a slightly smaller SSU if we can increase the number of PSUs sampled since this gives us a cost efficiency.

The client has also provided other new information: the estimated size of the target population is N = 830,138. Given this population size and your overall sample size (n) under the new optimum subsample size computed above, what is your overall working sampling fraction (f)? Does it seem like finite population corrections will be necessary in your sampling variances if you choose to perform SRSWOR at some point?

```
# total pop
N <- 830138

# optimum n
total_nm <- 4721
samp_frac <- total_nm / N; samp_frac
```

```
[1] 0.005687006
```

```
MI_school_samples |>  select(Outcome, expect_mean, roh,
                             m_opt, n_opt, total_nm) |>
  # we can print projected total cost for n=50
  mutate(m_opt = m_opt[2], # optimum m from first row
         n_opt = n_opt[2],
         total_nm = m_opt*n_opt,
         # calculate new deff
         deff_new = 1 + (m_opt-1) * roh,
         # recalcualte element variance
         var = c(.24, .1275, 9),
         # calcualte SRS variance with sampl_fraction
         var_srs =(1 - samp_frac) * var / (total_nm - 1),
         # calculate complex design variance
         var_crs = var_srs * deff_new,
         # compute confidence intervals
         se = sqrt(var_crs),
         lower = expect_mean  - 1.96*se,
         upper = expect_mean + 1.96*se ) |>
  select(Outcome, var_crs, se, lower, upper) |>
```

```
  mutate_at(2:4, round, 5) |>
  kable()
```

Outcome	var_crs	se	lower	upper
smoked_cig	0.00013	0.01149	0.22748	0.2725212
smoked_mj	0.00006	0.00746	0.13537	0.1646295
age_approached_to_smoke	0.00332	0.05765	11.88701	12.1129934

Our overall sampling fraction is .0057. In examining the complex design variances, and recalculating the expected standard error and 95% confidence interval given the sampling fraction, it does not appear that accounting for a population correction makes a huge impact, and we suggest it will not be necessary in our sampling variance for an SRSWOR design

SM 625: Week 8 Sampling Project Notes

Assume that you will decide to allocate your final computed n_{opt} number of clusters to each of the nine project strata based on the proportions of the total number of students in the population in each stratum (i.e., if 20% of the population of students comes from Region 1, you would sample 20% of your clusters from that region). Describe the first-stage sampling fractions for each stratum, where the total number of schools to sample at the first stage in each stratum is defined by your proportionate allocation of the n_{opt} clusters.

Next your team should extend your design to consider stratified PPeS selection of schools from each of the nine strata at the first stage of your sample design.

You have been provided with a sampling frame that lists the schools within each region. Given the information on the sampling frame, how might you sort this list to achieve implicit stratification within the regions? You can treat the overall student count from a previous year (tot_all) as the measure of size for the PPeS sampling. Given this information, compute your zone size for systematic PPeS sampling within each of the nine strata (regions), and proceed with systematic selection based on fractional intervals to select the allocated number of schools within each stratum using PPeS sampling. What is your first-stage sampling fraction within each of the nine strata?

- Using the proportionate allocation by strata computed earlier, we assign and add cluster allocation by stratum by $n_{opt} \times prop\_allocation$.
- nonresponse adjustment is achieved by taking our optimum values and adjusting them by the amount of respondents that are likely to complete the survey.
- We also calculate the zone size which we label as k_h as:

$$k_h = \frac{nMOS_i}{\sum_t MOS_i}$$

```
set.seed(9999)

# response rates
school_rr <- .30
student_rr <- .70
# Given values
n_opt <- round(87 / school_rr)
m_opt <- round(53 / student_rr)


# Compute proportional allocation of clusters to each stratum
region_summary <- strata_Prop_allocate |>
  # Ensure at least 1 cluster per
```

```
  mutate(n_h = round(n_opt * prop_allocation),
           # we need to adjust the last n_h to get an exact 290
         n_h = ifelse(n_h == 131, n_h-1, n_h)) |>
  mutate(N_h = as.double(N_h)) |>
  group_by(Region) |>
  reframe(across(where(is.double), ~ sum(.x))) |>
  mutate(
         f_h = n_h / N_h,     # sampling fraction
         k_h =  round(M_h / n_h)) |>    # zone size
  # create random start values
  rowwise() |>
  mutate(RN = sample(1:k_h, 1)) |>
  ungroup() |>
  mutate_at(vars(prop_allocation, f_h), round, 3)


region_summary |>
  select(Region, prop_allocation, n_h, k_h, RN) |>
  kable()
```

Region	prop_allocation	n_h	k_h	RN
1	0.004	1	3561	3168
2	0.007	2	2737	2310
3	0.010	3	2877	1321
4	0.006	2	2428	131
5	0.023	7	2701	2122
6	0.040	12	2761	2114
7	0.231	67	2866	374
8	0.227	66	2861	380
9	0.451	130	2883	1673

- To achieve implicit stratification we order the school list sorted by size of student in each region. To compute zone size we use

```
# sort list of schools by student size
school_frame_sorted <- school_frame #|>
  #arrange(Region, desc(tot_all))

min_MOS <- m_opt

# create vectors of selection values for each stratum
```

```r
RN_sample <- map(1:nrow(region_summary), function(x){

  # pass table created in last code chunk
  round(seq(region_summary$RN[x], # random start
      region_summary$M_h[x], # total number of students
      region_summary$k_h[x])) # k sampling interval
})
```

```r
# we link the selected blocks
dat <- school_frame_sorted |>
  group_by(Region) |>
  mutate(
    # assing ids
    id = row_number(),
    # flag if minimum MOS not met
    min_m_req = ifelse(tot_all >= min_MOS, 1 , 0),
    # create links and convert to clusters
    linking = lead(min_m_req, default=1),
    # assign clustering
    cluster = cumsum(lag(linking, default=1)),
    # add cumulative counts
    cumulative_max = cumsum(tot_all),
    cumulative_min = 1 + lag(cumulative_max, default = 0) )
```

```r
# for each region loop through RN_sample & assign selection to schools
dat_selected <- map_dfr(1:9, function(x){

  dat |>
    filter(Region %in% x) |>
    add_column(RN_sample[[x]] |> tibble() |> data.table::transpose()) |>
    # create flag for blocks that are selected
    mutate(selected =
             as.numeric(if_any(starts_with("V"), ~
                         between(.x, cumulative_min, cumulative_max)))) |>
    # drop select population elements
    select(-starts_with("V"))

})
```

```r
# this is where schools are linked
dat_linked <- dat_selected |>
  group_by(Region) |>
```

```
  mutate(
    # flag if minimum MOS not met
    min_n_req = ifelse(tot_all >= min_MOS, 1 , 0),
    # create links and convert to clusters
    linking = lead(min_n_req, default=1),
    # assign clustering
    cluster = cumsum(lag(linking, default=1))) |>
  ungroup()
```

```
# show cluster of blocks selected, total HUs
sample_selected <- map_dfr(1:9, function(x){

  linkage = dat_linked |>
    filter(Region %in% x, selected == 1) |>
    select(Region, cluster) |>
    mutate(Selection = RN_sample[[x]]) |>
    pull(cluster)

  dat = dat_linked |>
    filter(Region %in% x,
           cluster %in% linkage) |>
    mutate(MOS = as.numeric(tot_all)) |>
    group_by(cluster) |>
    mutate(
      cluster = cur_group_id(),
      total_MOS = sum(MOS, na.rm = TRUE)
    ) |>
    arrange(desc(id))  # optional: sort within cluster

  if (x == 1) {
    # get unique cluster id from Region 1
    first_cluster_id <- dat |>
      filter(Region == 1) |>
      pull(cluster) |>
      unique() |>
      min()

    # filter the first cluster
    first_cluster <- dat |> filter(cluster == first_cluster_id)

    # split it into two halves (or roughly)
    n <- nrow(first_cluster)
```

```
    first_half <- first_cluster[1:floor(n/2), ] |>
      mutate(
        SECU = "1A",
        SECU_MOS = sum(MOS/2, na.rm = TRUE)
      )
    second_half <- first_cluster[(floor(n/2) + 1):n, ] |>
      mutate(
        SECU = "1B",
        SECU_MOS = sum(MOS/2, na.rm = TRUE)
      )

    # everything else from Region 1
    remaining <- dat |> filter(cluster != first_cluster_id) |>
      mutate(SECU = as.character(cluster),
             SECU_MOS = total_MOS)

    # combine all Region 1 units
    dat <- bind_rows(first_half, second_half, remaining)
  } else {
    dat <- dat |>
      mutate(
        SECU = as.character(cluster),
        SECU_MOS = total_MOS
      )
  }

  return(dat)
}) |>
  ungroup()

# save data to github
#write_xlsx(sample_selected,
 #          "~/work/d/SURV625project/data/sample_selected.xlsx")
```

```
# Form Pseudo-Strata for Paired Selection Model
pseudo_strata_df <- sample_selected |>
  group_by(Region) |>
  arrange(desc(MOS)) |>   # optionally sort by size for pairing
  mutate(row_in_group = row_number(),
         pseudo_stratum_id = paste0("R", Region, "_P", ceiling(row_in_group / 2))) |>
  ungroup()
```

```r
# how many pseudo strata in each region?
pseudo_strata_df |>
  group_by(Region) |>
  distinct(pseudo_stratum_id) |>
  count() |>
  kable()
```

Region	n
1	1
2	3
3	2
4	1
5	6
6	11
7	50
8	39
9	89

SM 625: Week 10 Sampling Project Notes

There are four primary tasks for your team to consider over the next week:

1. Given your overall m_{opt} n_opt and N (based on the sampling frame), you've already computed the overall sampling fraction, . For each of the nine strata, compute the required number of students to subsample from each sampled school based on the stratified PPeS design in order to maintain epsem across all strata.

- Within strata, retain epsem for stratified PPS sampling across strata $f = f_h$ for all h.

$$f_h = \frac{n_h MOS_{hi}}{\sum_{,i \in h} MOShi} \frac{m_h^*}{MOS_{hi}}$$

```
# Required Students per School (m_h_star) to Maintain EPSEM:
region_summary<- region_summary |>
  mutate(m_h_star=c(samp_frac*k_h))

region_summary |>
  select(Region, k_h, RN, m_h_star) |>
  kable()
```

Region	k_h	RN	m_h_star
1	3561	3168	20.25143
2	2737	2310	15.56534
3	2877	1321	16.36152
4	2428	131	13.80805
5	2701	2122	15.36060
6	2761	2114	15.70182
7	2866	374	16.29896
8	2861	380	16.27052
9	2883	1673	16.39564

2. Do each of the schools that you sampled in a given region have the minimum sufficient size, given the stratum-specific subsample sizes computed in Task #1? Do subsequent schools on the list have the minimum sufficient size? If not, what will you do?

```
region_min_MOS <- region_summary %>%
  group_by(Region) %>%
  mutate(
    min_MOS2 = ceiling(m_h_star / (0.3 * 0.7))  # Total response rate = 0.21, expanded sample
```

```r
  )

# Processing schools by region and generating clusters of links
linked_schools <- sample_selected %>%
  left_join(region_min_MOS, by = "Region") %>%  # Combined Minimum MOS
  group_by(Region) %>%
  arrange(desc(tot_all)) %>%  # Listed in descending order of MOS (prioritizing large schools
  mutate(
    # Initialize cumulative MOS and link tags
    cumulative_mos = cumsum(tot_all),
    need_link = if_else(tot_all < min_MOS2, 1, 0),
    # Dynamic generation of cluster IDs: linking when cumulative MOS is insufficient
    cluster_id = cumsum(
      if_else(
        cumulative_mos - lag(cumulative_mos, default = 0) >= min_MOS2 | row_number() == 1,
        1, 0
      )
    )
  ) %>%
  ungroup()

# how many linked clusters by region
linked_schools |>
  group_by(Region) |>
  count(cluster_id) |>
  count() |>
  kable()
```

Region	n
1	2
2	2
3	3
4	2
5	7
6	12
7	66
8	66
9	130

```r
# Summarize the total MOS for each cluster and check for compliance
cluster_summary <- linked_schools %>%
  group_by(Region, cluster_id) %>%
  summarise(
    total_mos = sum(tot_all),
    schools = toString(BCODE),
    min_MOS2 = first(min_MOS2),
    .groups = "drop"
  ) %>%
  mutate(
    sufficient = if_else(total_mos >= min_MOS2, "Yes", "No")
  )
# Output clusters that need to be relinked (total MOS still insufficient)
clusters_to_relink <- cluster_summary %>% filter(sufficient == "No")

# Recursive linking until all clusters are up to standard
while (nrow(clusters_to_relink) > 0) {
  linked_schools <- linked_schools %>%
    group_by(Region) %>%
    mutate(
      cluster_id = if_else(
        cluster_id %in% clusters_to_relink$cluster_id,
        cluster_id + 1,  # Merge to the next cluster
        cluster_id
      )
    ) %>%
    ungroup()

  # Summary of recomputation clusters
  cluster_summary <- linked_schools %>%
    group_by(Region, cluster_id) %>%
    summarise(
      total_mos = sum(tot_all),
      schools = toString(BCODE),
      min_MOS2 = first(min_MOS2),
      .groups = "drop"
    ) %>%
    mutate(sufficient = if_else(total_mos >= min_MOS2, "Yes", "No"))

  clusters_to_relink <- cluster_summary %>% filter(sufficient == "No")
}
```

```r
final_clusters <- linked_schools %>%
  group_by(Region, cluster_id) %>%
  summarise(
    linked_schools = paste(BCODE, collapse = ", "),
    total_mos = sum(tot_all),
    min_MOS2 = first(min_MOS2),
    .groups = "drop"
  ) %>%
  mutate(
    status = if_else(total_mos >= min_MOS2, "Valid", "Invalid")
  )


linked_schools <- linked_schools %>%
  left_join(
    cluster_summary %>% select(Region, cluster_id, total_mos),
    by = c("Region", "cluster_id")
  )

# Print results
final_clusters |>
  select(Region:total_mos) |>
  kable(col.names = c("h", "id", "n", "m"))
```

h	id	n	m
1	1	08558	190
1	2	08558	190
2	1	02514	261
2	2	08831, 09599, 09008, 08877, 08944	251
3	1	01858	261
3	2	05145	222
3	3	00621, 05362	223
4	1	00404	245
4	2	03024	152
5	1	00655	737
5	2	04050	621
5	3	01482	580
5	4	06369	259
5	5	01935	217
5	6	08833	168
5	7	04655, 08867, 01517, 00909, 08870, 08878	256

h	id	n	m
6	1	00554	2078
6	2	08470	1859
6	3	04200	1602
6	4	03171	586
6	5	02339	569
6	6	00487	499
6	7	02792	317
6	8	01045	289
6	9	06394	228
6	10	08993	226
6	11	07625	125
6	12	01817, 09039, 07673, 02881, 09041, 07707, 09040, 04839, 05086, 02816	372
7	1	04462	2299
7	2	03246	1950
7	3	05974	1849
7	4	01455	1840
7	5	01463	1830
7	6	06127	1578
7	7	03095	1412
7	8	00223	1383
7	9	02652	1373
7	10	01265	1368
7	11	02106	1359
7	12	03175	1326
7	13	02587	1326
7	14	02272	1315
7	15	03097	1304
7	16	00491	1304
7	17	01498	1212
7	18	04623	1044
7	19	02704	1037
7	20	00570	1031
7	21	01848	1017
7	22	03793	992
7	23	04095	990
7	24	01013	932
7	25	02095	929
7	26	04251	905
7	27	00794	858
7	28	00882	831
7	29	01096	783

h	id	n	m
7	30	04510	713
7	31	02865	693
7	32	06412	654
7	33	00322	654
7	34	00610	598
7	35	02005	576
7	36	01757	550
7	37	04906	509
7	38	05474	502
7	39	00435	500
7	40	08350	493
7	41	01430	477
7	42	00059	437
7	43	06272	436
7	44	01575	435
7	45	03562	422
7	46	08457	383
7	47	02997	336
7	48	01655	329
7	49	06045	313
7	50	06310	306
7	51	00754	301
7	52	09403	292
7	53	03461	280
7	54	02402	271
7	55	09187	266
7	56	03308	234
7	57	08845	217
7	58	02776	192
7	59	05799	174
7	60	05056	173
7	61	08633	167
7	62	07756	167
7	63	06811	166
7	64	04336	152
7	65	05886	118
7	66	08016, 07290, 06322, 03812, 05487, 01516, 03362, 09531, 06133, 03854, 09525, 03820, 07293, 03357, 00241, 09471, 03881, 05106, 03360, 03885, 02787, 05480, 07294, 06934, 03850, 01850, 04629, 06159, 09410, 05136, 03311, 06908, 06048, 04576, 09523	1115
8	1	01453	2562

h	id	n	m
8	2	04882	2488
8	3	00402	2243
8	4	05671	1946
8	5	05158	1779
8	6	06203	1772
8	7	01870	1732
8	8	05009	1642
8	9	02555	1633
8	10	01166	1623
8	11	02187	1549
8	12	01256	1537
8	13	00878	1536
8	14	00227	1529
8	15	00125	1408
8	16	01044	1407
8	17	02822	1404
8	18	01711	1391
8	19	06273	1210
8	20	04111	1205
8	21	05690	1201
8	22	01993	1163
8	23	00027	1111
8	24	05708	1110
8	25	02426	1082
8	26	00732	1046
8	27	04143	975
8	28	00648	952
8	29	00912	916
8	30	03385	907
8	31	06673	817
8	32	06118	755
8	33	08715	728
8	34	07691	697
8	35	01859	653
8	36	09383	632
8	37	06233	631
8	38	00972	596
8	39	03532	590
8	40	01297	573
8	41	02750	565
8	42	02467	541

h	id	n	m
8	43	02920	532
8	44	02635	528
8	45	01625	527
8	46	01391	526
8	47	05950	504
8	48	06304	484
8	49	05138	475
8	50	04515	385
8	51	02316	374
8	52	00440	359
8	53	00284	351
8	54	05801	319
8	55	03021	297
8	56	08178	262
8	57	09005	244
8	58	03112	241
8	59	00558	216
8	60	01543	206
8	61	03935	182
8	62	08573	158
8	63	05453	145
8	64	04415	127
8	65	09774	118
8	66	07242, 06326, 08716, 04981, 09779, 05808, 04956, 03947, 08590, 08580, 04960, 00310, 04974	459
9	1	08000	2467
9	2	02088	2426
9	3	01261	2380
9	4	02772	2195
9	5	04226	2108
9	6	00025	2104
9	7	04848	2091
9	8	02644	2070
9	9	08997	2027
9	10	06276	2014
9	11	05770	2013
9	12	06265	1989
9	13	06487	1988
9	14	05315	1961
9	15	06171	1953
9	16	00089	1930

h	id	n	m
9	17	04407	1897
9	18	03256	1868
9	19	00706	1865
9	20	00814	1860
9	21	06393	1859
9	22	06491	1733
9	23	02124	1726
9	24	00250	1713
9	25	01801	1691
9	26	04340	1685
9	27	01512	1641
9	28	09050	1620
9	29	05705	1610
9	30	01308	1610
9	31	01840	1593
9	32	01003	1543
9	33	03090	1534
9	34	05596	1517
9	35	01092	1505
9	36	05819	1492
9	37	01359	1475
9	38	07680	1419
9	39	06019	1419
9	40	03521	1389
9	41	02207	1352
9	42	04608	1346
9	43	04267	1345
9	44	02798	1337
9	45	03542	1316
9	46	01944	1311
9	47	02855	1306
9	48	00291	1250
9	49	03260	1244
9	50	00385	1233
9	51	01154	1224
9	52	00065	1206
9	53	00785	1176
9	54	02149	1175
9	55	03167	1167
9	56	05976	1160
9	57	02105	1154

h	id	n	m
9	58	02756	1150
9	59	00902	1109
9	60	02030	1067
9	61	01242	1045
9	62	07786	1018
9	63	02592	982
9	64	00631	960
9	65	01706	907
9	66	06177	834
9	67	00875	827
9	68	09740	807
9	69	05957	803
9	70	02416	776
9	71	09420	745
9	72	05879	691
9	73	03505	681
9	74	00519	668
9	75	00729	645
9	76	03555	641
9	77	03238	629
9	78	06425	628
9	79	09481	625
9	80	02956	619
9	81	06591	614
9	82	00700	605
9	83	04458	593
9	84	03216	587
9	85	01050	560
9	86	06678	553
9	87	01956	548
9	88	09153	545
9	89	09889	519
9	90	07497	519
9	91	08803	513
9	92	06367	513
9	93	00677	512
9	94	03079	506
9	95	05197	499
9	96	06292	493
9	97	00968	472
9	98	00552	461

h	id	n	m
9	99	02060	445
9	10000165		443
9	10101578		434
9	10202683		412
9	10305682		408
9	10409341		370
9	10504914		361
9	10609604		359
9	10701216		358
9	10806862		355
9	10908569		338
9	11005071		338
9	11108435		336
9	11203703		333
9	11302375		331
9	11402444		330
9	11508255		301
9	11601904		299
9	11704172		291
9	11807857		282
9	11907067		260
9	12008945		254
9	12105450		223
9	12203383		215
9	12304055		201
9	12408346		193
9	12504528		192
9	12604939		168
9	12708640		161
9	12807633		127
9	12908722		91
9	13001662, 08004, 05102, 05329, 09901, 03080, 09605, 03720, 09430, 08044, 06217, 09907, 01717, 03710, 09751, 09441, 02982, 01727, 06284, 02620, 03723, 09432, 02866, 09906, 09342, 05828, 01733, 05092, 01737, 06915, 06023, 05341, 00885, 05123, 02984, 05456, 01590, 05205, 09429, 06907, 08808, 02877, 01742, 05125, 05088, 04948, 08951, 05454		1477

Selection Technique:

Systematic sampling is a suitable technique. For school h_i

- Calculate the sampling interval $k_h = MOS_{hi}/n_h$.
- Choose a random starting number between 1 and k_{hi}.
- Select the student at the random start position and every $k'_{hi}th$ student thereafter from the ordered roster.
- If schools are linked due to insufficient numbers, the rosters need to be combined and sampled uniformly.
- Record unresponsive students and report adjusted weights.

4. Write down the overall sampling fraction based on the stratified PPeS design, indicating the overall probability of inclusion for a given student, from a given school (or linked set of schools), in a given stratum. Be careful with notation. Keep in mind that the MOS values used for the sampled schools at the first stage and the denominator at the second stage (Did you sample a single school? Or a linked set of schools?) will depend on your response to Task #2 above

- The overall sampling fraction is $f = \frac{n}{N} = \frac{4{,}721}{830138} = .0057$
- The inclusion probability for a given student is $P_{hi} = \frac{n_h \times MOS_{hi}}{MOS_h} \times \frac{m_h}{MOS_{hi}} = \frac{n_h \times m_h}{MOS_h}$.

```
linked_schools<- linked_schools%>%
  group_by(Region)%>%
  mutate(P_h = n_h*total_mos/M_h,
         P_i=m_h_star/total_mos,
         Prob=P_h*P_i,
         epsem_check = abs(Prob - mean(Prob)) < 1e-6)

stopifnot(all(linked_schools$epsem_check))

# check for false
linked_schools |>
  group_by(Region) |>
  reframe(Schools=n(), Students = sum(m_h_star )) |>
  kable()
```

Region	Schools	Students
1	2	40.50286
2	6	93.39202

Region	Schools	Students
3	4	65.44607
4	2	27.61610
5	12	184.32725
6	21	329.73831
7	100	1629.89599
8	78	1269.10094
9	177	2902.02811

SM 625: Week 11 Sampling Project Notes

```
# Step 1:  Calculate how many students to sample
f_overall <- 0.0057
oneschool <- school_frame
MOS_7 <- 242
ACT_MOS_7 <- nrow(oneschool)
M_h_7 <- 191992
m_h_start_7 <- ceiling(16.29896)
# Step 2: Sampling rate
sam_rate <- m_h_start_7/ACT_MOS_7


# Step 2: Calculate sampling interval
k_interval <- ACT_MOS_7 / m_h_start_7
round_k_interval <- k_interval*100000
round_mos <- 219*100000+99999



# Step 3: Random start between 1 and interval
set.seed(123)
start <- sample(1:round_k_interval, 1)

# Step 4: Select every `interval`-th student starting from `start`
indices <- seq(start, by = round_k_interval, length.out = m_h_start_7)
true_indices <- floor(indices/100000)
sampled_students <- oneschool[true_indices, ]

# View sampled students
sampled_students |> kable()
```

BCOSNAME	DNAME	DCODE	District_Name	Region	County	County_Name	Public/nonpublic	grade_7_total	grade_8_total	grade_9_total	grade_10_total	grade_11_total	grade_12_total	total_all
06812	Malcolm High School	17010	Sault Ste. Marie Area Schools	4	17	Chippewa	Public	0	0	0	9	17	38	64
03568	SPRING VALE ACADEMY	78110	NA	6	83	Wexford	nonpublic	0	0	7	14	13	17	51
00921	DIVINE CHILD ELEMEN-TARY SCH	82030	NA	7	41	Kent	nonpublic	93	98	0	0	0	0	191
03217	Riverside School	11670	Hagar Township S/D #6	7	11	Berrien	Public	3	6	0	0	0	0	9
05485	ST STEPHEN PARISH SCHOOL	73010	NA	7	13	Calhoun	nonpublic	32	32	0	0	0	0	64
08007	REFORMED HER-ITAGE CHRIS-TIAN	39010	NA	7	11	Berrien	nonpublic	7	3	4	3	3	4	24
09904	Alpha House	64080	Shelby Public Schools	7	64	Oceana	Public	3	2	2	3	0	0	10
02436	Howell High School	47070	Howell Public Schools	8	47	Livingston	Public	0	1	678	633	636	632	2580
05150	Alternative Education	47060	Hartland Consoli-dated Schools	8	47	Livingston	Public	0	0	6	15	33	43	97
08061	LOYOLA HIGH SCHOOL	82010	NA	8	78	Shiawassee	nonpublic	0	0	43	43	42	31	159
00163	Bad Axe High School	32010	Bad Axe Public Schools	9	32	Huron	Public	0	0	102	96	131	114	443

BCODE	BNAME	DCODE	District_Name	Region	County	County_Name	Public/nonpublic	grade_7_total	grade_8_total	grade_9_total	grade_10_total	grade_11_total	grade_12_total	total_all
01798	Huron High School	82344	Huron School District	9	82	Wayne	Public	0	0	235	242	185	225	887
03232	COWDEN LAKE BIBLE ACADEMY	59093	NA	9	82	Wayne	nonpublic	0	3	2	2	1	1	9
04954	Annapolis High School	82040	Dearborn Heights School District #7	9	82	Wayne	Public	0	0	240	316	148	120	824
06618	STURGIS CHRIS-TIAN SCHOOL	75011	NA	9	63	Oakland	nonpublic	1	2	1	1	1	3	9
08569	Cesar Chavez Middle School	82916	Cesar Chavez Academy	9	82	Wayne	Public	175	163	0	0	0	0	338
09612	Blanche Kelso Bruce Academy-St. Jude's	82973	Blanche Kelso Bruce Academy	9	82	Wayne	Public	1	4	4	1	0	0	10

Week 13

1. Based on the final sample design that your team has developed, formulate a sampling error calculation model that users of your data will be able to employ to estimate sampling variance. That is, what stratum codes will you provide to users? How will you form sampling error computation units (SECUs)? How many SECUs will there be per stratum? What are expected sample sizes per SECU?

```
# Week 13 - Q1: Sampling Error Calculation Model


# Inputs based on project design
a_select <- 290 # n_opt w/ RR adjustment
b_star <- 53  # m_opt w/o RR adjustment
n_strata_var <- a_select / 2
n_regions <- 9

# Each pair of schools forms one variance stratum
# Each school is one SECU
variance_strata <- tibble(
  school_id = 1:a_select,
  var_stratum_id = rep(1:n_strata_var, each = 2),
  SECU_id = 1:a_select,
  expected_sample_size = b_star
)

# Summary table for documentation
summary_table <- tibble(
  Description = c(
    "Total Schools Selected (a_select)",
    "Explicit Strata (Regions)",
    "Variance Estimation Strata (Paired PSUs)",
    "SECUs per Variance Stratum",
    "Expected Sample Size per SECU (b*)"
  ),
  Value = c(
    a_select,
    n_regions,
    n_strata_var,
    2,
    b_star
  )
```

```
)

kable(summary_table, align = "lc")
```

Description	Value
Total Schools Selected (a_select)	290
Explicit Strata (Regions)	9
Variance Estimation Strata (Paired PSUs)	145
SECUs per Variance Stratum	2
Expected Sample Size per SECU (b*)	53

```
kable(head(variance_strata))
```

school_id	var_stratum_id	SECU_id	expected_sample_size
1	1	1	53
2	1	2	53
3	2	3	53
4	2	4	53
5	3	5	53
6	3	6	53

```
#kable(variance_strata)
```

2. Describe the variance estimation procedures that one would employ to form a confidence interval for one of the three key descriptive parameters. This should build on your proposed SECUs from the first task. How many degrees of freedom will your sampling error calculation model afford? In addition, write the formula for one of the estimated proportions or means; are weights necessary in forming this estimator, given your sample design? That is, is your design epsem, or will weights be needed to compensate for unequal probabilities of selection?

```
a_select <- 290
df <- a_select / 2           # Degrees of freedom = number of variance strata
num_strata <- df


# Simulate SECU-level estimates for a descriptive proportion (e.g., smoked a cigarette)
set.seed(9999)
```

```r
p_secu_1 <- runif(num_strata, 0.22, 0.28)  # SECU 1 estimates
p_secu_2 <- runif(num_strata, 0.22, 0.28)  # SECU 2 estimates


# Paired Difference Variance Estimation
diffs <- p_secu_1 - p_secu_2
var_estimate <- mean(diffs^2) / 2         # Variance across pairs
se_estimate <- sqrt(var_estimate)

# Confidence interval (95%)
t_crit <- qt(0.975, df = df)
estimate_mean <- mean(c(p_secu_1, p_secu_2))
CI_lower <- estimate_mean - t_crit * se_estimate
CI_upper <- estimate_mean + t_crit * se_estimate
```

Degrees of Freedom (df): 145

Standard Error (SE): 0.01789

95% Confidence Interval for estimated proportion:

[0.2135 , 0.2842]

Estimator Formula for Proportion:

$\hat{p}$ = sum(w_hij * y_hij) / sum(w_hij)

where:

y_hij = 1 if student j in school i of stratum h has the trait (e.g., smoked), 0 otherwise

w_hij = final weight for student hij (includes selection probability, nonresponse, etc.)

Are weights needed? YES.

1. Although the design aimed for EPSEM, weights are necessary in practice.

2. Adjustments are needed for:

- School-level nonresponse (30%)

- Student-level nonresponse (70%)

3. Weights also adjust for second-stage linking or other deviations during implementation.

3. Keep in mind the client's request for estimates and inference related to a 20% subclass. Will confidence intervals for the subclass be formed in the same way? Are your SECUs large enough to accommodate this request?

```
total_secus <- a_select             # Number of SECUs (schools)
expected_b_star <- b_star          # Expected completes per SECU
subclass_pct <- 0.20           # Subclass proportion (20%)
df <- total_secus / 2          # Degrees of freedom remains the same

# Estimate expected subclass size per SECU
expected_subclass_per_secu <- expected_b_star * subclass_pct
```

```
  Subclass Estimation for 20% Group

  Confidence Intervals:

CI for subclass estimates can be formed using the same paired difference method.

Degrees of Freedom remains: 145

  SECU Size Check:

Expected sample size per SECU (b*): 53

Estimated subclass members per SECU: 10.6

  This is generally sufficient for stable variance estimation at the subclass level.
```