SURV625 Applied Sampling

2025-04-15

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) |>
```

```r
  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

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

```r
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()
```

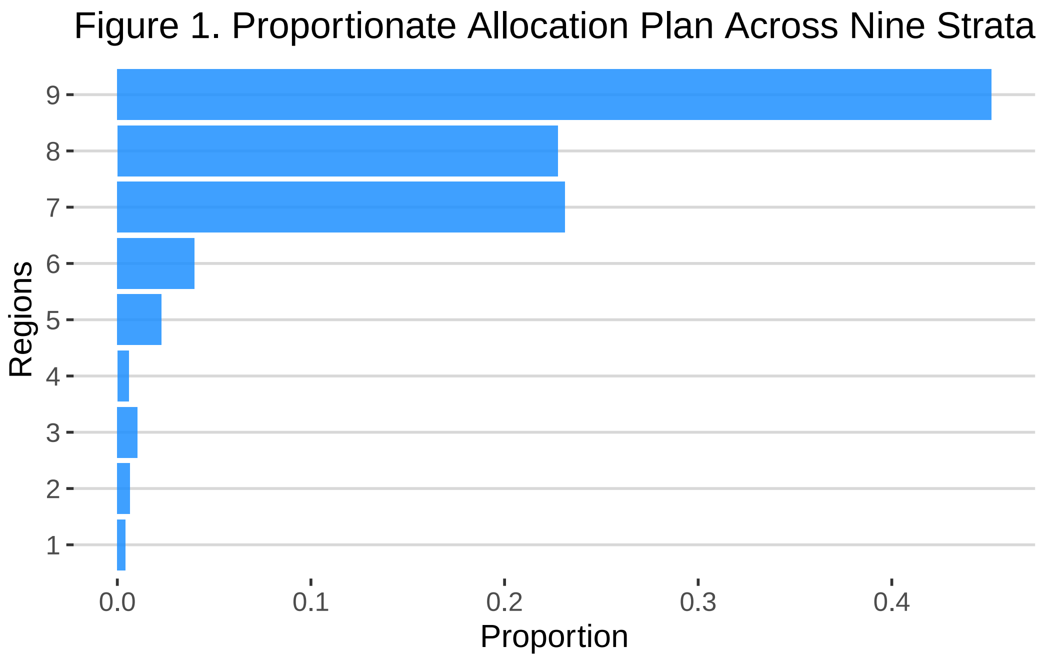
Figure 1. Proportionate Allocation Plan Across Nine Strata
Regions
9
8
7
6
5
4
3
2
1
0.0
0.1
0.2
0.3
0.4
Proportion

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)
```

```
        )

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

Outcome	desire_deff	roh
smoked_cig	2.5	0.0306122
smoked_mj	2.0	0.0204082
age_approached_to_smoke	1.7	0.0142857

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	43.58899	96.53536
smoked_mj	53.66563	87.97734
age_approached_to_smoke	64.34283	80.42281

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) |>
  mutate_at(2:3, floor) |>
  mutate(total_nm = n_opt*m_opt,
         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)) |>
  kable()
```

Outcome	m_opt	n_opt	deff_new	total_nm	cost
smoked_cig	43	96	2.303745	4128	$494,400
smoked_mj	53	87	2.074809	4611	$491,550
age_smoke	64	80	1.904898	5120	$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|  2.30374| 0.00006| 0.00013|
|smoked_mj                |    43|    96|     4128|  1.86916| 0.00003| 0.00006|
|age_approached_to_smoke  |    43|    96|     4128|  1.60841| 0.00214| 0.00344|
```

```
$`Option 2`


|Outcome                 | m_opt| n_opt| total_nm| deff_new| var_srs| var_crs|
|:-----------------------|-----:|-----:|--------:|--------:|-------:|-------:|
|smoked_cig              |    53|    87|     4611|  2.61221| 0.00005| 0.00013|
|smoked_mj               |    53|    87|     4611|  2.07481| 0.00003| 0.00006|
|age_approached_to_smoke |    53|    87|     4611|  1.75237| 0.00191| 0.00334|

$`Option 3`


|Outcome                 | m_opt| n_opt| total_nm| deff_new| var_srs| var_crs|
|:-----------------------|-----:|-----:|--------:|--------:|-------:|-------:|
|smoked_cig              |    64|    80|     5120|  2.93907| 0.00005| 0.00014|
|smoked_mj               |    64|    80|     5120|  2.29271| 0.00002| 0.00006|
|age_approached_to_smoke |    64|    80|     5120|  1.90490| 0.00174| 0.00331|
```

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.

```r
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| 11.8872| 12.1128|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?

```r
# total pop
N <- 830138

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

```
[1] 0.005687006
```

```r
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	0.2725237
smoked_mj	0.00006	0.00746	0.13537	0.1646310
age_approached_to_smoke	0.00332	0.05764	11.88703	12.1129702

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) |>
  kable()
```

Region	prop_allocation	n_h	k_h
1	0.004	1	3561
2	0.007	2	2737
3	0.010	3	2877
4	0.006	2	2428
5	0.023	7	2701
6	0.040	12	2761
7	0.231	67	2866
8	0.227	66	2861
9	0.451	130	2883

- 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)
```

```r
    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")
```

```r
# 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	6
3	2
4	1
5	18
6	27
7	140
8	112
9	188

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.

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

```r
# 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 |> kable()
```

Region	M_h	N_h	prop_allocation	n_h	f_h	k_h	RN	m_h_star
1	3561	20	0.004	1	0.050	3561	3168	20.25143
2	5474	30	0.007	2	0.067	2737	2310	15.56534
3	8631	33	0.010	3	0.091	2877	1321	16.36152
4	4855	31	0.006	2	0.065	2428	131	13.80805
5	18907	80	0.023	7	0.088	2701	2122	15.36060
6	33133	133	0.040	12	0.090	2761	2114	15.70182
7	191992	644	0.231	67	0.104	2866	374	16.29896
8	188830	549	0.227	66	0.120	2861	380	16.27052
9	374755	923	0.451	130	0.141	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?

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

```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	65
8	65
9	127

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

h	id	n	m
1	1	00652	130
1	2	00652	130
2	1	01852	308
2	2	08831, 09599, 04636, 08221, 09008, 04360, 01770, 08877, 09851, 01762, 08944	493
3	1	01155	928
3	2	06268	464
3	3	00621	217
4	1	06068	893
4	2	05631	278
5	1	01375	1090
5	2	00655	737
5	3	04050	621
5	4	01374	486
5	5	06355	294
5	6	03767	203
5	7	02174, 05867, 04036, 08867, 07932, 07898, 07472, 09822, 08679, 08481, 01517, 08531, 08537, 00909, 08870, 04631, 09026, 08343, 08341, 04212, 00423, 06963, 09689, 05580, 09597, 08878, 06225, 04842, 05098	761
6	1	08470	1859
6	2	04200	1602
6	3	02279	794
6	4	03171	586
6	5	02041	504
6	6	01992	461
6	7	01018	370
6	8	02792	317
6	9	07453	254
6	10	00609	212
6	11	08243	149

h	id	n	m
6	12	08597, 08626, 02764, 01741, 09039, 08340, 07673, 03990, 07052, 08060, 03568, 09518, 07133, 08342, 03870, 02881, 03678, 09041, 04006, 07707, 08936, 07812, 08545, 03325, 09040, 08257, 03777, 07492, 05941, 00242, 05465, 05909, 09783, 08976, 04839, 08391, 08873, 05086, 03504, 09114, 02816, 09748	1390
7	1	04462	2299
7	2	03246	1950
7	3	01455	1840
7	4	01463	1830
7	5	06127	1578
7	6	00223	1383
7	7	01265	1368
7	8	02587	1326
7	9	01697	1317
7	10	00491	1304
7	11	02017	1234
7	12	02768	1165
7	13	01785	1138
7	14	00570	1031
7	15	00475	1004
7	16	04095	990
7	17	06294	966
7	18	03253	946
7	19	09316	928
7	20	03065	898
7	21	03197	874
7	22	03560	851
7	23	03567	833
7	24	00062	802
7	25	01096	783
7	26	09057	753
7	27	03440	743
7	28	00765	705
7	29	02865	693
7	30	01475	676
7	31	00322	654
7	32	06022	640
7	33	04586	613
7	34	00610	598
7	35	02005	576
7	36	06357	559
7	37	04651	535

h	id	n	m
7	38	04652	510
7	39	00435	500
7	40	05229	485
7	41	01607	479
7	42	00189	470
7	43	01519	457
7	44	00775	442
7	45	06000	435
7	46	08421	424
7	47	02695	404
7	48	00539	389
7	49	04398	372
7	50	08059	360
7	51	03218	341
7	52	08574	330
7	53	04509	321
7	54	05440	312
7	55	09403	292
7	56	07783	271
7	57	06530	256
7	58	08172	237
7	59	09555	216
7	60	08800	195
7	61	04246	175
7	62	00201	155
7	63	03004	139
7	64	09290	119

h	id	n	m
7	65	08503, 00576, 08372, 09913, 08967, 08817, 02193, 03546, 06322, 01731, 01448, 04967, 06741, 03812, 05487, 08948, 08161, 07764, 03681, 09922, 08576, 04032, 03624, 05790, 04787, 01516, 02585, 09898, 08919, 08530, 05839, 03362, 07389, 02366, 05794, 09562, 09309, 07256, 06730, 07943, 07765, 07289, 05619, 01735, 00671, 03963, 03713, 02577, 09531, 06133, 04214, 08227, 06608, 01233, 03854, 08410, 05428, 01398, 00685, 08923, 06434, 09525, 06215, 03820, 04929, 09757, 07293, 09002, 07917, 04020, 03699, 04002, 03357, 03185, 00241, 09699, 09304, 03656, 09471, 03881, 09542, 05106, 03778, 00108, 03360, 09329, 09149, 07935, 05387, 03885, 01200, 09129, 07391, 05940, 02916, 08556, 07684, 06524, 05342, 02904, 02787, 08212, 08165, 05480, 05470, 09640, 09068, 08007, 08881, 04014, 03732, 01726, 08973, 06031, 03629, 09635, 08239, 07784, 07005, 03680, 09598, 08583, 01823, 09766, 09764, 09107, 08994, 01829, 04209, 00426, 08900, 08442, 03945, 02680, 08367, 07294, 04756, 04208, 09743, 05455, 08539, 08231, 07617, 06934, 04703, 04022, 03850, 01850, 08781, 07404, 04629, 09176, 08666, 08321, 07890, 07841, 06976, 06451, 05841, 03779, 02901, 09903, 09085, 06159, 03959, 00722, 00010, 09410, 08758, 05327, 05297, 03217, 03099, 09394, 06711, 05231, 05169, 05136, 03974, 03526, 09768, 08860, 08837, 01743, 00222, 08236, 06332, 03783, 00102, 09369, 09076, 09075, 04691, 04332, 03311, 06473, 05458, 05041, 04750, 09712, 08179, 07267, 06908, 06048, 06020, 05307, 05187, 04576, 09523, 08955, 08883, 07288, 06602, 06461, 05291, 00602	6058
8	1	02436	2580
8	2	01453	2562
8	3	04882	2488
8	4	00402	2243
8	5	01601	1804
8	6	05158	1779
8	7	01870	1732
8	8	01457	1725
8	9	02555	1633
8	10	02187	1549
8	11	00878	1536
8	12	03038	1523
8	13	00125	1408
8	14	02822	1404
8	15	06257	1366
8	16	04111	1205
8	17	00913	1171
8	18	02636	1119
8	19	05708	1110
8	20	04403	1061
8	21	01286	984
8	22	02558	969
8	23	03336	951

h	id	n	m
8	24	00618	928
8	25	06199	915
8	26	06207	902
8	27	08606	886
8	28	00879	851
8	29	06673	817
8	30	04123	768
8	31	05166	742
8	32	05774	701
8	33	01775	673
8	34	01257	651
8	35	09383	632
8	36	00972	596
8	37	05333	585
8	38	06282	569
8	39	03249	549
8	40	05798	541
8	41	04144	523
8	42	00536	504
8	43	05883	495
8	44	08712	481
8	45	00649	475
8	46	03159	460
8	47	03393	447
8	48	00746	435
8	49	00353	419
8	50	00190	399
8	51	04515	385
8	52	02685	361
8	53	01058	345
8	54	00218	318
8	55	06942	303
8	56	00973	278
8	57	01019	262
8	58	05687	243
8	59	00558	216
8	60	00131	197
8	61	07059	180
8	62	08741	156
8	63	08738	129
8	64	08771	102

h	id	n	m
8	65	08431, 04624, 03694, 08731, 08571, 04372, 01827, 09349, 08058, 04074, 09446, 08777, 07631, 07726, 03666, 09752, 07894, 01283, 09897, 07973, 09833, 09711, 08826, 06326, 08721, 06606, 03759, 00068, 09317, 08653, 03920, 08716, 02900, 07801, 03764, 06889, 04981, 03577, 07412, 09779, 08314, 08241, 03989, 03611, 09923, 08857, 08323, 07585, 05808, 01816, 08570, 07743, 04717, 03953, 09083, 04956, 08659, 08322, 04029, 07156, 09747, 08727, 08655, 06415, 04027, 04018, 03956, 03947, 03846, 01830, 00339, 08548, 07742, 05434, 04707, 03978, 07882, 03991, 09918, 09849, 09484, 08590, 08519, 08664, 08580, 07871, 07810, 03913, 07502, 05482, 07410, 07016, 06734, 05391, 04699, 09374, 06736, 03359, 02121, 00360, 08688, 08216, 04391, 03961, 04960, 03976, 04190, 00012, 08755, 06883, 05984, 05567, 05439, 04261, 01831, 01730, 09668, 09478, 08493, 06220, 03788, 01449, 09725, 06024, 04207, 09832, 09658, 08700, 08356, 07198, 09750, 03960, 03979, 08862, 06435, 04378, 00310, 09737, 07848, 04206, 08482, 09152, 08827, 08295, 03603, 08792, 05556, 07092, 06921, 06013, 03965, 03062, 09724, 04974, 08698, 00737, 09236, 06603, 04650	4589
9	1	08000	2467
9	2	02088	2426
9	3	03540	2297
9	4	02772	2195
9	5	00679	2181
9	6	04226	2108
9	7	00025	2104
9	8	04848	2091
9	9	02644	2070
9	10	05959	2063
9	11	08997	2027
9	12	05770	2013
9	13	06503	1999
9	14	06487	1988
9	15	05315	1961
9	16	00089	1930
9	17	04407	1897
9	18	00706	1865
9	19	00814	1860
9	20	00645	1809
9	21	01302	1779
9	22	05535	1745
9	23	06491	1733
9	24	02034	1732
9	25	02124	1726
9	26	01801	1691
9	27	04340	1685

h	id	n	m
9	28	04477	1677
9	29	01043	1670
9	30	01512	1641
9	31	07669	1640
9	32	01308	1610
9	33	01840	1593
9	34	01003	1543
9	35	05419	1528
9	36	01092	1505
9	37	01634	1487
9	38	01359	1475
9	39	06596	1440
9	40	02729	1396
9	41	02130	1385
9	42	04608	1346
9	43	03267	1342
9	44	02798	1337
9	45	02437	1322
9	46	03542	1316
9	47	02855	1306
9	48	00291	1250
9	49	03260	1244
9	50	00385	1233
9	51	01154	1224
9	52	06267	1194
9	53	02149	1175
9	54	03167	1167
9	55	02089	1152
9	56	00525	1113
9	57	00739	1079
9	58	01086	1057
9	59	06971	1043
9	60	06172	1013
9	61	03664	985
9	62	04984	951
9	63	00617	914
9	64	02201	898
9	65	03084	890
9	66	05674	885
9	67	02507	872
9	68	02718	842

h	id	n	m
9	69	02376	824
9	70	01236	809
9	71	07794	795
9	72	07548	774
9	73	00841	755
9	74	00054	728
9	75	04463	717
9	76	00721	685
9	77	01811	669
9	78	05692	652
9	79	00529	641
9	80	03238	629
9	81	03517	621
9	82	08730	611
9	83	08610	593
9	84	08611	581
9	85	04926	566
9	86	01807	561
9	87	09292	555
9	88	01956	548
9	89	02624	538
9	90	02432	523
9	91	04287	518
9	92	09372	513
9	93	06292	493
9	94	06361	481
9	95	01880	471
9	96	02810	463
9	97	04818	457
9	98	01069	449
9	99	07833	442
9	100	08261	435
9	101	03400	417
9	102	02119	412
9	103	02793	403
9	104	06389	395
9	105	06368	388
9	106	08048	376
9	107	01216	358
9	108	09621	344
9	109	02375	331

h	id	n	m
9	110	02206	317
9	111	02042	308
9	112	03261	297
9	113	09469	285
9	114	01808	271
9	115	03558	255
9	116	03857	228
9	117	02390	218
9	118	02428	201
9	119	08865	188
9	120	00468	177
9	121	06087	167
9	122	07701	152
9	123	03228	141
9	124	04243	130
9	125	08474	116
9	126	03939	100
9	127	08396, 05006, 09452, 08635, 07729, 08855, 06549, 00660, 09295, 00964, 03361, 08004, 08733, 07880, 05102, 08656, 07581, 07103, 09461, 09615, 08278, 06437, 05329, 03372, 03801, 09210, 07611, 07007, 08529, 07026, 03997, 09901, 08848, 03080, 09605, 09548, 03720, 01456, 09628, 09430, 08044, 06217, 09907, 09730, 07450, 04092, 03930, 03445, 01646, 09709, 08053, 02085, 00684, 09293, 08670, 07809, 04976, 08408, 08338, 07597, 09396, 01717, 09156, 00735, 09716, 07606, 03710, 02383, 09751, 08560, 07745, 06627, 01116, 07834, 06459, 09155, 03933, 09441, 09362, 05945, 08224, 07885, 06384, 05939, 03675, 02982, 01727, 09243, 00311, 06284, 02620, 04040, 03969, 03723, 00087, 02650, 09432, 08307, 00135, 09786, 04159, 04003, 03697, 08711, 07287, 02866, 01825, 06011, 09906, 08668, 07158, 03966, 09342, 08669, 07390, 05956, 03907, 09971, 08819, 07820, 05942, 03970, 00058, 07700, 03967, 07903, 03654, 02902, 09618, 07560, 07178, 05828, 04210, 02399, 02045, 08991, 08434, 05063, 03797, 03601, 03350, 09354, 02513, 02401, 08181, 01733, 08632, 07528, 07402, 05635, 03973, 03968, 03903, 03781, 03661, 09611, 03612, 00197, 06324, 04720, 00361, 05170, 05092, 04628, 01737, 06982, 06915, 06023, 03977, 03368, 09694, 08962, 05341, 03642, 07425, 07115, 05514, 04219, 03122, 02038, 00885, 09613, 07587, 03927, 01467, 00309, 05123, 03862, 03760, 09929, 08386, 07419, 06168, 04365, 03780, 09612, 03306, 02984, 00837, 06613, 05456, 03232, 09607, 08852, 05513, 01590, 09592, 09537, 05512, 05205, 03381, 01145, 01066, 00690, 09353, 08832, 06995, 04761, 04760, 01809, 09429, 08686, 06907, 03912, 09330, 08808, 05358, 02877, 01742, 00319, 09033, 08928, 06497, 05661, 05375, 05125, 05088, 01997, 01063, 06719, 06544, 05192, 04948, 04705, 02461, 09474, 08951, 05454, 04293, 02050	7718

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 $P_{hi} = \frac{a_h \times MOS_{hi}}{MOS_h} \times \frac{m_h}{MOS_{hi}} = \frac{a_h \times m_h}{MOS_h}$.

```
linked_schools <- linked_schools |>
  group_by(Region) |>
  mutate(P_h = n_h*tot_all/M_h,
         P_i=m_h_star/tot_all,
         Prob=P_h*P_i,
         epsem_check = abs(Prob - mean(Prob)) < 0.000001)

stopifnot(all(linked_schools$epsem_check))

# check for false
linked_schools |>
  count(epsem_check) |>
  knitr::kable()
```

Region	epsem_check	n
1	TRUE	2
2	TRUE	12

Region	epsem_check	n
3	TRUE	3
4	TRUE	2
5	TRUE	35
6	TRUE	53
7	TRUE	280
8	TRUE	223
9	TRUE	376

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:

p̂ = 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?

```r
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.
```