Complex Sampling Design

Kursus R: Pengenalan dan Praktikal (Sesi 3)

Mohd Azmi Bin Suliman

azmi.suliman@moh.gov.my

Pusat Penyelidikan Penyakit Tak Berjangkit, Institut Kesihatan Umum

Friday, 04 October 2024

Complex Sampling Design

- What/Why?
- Sampling Plan
- Descriptive Analysis
- Inferential Analysis

Complex Sampling Design

Why?

- Government agencies usually conduct census to get overview picture of the population
  - → However, it is costly and time consuming.
  - → For example, the Malaysian Census (Banci) was conducted every 10 years.
- Therefore, sampling is more feasible - less costly and time consuming.
  - → However, simple random sampling may not be feasible.

Why Not SRS?

- In research involving large population or geographic area, simple random sampling (SRS) may not be feasible.
- In SRS, it is assumed that each unit has equal chance of being selected.

  $\longrightarrow$ To ensure each unit has equal chance, it is necessary to have complete list of population.

  $\longrightarrow$ In reality, even DOSM don't have the complete list of population.

- In SRS, mathematically, the sample may not adequately represent the population, particularly minorities.
- In the next slide, I'll simulate the effect of SRS in a hypothetical population.

Simulation

- Information of number of population was taken from OpenDOSM Population Table by Administrative Districts, 2023.

Link: https://open.dosm.gov.my/data-catalogue/population_district

```r
library(arrow)
pop_mydist <- read_parquet("https://storage.dosm.gov.my/population/population_

pop_my23 <- pop_mydist %>%
  filter(date == dmy("01/01/23"),
         sex != "both",
         age %in% c("20-24", "24-29", "30-34", "35-39", "40-44", "45-49",
                    "50-54", "55-59"),
         ethnicity == "overall") %>%
  mutate(population = population*10)
```

Simulation

- From the DOSM dataset, we know that, the selected population contained
  - → 16 states
  - → 160 district (Daerah Pentadbiran)
  - → Total population of approx. 16.3 million

Note: In the dataset, each 1 unit of population is 1,000 population. But in our dataset, I've multiplied by 10, so each 1 unit of population is 100 population.

```r
pop_simdist <- pop_my23 %>%
  group_by(state, district) %>%
  summarise(population = sum(population)) %>%
  ungroup()
```

Simulation

- Below is the snippet of the dataset

```r
pop_simdist %>%
  summarise(total_state = n_distinct(state),
            total_district = n_distinct(district),
            total_population = scales::comma(sum(population)))
```

```
# A tibble: 1 × 3
  total_state total_district total_population
        <int>          <int> <chr>
1          16            160 163,619
```

```r
set.seed(123)
pop_simdist %>%
  slice_sample(n = 5)
```

```
# A tibble: 5 × 3
  state           district    population
  <chr>           <chr>            <dbl>
1 W.P. Labuan     W.P. Labuan        487
2 Kedah           Kuala Muda        2536
3 Pahang          Maran              548
4 Sarawak         Lundu              162
5 Negeri Sembilan Tampin             404
```

Simulation

- In this simulation, we will randomly select 100 units from the population.
- Will all districts have representative?
- Below is total number of district that have the populations included in sample.

```r
set.seed(123)
pop_simdist %>%
  slice_sample(n = 100) %>%
  count(state, district) %>%
  count(state) %>%
  rename(sample_district = n)
```

Simulation

```r
set.seed(123)
pop_simdist %>%
  slice_sample(n = 100) %>%
  count(state, district) %>%
  count(state) %>%
  rename(sample_district = n)
```

```
# A tibble: 15 × 2
   state                sample_district
   <chr>                          <int>
 1 Johor                              4
 2 Kedah                              8
 3 Kelantan                           8
 4 Melaka                             2
 5 Negeri Sembilan                    6
 6 Pahang                             7
 7 Perak                              5
 8 Pulau Pinang                       3
 9 Sabah                             19
10 Sarawak                           21
11 Selangor                           7
12 Terengganu                         7
13 W.P. Kuala Lumpur                  1
14 W.P. Labuan                        1
15 W.P. Putrajaya                     1
```

Simulation

- Compare that with the total district in Malaysia

```r
sel_dist <- pop_simdist %>%
  slice_sample(n = 100) %>%
  count(state, district) %>%
  count(state) %>%
  rename(sample_district = n)

all_dist <- pop_simdist %>%
  count(state, district) %>%
  count(state) %>%
  rename(all_district = n)

sel_dist %>%
  left_join(all_dist, by = "state") %>%
  mutate(perc = scales::percent(sample_district / all_district))
```

Simulation

- As you can see below, with random selection, not all district will be represented.
- We can imagine how these random selection will affect other selection criteria, e.g., gender, age, etc.

```
# A tibble: 14 × 4
   state              sample_district all_district perc  
   <chr>                        <int>        <int> <chr> 
 1 Johor                            7           10 70.0% 
 2 Kedah                            6           12 50.0% 
 3 Kelantan                         9           11 81.8% 
 4 Melaka                           1            3 33.3% 
 5 Negeri Sembilan                  3            7 42.9% 
 6 Pahang                           8           11 72.7% 
 7 Perak                            7           13 53.8% 
 8 Pulau Pinang                     2            5 40.0% 
 9 Sabah                           18           27 66.7% 
10 Sarawak                         25           40 62.5% 
11 Selangor                         5            9 55.6% 
12 Terengganu                       7            8 87.5% 
13 W.P. Kuala Lumpur                1            1 100.0%
14 W.P. Labuan                      1            1 100.0%
```

Why Not SRS?

- Stratification is a method where the population is divided into strata
  - → For example, the population is divided by gender and ethnicity
  - → Stratification ensures that important subgroups are adequately represented in the sample

- Clustering is a method where the population is divided into clusters
  - → For example, an enumeration block, in which the population is divided by geographical area
  - → Clustering can increase efficiency and reduce cost, especially in geographical dispersed population

Complex Sampling Design in NHMS

- However, clustering and stratification affect the sampling probability.
- Thus the need for complex sampling design.

- Our NHMS apply complex sampling design.
- This is snippet from our NHMS 2022 MCH

> The survey employed a two-stage stratified random sampling design covering all states and federal territories in Malaysia for national representation. States and territories formed the primary stratum, while urban and rural areas were the secondary stratum. DOSM randomly selected Enumeration Blocks (PSUs) and living quarters (SSUs) based on sample size.

Sampling Weight

- In complex sampling design, each unit may have different probability of being selected.
- Therefore, each unit is assigned a weight.
- To correctly estimate the population parameter, the weight must be taken into account.
- This is snippet from our NHMS 2022 MCH

> A weight factor was applied to each individual to adjust for the varying probabilities of selection (design weight), non-response rate, and post-stratification factors which were adjusted for the Malaysian population projection by DOSM in the year 2022.

Sampling Weight NHMS

- The weight is calculated as follows:
  - $\rightarrow$ W1 = the inverse probability of selecting the EBs of each state
  - $\rightarrow$ W2 = the inverse probability of selecting the LQs over the total frame during the listing activities
  - $\rightarrow$ W3 = the inverse probability of successful LQ from the total listing frame
  - $\rightarrow$ F = the non-response adjustment factor for LQ and individual
  - $\rightarrow$ PS = post-stratification adjustment factor calculated by state, gender and ethnicity

Practical!

- Setting up complex sampling design plan
- Descriptive analysis
- Inferenial analysis

Setting Up (Create New Project & Quarto Notebook)

1. Open RStudio
2. Create New Project
    - `File` > `New Project` > `New Directory` > `New Project`
3. Set the name and directory.
    - Name: `Complex Sampling Design`
    - Directory: `.../Documents/RStudio`
4. Click `Create Project`
5. Create a new Quarto document
    - `File` > `New File` > `Quarto Document...`
6. Set the title
    - Title: `Complex Sampling Design`
    - Untick `Use the visual editor`
7. Click `Create Empty Document`
8. Edit the YAML

Setting Up (Load Libraries)

- Add Level 1 Header

  → `# Setup, Dataset and Library`

- Add setup code chunk

~~~
```{r}
#| label: setup

library(tidyverse)
library(haven)
library(survey)
```
~~~

Data Import

- Copy the simulation dataset in our working directory
- We can put our import data code in setup chunk

```r
selsimdmds <- read_sav("Dataset/SelSimDataClean.sav") %>%
  as_factor()

head(selsimdmds)
```

```
# A tibble: 6 × 22
     id district gender   age agegp ethnicity height weight   bmi PAhour hba1c
  <dbl> <fct>    <fct>  <dbl> <fct> <fct>      <dbl>  <dbl> <dbl>  <dbl> <dbl>
1     1 Gombak   Female    55 50-59 Malay        164   75.8  28.2      5   8  
2     2 Gombak   Male      57 50-59 Malay        169   91.6  32.1      4   7.4
3     4 Gombak   Female    50 50-59 Chinese      165   71.7  26.3      5   6.1
4     5 Gombak   Male      37 30-39 Malay        164   68.4  25.4      5   6.3
5     6 Gombak   Male      35 30-39 Chinese      160   60.2  23.5      2   4.7
6     7 Gombak   Female    38 30-39 Malay        162   63.6  24.2      2   5.4
# ℹ 11 more variables: dm_status <fct>, pop_d <dbl>, sampnum_d <dbl>, W1 <dbl>,
#   W2 <dbl>, totresp_d <dbl>, NRF <dbl>, totresp_ps <dbl>, pop_ps <dbl>,
#   PS <dbl>, final_weight <dbl>
```

Data Exploratory

- Add Level 1 Header
  - → `# Data Exploratory`
- In this dataset, there were several variables related to the outcome, namely
  - → glucose control level: `hba1c`
  - → **Diabetes Status: `dm_status`**

```
names(selsimdmds)
```

```
 [1] "id"             "district"       "gender"         "age"            "agegp"
 [6] "ethnicity"      "height"         "weight"         "bmi"            "PAhour"
[11] "hba1c"          "dm_status"      "pop_d"          "sampnum_d"      "W1"
[16] "W2"             "totresp_d"      "NRF"            "totresp_ps"     "pop_ps"
[21] "PS"             "final_weight"
```

Setting Up Complex Sampling Design

- While R able to recognize factor level, but based on my short personal experience with R, the analysis is simpler if we convert the outcome to binary `0` and `1` form.
  - → We can update our setup chunk code

```r
selsimdmds <- read_sav("Dataset/SelSimDataClean.sav") %>%
  as_factor() %>%
  mutate(dm_statusN = as.numeric(dm_status) - 1,
         .after = dm_status)

head(selsimdmds)
```

```
# A tibble: 6 × 23
     id district gender   age agegp ethnicity height weight   bmi PAhour hba1c
  <dbl> <fct>    <fct>  <dbl> <fct> <fct>      <dbl>  <dbl> <dbl>  <dbl> <dbl>
1     1 Gombak   Female    55 50-59 Malay        164   75.8  28.2      5   8
2     2 Gombak   Male      57 50-59 Malay        169   91.6  32.1      4   7.4
3     4 Gombak   Female    50 50-59 Chinese      165   71.7  26.3      5   6.1
4     5 Gombak   Male      37 30-39 Malay        164   68.4  25.4      5   6.3
5     6 Gombak   Male      35 30-39 Chinese      160   60.2  23.5      2   4.7
6     7 Gombak   Female    38 30-39 Malay        162   63.6  24.2      2   5.4
# ℹ 12 more variables: dm_status <fct>, dm_statusN <dbl>, pop_d <dbl>,
```

Setting Up Complex Sampling Design

- While R able to recognize factor level, but based on my short personal experience with R, the analysis is simpler if we convert the outcome to binary `0` and `1` form.
  - → Hint: we can use cross-tab to check our transformation

```r
with(selsimdmds, table(dm_status, dm_statusN))
```

```
          dm_statusN
dm_status   0  1
  Normal   84  0
  Diabetes  0 74
```

Setting Up Complex Sampling Design

- Add Level 1 Header

  → `# Complex Sampling Design Plan`

- Specifying the complex sampling design is the first step in analysing complex sampling design data.
- We have to check back with the original research team, how was the study was designed.

Setting Up Complex Sampling Design

- This is the snippet from NHMS 2019 (this dataset)

> To ensure national representativeness, two stage stratified random sampling was used. The **two strata are primary stratum, which made up of states of Malaysia, including Federal Territories, and secondary stratum, which made up of urban and rural strata formed within the primary stratum**. Sampling involved two stages; **the Primary Sampling Unit (PSU), which were the EBs** and the Secondary Sampling Unit (SSU), which were the LQs within the selected EBs.

Setting Up Complex Sampling Design

- These are information required for our complex sampling design

```r
?svydesign
```

Required Information/Specification	**Common NHMS Variable Name**
Cluster IDs (PSU)	EB ID
Strata	State.Strata, State.wt
Sampling Weight	ADW, weight_final, weight

Setting Up Complex Sampling Design

- We can use `svydesign` function to specify the complex sampling design
  - → First, setup the **unweighted** design

```r
selsim_unwdsg <- svydesign(ids = ~1,
                           weights = ~1,
                           data = selsimdmds)
```

Setting Up Complex Sampling Design

- We can use `svydesign` function to specify the complex sampling design
  - → Then, setup the **weighted** design

```r
selsim_svydsg <- svydesign(ids = ~1,
                           weights = ~final_weight,
                           data = selsimdmds,
                           nest = T)
```

Setting Up Complex Sampling Design

- We can use `svydesign` function to specify the complex sampling design

    → We can use `summary( )` to check the design

```r
summary(selsim_svydsg)
```

```
Independent Sampling design (with replacement)
svydesign(ids = ~1, weights = ~final_weight, data = selsimdmds,
    nest = T)
Probabilities:
     Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
8.178e-06 7.805e-05 2.062e-04 5.818e-04 7.031e-04 6.000e-03
Data variables:
 [1] "id"            "district"     "gender"       "age"          "agegp"
 [6] "ethnicity"     "height"       "weight"       "bmi"          "PAhour"
[11] "hba1c"         "dm_status"    "dm_statusN"   "pop_d"        "sampnum_d"
[16] "W1"            "W2"           "totresp_d"    "NRF"          "totresp_ps"
[21] "pop_ps"        "PS"           "final_weight"
```

Complex Sampling Design Analysis

- Descriptive Analysis

Descriptive Analysis

- Unweighted Count
- Estimated Population
- Prevalence and Confident Interval
- Subpopulation

Descriptive Analysis

- Add Level 1 Header
  - → `# Descriptive Analysis`
- For descriptive analysis, we can refer our NHMS report for reporting.
- There were several parameter that reported, including
  - → Unweighted Count
  - → Estimated Population
  - → Prevalence and Confident Interval

Practical: Unweighted Count

- For unweighted parameter, we use unweighted design.
- For count, `svytotal( )` is used.
  - → For this example, the unweighted count for overall is calculated

```r
svytotal(x = ~dm_statusN,
         design = selsim_unwdsg,
         na.rm = T)
```

```
           total     SE
dm_statusN    74 6.2922
```

Practical: Estimated Population

- For estimated population, we use weighted design.
- For count, `svytotal( )` is used.
  → For this example, the estimated population for overall is calculated

```r
svytotal(x = ~dm_statusN,
         design = selsim_svydsg,
         na.rm = T)
```

```
            total     SE
dm_statusN 740866 132739
```

Practical: Prevalence

- For prevalence
  - → Weighted design is used
  - → `svymean( )` is used.
  - → For this example, the prevalence and confident interval for overall is calculated

```r
svymean(x = ~dm_statusN,
        design = selsim_svydsg,
        na.rm = T)
```

```
              mean     SE
dm_statusN 0.39264 0.0674
```

Practical: Confident Interva for Prevalence

- For confident interval for prevalence
  - → Weighted design is used
  - → `svyciprop( )` is used.

```r
svyciprop(formula = ~dm_statusN,
          design = selsim_svydsg) %>%
  attr(., "ci")
```

```
    2.5%    97.5% 
0.270001 0.530504 
```

Practical: Subpopulation (Unweighted Count)

- For subpopulation analysis, we can use `svyby( )` function
- For example, we want to calculate the unweighted count, by locality

```r
svyby(formula = ~dm_statusN,
      by = ~gender,
      design = selsim_unwdsg,
      FUN = svytotal,
      na.rm.all = T)
```

```
       gender dm_statusN       se
Male     Male         44 5.652349
Female Female         30 4.945564
```

Practical: Subpopulation (Estimated Population)

- For subpopulation analysis, we can use `svyby( )` function
- For example, we want to calculate the estimated population, by locality

```r
svyby(formula = ~dm_statusN,
      by = ~gender,
      design = selsim_svydsg,
      FUN = svytotal,
      na.rm.all = T)
```

```
       gender dm_statusN       se
Male     Male   416090.8 105104.5
Female Female   324774.8  91072.9
```

Practical: Subpopulation (Prevalence)

- For subpopulation analysis, we can use `svyby( )` function
- For example, we want to calculate the prevalence, by locality

```r
svyby(formula = ~dm_statusN,
      by = ~gender,
      design = selsim_svydsg,
      FUN = svymean,
      na.rm.all = T)
```

```
       gender dm_statusN         se
Male     Male  0.4416367 0.10235204
Female Female  0.3437781 0.08827912
```

Practical: Mean for Continuous Variable

- We can calculat ethe mean for continuous variable using `svymean( )` function
  - → for unweighted parameter, use unweighted design
  - → and for population parameter, use weighted design

```r
svymean(x = ~hba1c,
        design = selsim_unwdsg,
        na.rm = T)
```

```
        mean     SE
hba1c 6.462 0.1158
```

```r
svymean(x = ~hba1c,
        design = selsim_svydsg,
        na.rm = T)
```

```
         mean    SE
hba1c 6.4256 0.165
```

Practical: Mean for Continuous Variable

- We can calculate the mean for continuous variable using `svymean( )` function

  → use `confint( )` to calculate the confident interval

```
svymean(x = ~hba1c,
        design = selsim_unwdsg,
        na.rm = T) %>%
  confint()
```

```
          2.5 %   97.5 %
hba1c 6.235037 6.689014
```

```
svymean(x = ~hba1c,
        design = selsim_svydsg,
        na.rm = T) %>%
  confint()
```

```
          2.5 %   97.5 %
hba1c 6.102196 6.748947
```