R Training Book for IKU

Table of contents

Preface

Objectives

The Institute for Public Health (IPH) (Malay: Institut Kesihatan Umum, IKU) is a research institution under the Ministry of Health Malaysia, primarily focusing on public health research. In its daily activities, software like SPSS and STATA plays a crucial role in data analysis. However, using these softwares results in significant operational costs for the institute due to the purchase of software licenses. Recognising this issue, IKU is committed to transitioning towards using open-source and free software such as R and Python. This shift reduces cost burdens and empowers IKU staff with more flexible and advanced tools for data analysis.

R is a practical programming language for statistical analysis and graphics production. Its open-source and free nature makes it the preferred choice for research in public health. Through this book, it is hoped that the data analysis skills among IKU staff will be enhanced, leading to improvements in the quality of IKU's research.

Way Forward

R offers capabilities that extend well beyond statistical analysis. As more IKU staff become proficient in R, we anticipate leveraging R's diverse project capabilities to benefit IKU significantly:

1. Shiny: Develop interactive dashboards for dynamic and near-real-time result presentation.
2. Quarto: Utilize this publishing system for expedited reports and paper production.
3. IKU-specific R packages: Create tailored R packages incorporating functions for tasks such as sample size calculation, importing data from REDCap via API, standardising analysis of NHMS data, and uniform reporting of NHMS findings.

This forward-looking approach aims to harness R's full potential to streamline and enhance IKU's research and reporting processes, making them more efficient and impactful.

Collaborators Are Welcome

In the spirit of open science and continuous improvement, individuals both within and outside IPH are invites for collaboration. Whether one is an author with insights to share, an editor with an eye for detail, or possesses constructive suggestions, these contribution can significantly enhance the utility and reach of this manual. It is particularly interested in contributions in the following areas:

- Content Enhancement: The addition of new chapters or sections covering unexplored areas of R, the introduction of advanced statistical techniques, or the expansion on the applications of R in public health research are welcomed.
- Technical Review: Contributors can help ensure the accuracy of code examples, update or optimize R scripts, and contribute towards a repository of R functions tailored for public health data analysis.
- Case Studies: The IPH appreciates the sharing of real-world applications of R in public health, especially those within the context of IKU's research projects. This could include case studies on data visualization, statistical analysis, or the development of interactive applications with Shiny.
- Educational Materials: There is a need for developing tutorials, exercises, or additional learning resources that complement the manual's content, thereby facilitating a deeper understanding of R programming among IPH staff.

How to Contribute:

Individuals interested in contributing or who have suggestions to improve this manual are encouraged not to hesitate in reaching out. Your input is invaluable in making this resource more comprehensive, accurate, and beneficial for all users.

Contact Information:

Ideas, proposals for collaboration, or any feedback should be emailed to Mohd Azmi Bin Suliman at the Centre for Non-communicable Diseases Research (CNCDR). The institute looks forward to hearing from contributors and exploring how collaboration can further advance public health research through the power of R programming.

Mohd Azmi Bin Suliman
Centre for Non-communicable Diseases Research (CNCDR)
February 2024

1 Introduction

1.1 R

1.2 RStudio

1.3 Quarto

2 Data Wrangling

In summary, this book has no content whatsoever.

```
1 + 1
```

```
[1] 2
```

3 Data Wrangling

In summary, this book has no content whatsoever.

```
1 + 1
```

```
[1] 2
```

4 Data Wrangling

In summary, this book has no content whatsoever.

```
1 + 1
```

```
[1] 2
```

5 Complex Sampling Design in NHMS

5.1 Why Complex Sampling Design?

Surveys are essential for understanding population characteristics, offering a more efficient and resource-friendly alternative to censuses. Censuses, aiming to collect data from every individual within a population, are historically resource-intensive. In contrast, surveys, whether conducted by governments or researchers, enable effective population inferences with less expenditure.

Simple random sampling, while a traditional gold standard for its straightforward approach and unbiased estimates, often falls short in achieving comprehensive representativeness, particularly in diverse populations. This limitation becomes apparent in the context of the National Health and Morbidity Survey (NHMS), where both national and state-level representativeness are crucial. Simple random sampling might not adequately represent all geographic areas, especially when population densities and distributions vary significantly across different states. This could lead to over representation of more populous areas while leaving less populous regions under-represented.

Furthermore, this sampling method might not effectively capture the diversity within minority groups, as their smaller numbers in the overall population reduce the likelihood of their selection in a simple random sample. To overcome these challenges, NHMS employs more intricate sampling designs like stratified sampling. By dividing the population into distinct strata based on states or regions, and further considering sub-groups within these strata, it ensures that both geographic areas and minority groups are appropriately represented. Although these complex sampling designs introduce potential biases in selection probabilities and are more challenging to implement, they are indispensable for achieving the depth of representativeness required for national health assessments and policy planning.

One of the significant advantages of complex sampling designs is their feasibility without a comprehensive population list, focusing instead on broader stratifications like specific localities, simplifying the sampling process.

5.1.1 Benefits of Complex Sampling Design

The National Health and Morbidity Survey (NHMS), conducted by the Institut Kesihatan Umum (IKU), benefits extensively from complex sampling designs, showcasing several advantages:

1. Cost Efficiency: By clustering samples within selected strata or areas, operational costs are notably reduced, obviating the need to cover extensive and potentially scattered geographical locations.
2. Enhanced Representativeness: Stratification techniques ensure the sample accurately reflects specific subgroups or geographic areas, improving the survey's overall representativeness and reliability.
3. Data Analysis Advantages: Complex sampling designs facilitate the adjustment of sampling weights, enabling the generation of accurate national or state-level estimates. Furthermore, they support comprehensive subgroup analyses, ensuring sufficient statistical power.

5.1.2 Challenges in Implementing Complex Sampling Design

Despite their benefits, complex sampling designs require meticulous planning and sophisticated analytical techniques. These designs necessitate accounting for factors like clustering and weighting, demanding specialised expertise for both the sample's design and subsequent data analysis.

5.1.3 Example: Sampling Probability of a Sabahan

Problem: Consider a hypothetical scenario within a diverse group of 100 people, composed of 60% Malay, 20% Chinese, 15% Indian, and an additional 5% from other ethnic backgrounds, including 1% Sabahan. How sure are we, than when we randomly select 10 people from the group, at least one of the 10 people will be a Sabahan?

Answer: To calculate the probability of selecting at least one Sabahan in a 10-person sample, one might initially consider the likelihood of not choosing a Sabahan and subtract this figure from 1. With 99 of the 100 individuals not being Sabahan, the probability of not selecting a Sabahan in a single attempt is 99/100. Over 10 independent selections, this probability becomes (99/100)^10. Consequently, the probability of selecting at least one Sabahan is 1 - (99/100)^10, equating to approximately 9.56%. This calculation suggests a close to 10% chance that the sample will include at least one Sabahan.

Or in other word, since minorities were in fact had lower percentage, when we sample our population, we might even did not get the minorities in our sample!.

5.2 Practical

In complex survey analysis using the `survey::` package in R, it's crucial to account for the design aspects of the survey beyond just the outcome variables and covariates. This includes specifying:

Table 5.1: Required Information for Complex Sampling Design

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

5.2.1 Setup Project

1. Setup your project
2. Copy the NHMS dataset into the working directory
3. Create Quarto document
    - update the YAML metadata to make the document self-contained

```yaml
---
title: "Sesi 4 - NHMS"
format:
  html:
    embed-resources: true
---
```

5.2.2 Analysis

5.2.2.1 Setup

0. Understand the dataset context
    - In this practical, the example was shown using NHMS NCD 2019's cholesterol dataset.
    - Two outcome will be selected
        - Categorical Type: known hypercholesterolaemia status (column `known_chol`)
        - Numerical Type: capillary total cholesterol level (column `u303`)

1. Import Dataset
    - On the `Files` pane, click on the spps .sav file
    - Select `Import Dataset ...`
    - Copy the code into the r code chunk
    - add function `as_factor(_)` to convert labelled code

```r
#| output: false

library(tidyverse)
library(haven)

nhms19ds <- read_sav("nhms19ds.sav") %>%
  as_factor()

nhms19ds
```

> **Note**
>
> there are 40 columns in the dataset, hence the dataset is not shown here.

2. Briefly (or in detail, up to you), explore the dataset.
    - Identify the outcome variable
        - data type: numerical, character or factor?
        - any missing data
    - Identify the complex sampling related variable:
        - the cluster ids
        - the strata
        - the sampling weight

> **Tip**
>
> some packages and functions that offer a quick data exploration:
> - `skimr::` package: `skim(_)` function.
> - `summarytools::` package: `dfSummary(_)` function.

```r
#| eval: false

library(skimr)

nhms19ds %>%
  select(known_chol, u303) %>%
  skim()
```

Variable Name	Variable Label	Variable Name	Variable Label
state	[Final] State	c03a	years since was told to have high cholesterol
strata_gp	[Final] Locality	c04a	on medication for past 2 week
A2101	[Final] Gender	c04b	advice for special low fat diet
A2104	Age (Numerical)	c04c	advice to loose weight
A2104_grp	[Final] Age Group - 16 groups	c04d	advice to exercise
A2106_5grp	Ethnicity (5 groups)	c05	treatment - herbal/TCM
A2107	Citizenship	c06	common place to receive treatment
A2108_3grp	[Final] Marital Status (3 groups)	u303	Total Cholesterol (mmol/L)
A2109_4grp	[Final] Highest Education Level (5 groups)	known_chol	_no label_
A2221	If working, type of occupation	undiagnosed_chol	_no label_
A2222_7grp	Employement status (7 groups)	total_chol	_no label_
A2222_5grp	[Final] Occupation (5 groups)	bodyweight1	Body Weight (kg)
indvid	_no label_	bodyweight2	Body Weight (kg)
hh_id	_no label_	bodyheight1	Body Height (cm)
state_st	PSU	bodyheight2	Body Height (cm)
ebid	EB ID - Cluster	wc2	Waist Circumference (cm)
wtfinal_ncd	Sampling Weight	wc1	Waist Circumference (cm)
c01	ever had total blood cholesterol level measured	weight	Body Weight (kg)
c02	ever told have high cholesterol level	height	Body Height (cm)
c03	when told to have high cholesterol	wc	Waist Circumference (cm)

NHMS NCD 2019 - Cholesterol Module Dataset: Variables List

Table 5.2: Data summary

Name	Piped data
Number of rows	10472
Number of columns	2
Column type frequency:	
factor	2
Group variables	None

Variable type: factor

skim_variable	n_missing	complete_rate	ordered	n_unique	top_counts
known_chol	6	1.00	FALSE	2	No: 8451, Yes: 2015, N/A: 0
u303	594	0.94	FALSE	87	5: 448, 5.1: 383, 4.8: 378, 4.3: 373

Warning

- there are missing values in the outcome variable `known_chol`. while is it not a must to remove sample with no outcome, as the analysis will automatic remove sample with no outcome using `na.rm = T` parameter, it is advisable to remove any sample that do not have the outcome.
- the outcome variable of capillary total cholesterol was in categorical type. we need to convert it to numerical type

Tip

later in complex sampling design analysis, the analysis accept the variable outcome (i.e. the `known_chol`) variable in either numeric or factor type. but binary type is preferable

3. In this practical we will make some data wrangling
    - remove missing outcome
    - transform factor type to numerical binary type

```
  ```{r}
nhms19ds <- nhms19ds %>%
    as_factor() %>%
    filter(!is.na(known_chol)) %>%
    mutate(known_cholN = as.numeric(known_chol)-2,
           u303 = as.numeric(as.character(u303)))
  ```
```

> **Note**
>
> The variable `known_col` have there levels, which can be check using `levels(_)` function: `levels(nhms19ds$known_chol)`. When converted to numeric using `as.numeric(_)` function, the `known_chol` value was either 1 (correspond to NA), 2 (correspond to No) and 3 (correspond to Yes), thus the value need to minus 2, so that `No` is correspond to value `0` and `Yes` is correspond with value 1.
>
> the conversion can be check by looking at both the variable
>
> ```
> ```{r}
> #| eval: false
>
> nhms19ds %>%
>   select(known_chol, known_cholN)
> ```
> ```

4. Specifying the Complex Sampling Design
    - Add options at the top of Quarto file
    - These option is to handle in which if there is single PSU within strata or domains

```
```{r}
library(survey)

options(survey.lonely.psu = 'adjust',
        survey.adjust.domain.lonely = TRUE)
```
```

    - Unweighted Design
        - cluster ids set as 1 (i.e., no clustering)
        - weight as 1 (i.e., same probability)

```
```{r}
nhms_unwdsg <- svydesign(id = ~1,
                         weights = ~1,
                         data = nhms19ds)
```
```

- we can use function `summary(_)` to view our complex sample design

```
summary(nhms_unwdsg)
```

```
Independent Sampling design (with replacement)
svydesign(id = ~1, weights = ~1, data = nhms19ds)
Probabilities:
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
      1       1       1       1       1       1
Data variables:
 [1] "state"         "strata_gp"      "A2101"         "A2104"
 [5] "A2104_grp"     "A2106_5grp"     "A2107"         "A2108_3grp"
 [9] "A2109_4grp"    "A2221"         "A2222_7grp"     "A2222_5grp"
[13] "indvid"        "hh_id"         "state_st"       "ebid"
[17] "wtfinal_ncd"     "c01"           "c02"           "c03"
[21] "c03a"          "c04a"          "c04b"           "c04c"
[25] "c04d"          "c05"           "c06"            "u303"
[29] "known_chol"    "undiagnosed_chol" "total_chol"     "bodyweight1"
[33] "bodyweight2"    "bodyheight1"    "bodyheight2"    "wc2"
[37] "wc1"           "weight"         "height"         "wc"
[41] "known_cholN"
```

- in unweighted design, the probability for sample range from 1 to 1.
- Weighted Design
  - cluster `id` set as the PSU (commonly the variable `ebid`)
  - `strata` set as the stratification. since most NHMS applied two stage of stratification, the strata must include both 1st stage and 2nd stage (commonly the variable `state_st`)
  - `weights` set as the sampling weight
  - Note that parameter `nest = T` to ensure that the cluster is nested within the specified strata

```
```{r}
nhms_surdsg <- svydesign(id = ~ebid,
                         strata = ~state_st,
                         weights = ~wtfinal_ncd,
                         data = nhms19ds,
                         nest = T)
```
```

- we can use function `summary(_)` to view our complex sample design

```
```{r}
options(width = 70) # the output width limit

summary(nhms_surdsg)
```
```

```
Stratified 1 - level Cluster Sampling design (with replacement)
With (475) clusters.
svydesign(id = ~ebid, strata = ~state_st, weights = ~wtfinal_ncd,
    data = nhms19ds, nest = T)
Probabilities:
     Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
1.405e-05 3.608e-04 7.000e-04 2.850e-03 2.000e-03 1.200e-01
Stratum Sizes:
            1   2   3   4   5   6   7   8   9  10  11  12  13  14
obs       584 274 281 263 281 307 331 319 294 245 302 338 307 333
design.PSU  27  13  13  11  12  12  12  12  12  12  12  12  14  12
actual.PSU  27  13  13  11  12  12  12  12  12  12  12  12  14  12
           15  16  17  18  19  20  21  22  23  24  25  26  27  28 29
obs       317 265 258 294 898 224 301 341 405 429 388 358 504 420 99
design.PSU  16  11  12  12  53  11  11  13  20  19  16  14  25  19  4
actual.PSU  16  11  12  12  53  11  11  13  20  19  16  14  25  19  4
            30
obs        506
design.PSU  33
actual.PSU  33
Data variables:
 [1] "state"            "strata_gp"        "A2101"
 [4] "A2104"            "A2104_grp"        "A2106_5grp"
 [7] "A2107"            "A2108_3grp"       "A2109_4grp"
[10] "A2221"            "A2222_7grp"       "A2222_5grp"
[13] "indvid"           "hh_id"            "state_st"
[16] "ebid"             "wtfinal_ncd"      "c01"
[19] "c02"              "c03"              "c03a"
[22] "c04a"             "c04b"             "c04c"
[25] "c04d"             "c05"              "c06"
[28] "u303"            "known_chol"      "undiagnosed_chol"
[31] "total_chol"       "bodyweight1"      "bodyweight2"
[34] "bodyheight1"      "bodyheight2"      "wc2"
[37] "wc1"              "weight"           "height"
[40] "wc"               "known_cholN"
```

- in weighted design summary, several info were given
  - the sampling probabilities. in this dataset, each of the sample have probability from 0.00001 to 0.12
  - the number of strata, number of sample in each of the strata and number of PSU (EB) in each strata. in this dataset, there are total 30 strata (13 states + 3 federal territories, with each state have 2 locality urban and rural).

5.2.2.2 Count the unweighted sample

1. To count the number of sample, we will use function `svymean(_)` from `survey::`.
    - the outcome variable can be either factor type, or if it in numerical type, it must be binary 0-1 number.
    - to estimate the number of sample, we will use the unweighed design.
    - the `x =` parameter must be in formula form with ~ (tilde) symbol before the variable name, i.e. `~known_chol`.
2. this is if we want to use the original factor type.

````
```{r}
svytotal(x = ~known_chol,
         design = nhms_unwdsg,
         na.rm = T)
```
````

```
              total     SE
known_cholN/A     0  0.000
known_cholNo   8451 40.339
known_cholYes  2015 40.339
```

3. this is if we want to use the converted to binary 0-1 numerical type. noticed the output differences.

````
```{r}
svytotal(x = ~known_cholN,
         design = nhms_unwdsg,
         na.rm = T)
```
````

```
            total     SE
known_cholN  2015 40.339
```

> **Note**
>
> Note 1: noticed that parameter `na.rm =` were set as T (TRUE). this is so that any sample with missing at parameter `x =` (i.e. the `known_chol`) will be removed.
>
> Note 2: From this point forward, I'll use `known_cholN` variable (the binary 0-1 numerical type) as the outcome. You are feel free to use the original factor type, and explore as you wish.

5.2.2.3 Estimating the estimated population

1. to estimate total number of population that have the outcome (i.e., `known_cholN`), same formula is used, with changes at the design used, i.e. the weighted design

```
```{r}
svytotal(x = ~known_cholN,
         design = nhms_surdsg,
         na.rm = T)
```
```

```
              total     SE
known_cholN 2868124 103013
```

5.2.2.4 Estimating Prevalence

0. Estimating the prevalence using the function of `svymean(_)` from `survey` `::` package.
    - if the outcome variable is factor type, both original factor type and converted numerical type can be used.
        - if original factor type is used, prevalence for both `No` and `Yes` will be estimated.
        - if the outcome have three or more levels, using original factor type is preferable.
        - when using the binary 0-1 numerical type (i.e., the `known_cholN`), `svymean(_)` will calculate prevalence by calculating how many `1` since `0` does not have value.

1. Using function `svymean(_)` to calculate

```
```{r}
svymean(x = ~known_cholN,
        design = nhms_surdsg,
        na.rm = T)
```
```

```
               mean     SE
known_cholN 0.13479 0.0051
```

5.2.2.5 Estimating Confidence Interval for Prevalence

1. To calculate the confidence interval for prevalence, function `svyciprop(_)` from package `survey` `::` will be used.
    - Generally, a generic function `confint(_)` can be used to calculate the confident interval for model parameter.
    - In R however, the function will treat proportion as mean of binary outcomes. While treating proportion as mean of binary outcomes is reasonable accepted to calculate the prevalence, however, when calculate the CI, it is preferable to treat apply logit transformation and transformed back to the original scale
    - the default method used in `svyciprop(_)` function is "logit"
    - however, to replicate result from SPSS and SUDAAN, the method parameter need to change to "xlogit"

```
  ```{r}
svyciprop(formula = ~known_cholN,
          design = nhms_surdsg,
          method = "xl") %>%
  attr(., "ci")
  ```
```

```
     2.5%     97.5%
0.1251425 0.1450549
```

> **Note**
>
> - function `attr(_)` is used to pull the attribute from the object (i.e., the output of the `svyciprop(_)` function), while the parameter `"ci"` in the `attr(_, "ci")` function is to pull the CI from the `svyciprop(_)`

5.2.2.6 Estimating the Unweighted Sample Proportion

Can you calculated the sample proportion using the same function?.

> **Tip**
>
> Hint:
>
> 1. Sample Proportion = Unweighted Proportion.
> 2. Unweighted design vs. Weighted design.

5.2.2.7 Estimating by Subpopulation

1. To estimates by subpopulation, we use `svyby(_)` function
2. Estimating the unweighted count by locality (urban vs rual)
    - Don't forget to use the unweighted design

```
  ```{r}
  svyby(formula = ~known_cholN,
        by = ~strata_gp,
        design = nhms_unwdsg,
        FUN = svytotal,
        na.rm.all = T)
  ```
```

```
      strata_gp known_cholN       se
Urban     Urban        1198 32.57255
Rural     Rural         817 27.44622
```

3. Estimating the estimated population by locality (urban vs rual)

```
  ```{r}
  svyby(formula = ~known_cholN,
        by = ~strata_gp,
        design = nhms_surdsg,
        FUN = svytotal,
        na.rm.all = T)
  ```
```

```
      strata_gp known_cholN       se
Urban     Urban   2282784.1 97025.32
Rural     Rural    585339.9 34607.55
```

4. Estimating the prevalence by locality (urban vs rual)

```
  ```{r}
  svyby(formula = ~known_cholN,
        by = ~strata_gp,
        design = nhms_unwdsg,
        FUN = svymean,
        na.rm.all = T)
  ```
```

```
      strata_gp known_cholN          se
Urban     Urban   0.1878626 0.004891561
Rural     Rural   0.1998044 0.006253350
```

5. Estimating the prevalence CI by locality (urban vs rual).
    - unfortunately, `svyciprop(_)` can't be used with `svyby(_)` function.
    - to estimate the CI, we need to subset the sample, to only the sub-population.

> **Warning**
>
> This however, will affect the degree of freedom (df). thus, we need to specified the df in the subset analysis, using the df of the overall design. to achieve this, add parameter `df = degf(design)`, where the design is the overall design

```
  ```{r}
  nhms_surdsg_urban <- subset(nhms_surdsg,
                              strata_gp == "Urban")

  svyciprop(formula = ~known_cholN,
            design = nhms_surdsg_urban,
            method = "xl",
            df = degf(nhms_surdsg)) %>%
    attr(., "ci")
  ```
```

```
     2.5%     97.5%
0.1258899 0.1498218
```

alternatively, we can create custom function
(the custom function code is shown next page)

```
  ```{r}
  svyciprop_by(x = ~known_cholN,
               by = ~strata_gp,
               design = nhms_surdsg,
               df = degf(nhms_surdsg),
               method = "xl")
  ```
```

```
  subset   ci.2.5.  ci.97.5.
1  Urban 0.1258899 0.1498218
2  Rural 0.1103945 0.1421803
```

the custom function code

```
```{r}
# create a svyby-like function specific for svyciprop
svyciprop_by <- function(x, by, design,
                         df = NULL, method = NULL) {
  # extract the levels in by
  by_var <- all.vars(by)[1]
  by_data <- model.frame(by, data = design$variables)
  by_levels <- sort(unique(by_data[[by_var]]))

  # run the svyciprop() functions on each levels in by
  calculate_ci <- function(stratum) {
    subset_design <-
      subset(design,
             design$variables[[by_var]] == stratum)
    # Use provided df or default to subset design df
    df_to_use <- if (is.null(df)) degf(subset_design) else df
    result <- svyciprop(x, design = subset_design,
                        method = method, df = df_to_use)
    return(attr(result, "ci"))
  }

  # tabulate the result
  ci_results <- lapply(by_levels, calculate_ci)
  results <- data.frame(subset = by_levels,
                        ci = do.call(rbind, ci_results))

  return(results)
}
```
```

> **Note**
>
> this custom function can be simplified, but i make it more general so it can be use to other too.

5.2.2.8 Total Sample and Estimated Population

Can you try calculate the total sample? Using the example from calculating the total number sample with the outcome.

The tutorial on estimated total population will be cover in Bonus II: Population Pyramid part

5.3 Bonus I: Regression (Linear Regression & Logistic Regression)

5.3.1 Logistic Regression

5.3.1.1 Simple Logistic Regression

```
  ```{r}
  svyglm(known_chol ~ strata_gp,
         nhms_surdsg,
         family = quasibinomial) %>%
    summary()
  ```
```

```
Call:
svyglm(formula = known_chol ~ strata_gp, design = nhms_surdsg,
    family = quasibinomial)

Survey design:
svydesign(id = ~ebid, strata = ~state_st, weights = ~wtfinal_ncd,
    data = nhms19ds, nest = T)

Coefficients:
               Estimate Std. Error t value Pr(>|t|)
(Intercept)    -1.83690    0.05134 -35.779   <2e-16 ***
strata_gpRural -0.10511    0.08976  -1.171    0.242
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for quasibinomial family taken to be 1.000096)

Number of Fisher Scoring iterations: 4
```

5.3.1.2 Multiple Logistic Regression

```
  ```{r}
  svyglm(known_chol ~ strata_gp + A2101 + A2108_3grp,
         nhms_surdsg,
         family = quasibinomial) %>%
    summary()
  ```
```

```
Call:
svyglm(formula = known_chol ~ strata_gp + A2101 + A2108_3grp,
    design = nhms_surdsg, family = quasibinomial)

Survey design:
svydesign(id = ~ebid, strata = ~state_st, weights = ~wtfinal_ncd,
    data = nhms19ds, nest = T)

Coefficients:
                            Estimate Std. Error t value Pr(>|t|)
(Intercept)                -3.42541    0.17958 -19.075  <2e-16 ***
strata_gpRural               -0.13590    0.09510  -1.429    0.154
A2101Female                   0.01669    0.07898   0.211    0.833
A2108_3grpMarried           1.76977    0.18140   9.756  <2e-16 ***
A2108_3grpWidow(er)/Divercee 2.79062    0.19772  14.114  <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for quasibinomial family taken to be 1.00007)

Number of Fisher Scoring iterations: 5
```

5.3.2 Linear Regression

5.3.2.1 Simple Linear Regression

```
  ```{r}
  svyglm(u303 ~ strata_gp,
         nhms_surdsg,
         family = gaussian) %>%
    summary()
  ```
```

```
Call:
svyglm(formula = u303 ~ strata_gp, design = nhms_surdsg, family = gaussian)

Survey design:
svydesign(id = ~ebid, strata = ~state_st, weights = ~wtfinal_ncd,
    data = nhms19ds, nest = T)

Coefficients:
               Estimate Std. Error t value Pr(>|t|)
(Intercept)     4.77993    0.02913 164.080   <2e-16 ***
strata_gpRural -0.04295    0.04288  -1.002    0.317
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for gaussian family taken to be 1.329693)

Number of Fisher Scoring iterations: 2
```

5.3.2.2 Multiple Linear Regression

```
  ```{r}
  options(width = 70) # the output width limit

  svyglm(u303 ~ strata_gp+ A2101 + A2108_3grp,
         nhms_surdsg,
         family = gaussian) %>%
    summary()
  ```
```

```
Call:
svyglm(formula = u303 ~ strata_gp + A2101 + A2108_3grp, design = nhms_surdsg,
    family = gaussian)

Survey design:
svydesign(id = ~ebid, strata = ~state_st, weights = ~wtfinal_ncd,
    data = nhms19ds, nest = T)

Coefficients:
                             Estimate Std. Error t value Pr(>|t|)
(Intercept)                 4.30828    0.04180 103.070  < 2e-16 ***
strata_gpRural               -0.03732    0.04205  -0.887    0.375
A2101Female                 0.40032    0.03174  12.612  < 2e-16 ***
A2108_3grpMarried            0.39027    0.03945   9.893  < 2e-16 ***
A2108_3grpWidow(er)/Divercee 0.38802    0.06279   6.180 1.46e-09 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for gaussian family taken to be 1.251478)

Number of Fisher Scoring iterations: 2
```

5.4 Bonus II: Mapping the Prevalence

We can map our prevalence.

1. save the prevalence by state into object to be used later

```
```{r}
kcprev_state <- svyby(formula = ~known_cholN,
                      by = ~state,
                      design = nhms_surdsg,
                      FUN = svymean,
                      na.rm.all = T) %>%
  as_tibble()

kcprev_state
```
```

```
# A tibble: 16 x 3
   state        known_cholN      se
   <fct>              <dbl>   <dbl>
 1 Johor             0.106  0.0112
 2 Kedah             0.168  0.0214
 3 Kelantan          0.106  0.00732
 4 Melaka            0.154  0.0189
 5 N. Sembilan       0.188  0.0293
 6 Pahang            0.112  0.0138
 7 P. Pinang         0.185  0.0270
 8 Perak             0.202  0.0248
 9 Perlis            0.177  0.0190
10 Selangor          0.120  0.0137
11 Terengganu        0.130  0.0138
12 Sabah             0.0836 0.0116
13 Sarawak           0.154  0.0156
14 WP Kl             0.154  0.0177
15 WP Labuan         0.149  0.0154
16 WP Putrajaya      0.146  0.0188
```

2. download the state map (geojson file) from DOSM github page

```
```{r}
#| eval: false

download.file(
  url = "https://raw.githubusercontent.com/dosm-malaysia/data-open/main/dat
  destfile = "administrative_1_state.geojson",
  mode = "wb")
```
```

> **Important**
>
> dosm github link to download the map dataset: https://raw.githubusercontent.com/dosm-malaysia/data-open/main/datasets/geodata/administrative_1_state.geojson

3. in R, map files like geojson and shp file is manipulated using `sf ::` package
    - load sf package, if not available, please install first.

````
```{r}
library(sf)
```
````

4. convert the geojson file and save in r object.
    - in the same time, we can do some data wrangling, to ensure the name of state in dosm dataaset and our dataset is consistent.

````
```{r}
my_state_sf <- read_sf("administrative_1_state.geojson") %>%
  arrange(code_state) %>%
  mutate(state = fct_recode(state,
                            "P. Pinang" = "Pulau Pinang",
                            "N. Sembilan" = "Negeri Sembilan",
                            "WP Kl" = "W.P. Kuala Lumpur",
                            "WP Putrajaya" = "W.P. Putrajaya",
                            "WP Labuan" = "W.P. Labuan"))
```
````

5. Join both prevalence by state result and dosm state map.
    - the combined dataset need to convert to sf object

````
```{r}
#| eval: false

kcprev_state_mapds <- left_join(kcprev_state, my_state_sf) %>%
  st_as_sf()

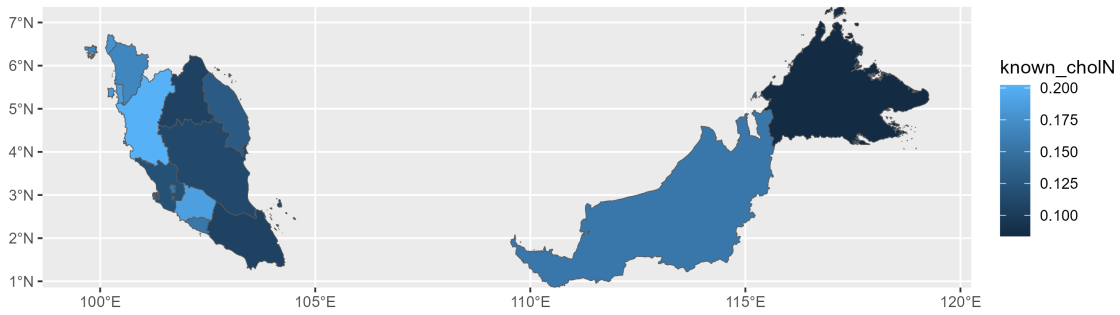
kcprev_state_mapds
```
````

> **Note**
>
> any sf item must have geometry column, which contain the information of the location

```
Simple feature collection with 16 features and 4 fields
Geometry type: MULTIPOLYGON
Dimension:     XY
Bounding box:  xmin: 99.6409 ymin: 0.85539 xmax: 119.269 ymax: 7.36098
Geodetic CRS:  WGS 84
# A tibble: 16 x 5
   state     known_cholN     se code_state                geometry
   <fct>           <dbl>  <dbl>     <int>      <MULTIPOLYGON [°]>
 1 Johor         0.106  0.0112         1 (((103.416 1.31868, 103.~
 2 Kedah         0.168  0.0214         2 (((100.7375 5.30512, 100~
 3 Kelantan      0.106  0.00732        3 (((101.8147 4.75934, 101~
 4 Melaka        0.154  0.0189         4 (((102.3322 2.04767, 102~
 5 N. Sembil~    0.188  0.0293         5 (((102.6248 2.62871, 102~
 6 Pahang        0.112  0.0138         6 (((103.9788 2.70211, 103~
 7 P. Pinang     0.185  0.0270         7 (((100.5371 5.2666, 100.~
 8 Perak         0.202  0.0248         8 (((100.7609 4.0423, 100.~
 9 Perlis        0.177  0.0190         9 (((100.2104 6.72068, 100~
10 Selangor      0.120  0.0137        10 (((101.7533 2.81998, 101~
11 Terengganu    0.130  0.0138        11 (((103.4645 4.57023, 103~
12 Sabah        0.0836 0.0116        12 (((118.6798 4.07375, 118~
13 Sarawak       0.154  0.0156        13 (((110.7571 1.55217, 110~
14 WP Kl         0.154  0.0177        14 (((101.6672 3.24432, 101~
15 WP Labuan     0.149  0.0154        15 (((115.1419 5.18637, 115~
16 WP Putraj~    0.146  0.0188        16 (((101.6985 2.97171, 101~
```

6. we can then plot the prevalence using ggplot

```
1  ```{r}
2  #| eval: false
3
4  CholPrevMalMap <- kcprev_state_mapds %>%
5    ggplot(aes(fill = known_cholN))
6  ```
```



5.5 Bonus III: Population Pyramid:

Despite weight adjustment which include post-stratification, the total estimated population may differ from the original population. Here, plotting population pyramid can help to compare NHMS estimated population and DOSM 2019 Population

1. Download the DOSM Population 2019 from DOSM opendata

```
```{r}
#| eval: false

download.file(
  url = "https://storage.dosm.gov.my/population/population_malaysia.par
  destfile = "population_malaysia.parquet",
  mode = "wb")
```
```

> **Important**
>
> dosm open website to download the dataset:
> https://open.dosm.gov.my/data-catalogue/population_malaysia

2. Import downloaded dataset and wrangle it
    - to exclude data not required
    - to make data "compatible" with our NHMS dataset

> the data wrangling code is in the next page

```
```{r}
dosmpop19
```
```

```
# A tibble: 32 x 4
   gender age_grp type  population
   <fct>  <fct>   <chr>      <dbl>
 1 Male   0-4     dosm     1331100
 2 Male   5-9     dosm     1317700
 3 Male   10-14   dosm     1299400
 4 Male   15-19   dosm     1480100
 5 Male   20-24   dosm     1675900
 6 Male   25-29   dosm     1746200
 7 Male   30-34   dosm     1502400
 8 Male   35-39   dosm     1304000
 9 Male   40-44   dosm     1022300
10 Male   45-49   dosm      895900
# i 22 more rows
```

```r
library(arrow)
dosmpop19 <- read_parquet("population_malaysia.parquet") %>%
  filter(date == "2019-01-01",             # to get 2019 population only
         sex ≠ "overall_sex",              # exclude overall
         ethnicity == "overall_ethnicity", # exclude overall
         age ≠ "overall_age") %>%          # exclude overall
  rename("gender" = "sex") %>%
  mutate(type = "dosm",
         gender = fct_recode(gender,
                             "Male" = "male",
                             "Female" = "female"),
         gender = fct_relevel(gender, "Male"),
         age_grp = case_when(age %in% c("75-79",
                                        "80-84",
                                        "85+") ~ "75+",
                             .default = age),
         age_grp = fct_relevel(age_grp,
                               "0-4", "5-9", "10-14",
                               "15-19", "20-24", "25-29",
                               "30-34", "35-39", "40-44",
                               "45-49", "50-54", "55-59",
                               "60-64", "65-69", "70-74",
                               "75+"),
         population = population * 1000) %>%
  select(-c(date, state, ethnicity, age)) %>% # not required
  group_by(gender, age_grp, type) %>%
  summarise(population = sum(population, na.rm = T)) %>%
  ungroup()
```

2. Calculate Total NHMS Estimated Population

- this require a bit of a work, since we want to calculate all sample
- one way of doing it, is by create a new column, with value of 1
- this new column need to be done in the original dataset, thus a new survey design need to be constructed
- then we can count the unweighted count and the estimated population, by age group and gender, using `svyby(_)` function
- save the estimated population to an R object to used later

```r
nhms19ds_all <- nhms19ds %>%
  mutate(cholall = 1)
```

```
```{r}
nhms_surdsg_all <- svydesign(id = ~ebid,
                             strata = ~state_st,
                             weights = ~wtfinal_ncd,
                             data = nhms19ds_all,
                             nest = TRUE)

nhmspop19 <- svyby(formula = ~cholall,
                   by = ~A2104_grp+A2101,
                   design = nhms_surdsg_all,
                   FUN = svytotal) %>%
  as_tibble() %>%
  rename("gender" = "A2101",
         "age_grp" = "A2104_grp",
         "population" = "cholall") %>%
  mutate(type = "nhms",
         age_grp = fct_recode(age_grp,
                              "75+" = "75  & above")) %>%
  select(-se)

nhmspop19
```
```

```
# A tibble: 26 x 4
   age_grp gender population type
   <fct>   <fct>       <dbl> <chr>
 1 15-19   Male      497478. nhms
 2 20-24   Male     1496318. nhms
 3 25-29   Male     1585427. nhms
 4 30-34   Male     1412422. nhms
 5 35-39   Male     1221228. nhms
 6 40-44   Male      965349. nhms
 7 45-49   Male      824204. nhms
 8 50-54   Male      758598. nhms
 9 55-59   Male      672648. nhms
10 60-64   Male      549455. nhms
# i 16 more rows
```

3. Join DOSM population and NHMS population

    - since we want female on left side, the female population need to be in negative form
    - the `female` also need to be in lower level
    - and save the join dataset to R object, to be used later

````
```{r}
join_pop19 <- full_join(dosmpop19, nhmspop19) %>%
  arrange(gender, age_grp) %>%
  filter(!age_grp %in% c("0-4", "5-9", "10-14")) %>%
  mutate(population = case_when(gender == "Male" ~ population,
                                gender == "Female" ~ 0 - population),
         gender = fct_relevel(gender, "Female"))

join_pop19
```
````

```
# A tibble: 52 x 4
   gender age_grp type  population
   <fct>  <fct>   <chr>      <dbl>
 1 Male   15-19   dosm     1480100
 2 Male   15-19   nhms      497478.
 3 Male   20-24   dosm     1675900
 4 Male   20-24   nhms     1496318.
 5 Male   25-29   dosm     1746200
 6 Male   25-29   nhms     1585427.
 7 Male   30-34   dosm     1502400
 8 Male   30-34   nhms     1412422.
 9 Male   35-39   dosm     1304000
10 Male   35-39   nhms     1221228.
# i 42 more rows
```

4. we can plot the pyramid plot

```r
join_pop19 %>%
  ggplot(aes(x = age_grp,
             y = population,
             fill = interaction(gender, type))) +
  geom_col(position = "dodge") +
  scale_y_continuous(expand = c(0, 0),
                     labels = function(x) scales::label_comma()(abs(x)),
                     breaks = scales::pretty_breaks()) +
  scale_fill_manual(values = hcl(h = c(15, 195, 15, 195),
                                 c = 100,
                                 l = 65,
                                 alpha = c(.4, .4, 1, 1)),
                    name = "") +
  coord_flip() +
  facet_wrap(. ~gender,
             scales = "free_x",
             strip.position = "bottom") +
  theme_bw() +
  theme(panel.border = element_blank(),
        axis.ticks.y = element_blank(),
        panel.grid.major.y = element_blank(),
        legend.position = "bottom",
        panel.spacing.x = unit(0, "pt"),
        strip.background = element_rect(colour = "black"))
```