

R Training Book for IKU

Table of contents

Preface	1
Objectives	1
Way Forward	1
Collaborators Are Welcome	1
1 How To Use This Book	3
2 Why Tidyverse?	5
3 Why dataset in SPSS file?	7
4 Introduction	9
4.1 R	9
4.2 RStudio	9
4.3 Quarto	9
5 Data Wrangling	11
6 Data Wrangling	13
7 Statistical Tests	15
7.1 Setup Project	15
7.2 Bivariate Analyses of Categorical Data	16
7.2.1 Contingency Table	16
7.2.2 Pearson's Chi-Square Test	16
7.3 Bivariate Analyses of Numerical Data	17
7.4 Linear Regression	17
7.5 Logistic Regression	17
8 Complex Sampling Design in NHMS	19
8.1 Why Complex Sampling Design?	19
8.1.1 Benefits of Complex Sampling Design	20
8.1.2 Challenges in Implementing Complex Sampling Design	20
8.1.3 Example: Sampling Probability of a Sabahan	20
8.2 Practical	21
8.2.1 Setup Project	21
8.2.2 Analysis	21
8.3 Bonus I: Regression (Linear Regression & Logistic Regression)	34
8.3.1 Logistic Regression	34
8.3.2 Linear Regression	36
8.4 Bonus II: Mapping the Prevalence	38
8.5 Bonus III: Population Pyramid:	41

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 invited for collaboration. Whether one is an author with insights to share, an editor with an eye for detail, or possesses constructive suggestions, these contributions can significantly enhance the utility and reach of this manual. It is particularly interested in contributions in the following areas:

Preface

- **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 How To Use This Book

2 Why Tidyverse?

3 Why dataset in SPSS file?

4 Introduction

4.1 R

4.2 RStudio

4.3 Quarto

5 Data Wrangling

In summary, this book has no content whatsoever.

```
1 1 + 1
```

```
[1] 2
```


6 Data Wrangling

In summary, this book has no content whatsoever.

```
1 1 + 1
```

```
[1] 2
```


7 Statistical Tests

Several statistical tests will be covered, which will be grouped into

- Bivariate analyses of categorical data
- Bivariate analyses of numerical data
- Linear regression
- Logistic regression

This book however was not intended to be a statistical book, nor should not be the main reference for statistical analyses. Please refer your statistical texts book for further information.

In addition of conducting the test in standard way, I'll also how to create a nice result table using related packages.

i Note

dataset used in this example can be downloaded from github site:
https://github.com/MohdAzmiSuliman/IKU_RBook/tree/master/dataset

7.1 Setup Project

1. Setup your project

- Open your RStudio
- Create New Project

2. Create Quarto document

- update the YAML metadata to make the document self-contained

```
1 ---
2 title: "Sesi 3 - Basic Statistical Analysis"
3 format:
4   html:
5     embed-resources: true
6 ---
```

7.2 Bivariate Analyses of Categorical Data

Bivariate analysis involves examining the relationship or association between two variables. When both variables are categorical, bivariate analysis looks at how these categories are related. The bivariate analyses often involve the use of contingency tables, also known as cross-tabulation or crosstab.

7.2.1 Contingency Table

Contingency table, or crosstab, is a simple matrix that displays the frequency of occurrence of combination of two categories for two categorical variables.

For example, this contingency table shows a cross tabulation between smoking status and lung cancer outcome.

Table 7.1: Contingency Table Structure

Status	Lung Cancer	No Lung Cancer	Total
Smoker	a	b	a+b
Non-Smoker	c	d	c+d
Total	a+c	b+d	N

In this example:

- the independent variable (or the predictor, i.e. the smoking status), is the row
- the dependent variable (or the outcome, i.e. the cancer status), is the column
- The letters a, b, c, and d represent the cell frequencies:
 - a: The number of smokers who have the disease.
 - b: The number of smokers who do not have the disease.
 - c: The number of non-smokers who have the disease.
 - d: The number of non-smokers who do not have the disease.
- “Total” is the sum of the frequencies in the respective row or column, with N being the grand total of all observations.

7.2.2 Pearson’s Chi-Square Test

Pearson Chi-square Test is a statistical test used to determine whether there is a significant association between two categorical variables.

Pearson Chi-square Test is conducted with these assumptions:

- both variables must be categorical (either nominal or ordinal)
- the observation must be independent. This means that the outcome of one observation is not influenced by the outcome of another observation
- the groupings are mutually exclusive
- <20% of cells have expected frequency of <5
- minimum expected frequency if >1

Example: Gender vs Employment Status

1. Prepare

7.3 Bivariate Analyses of Numerical Data

7.4 Linear Regression

7.5 Logistic Regression

8 Complex Sampling Design in NHMS

8.1 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 overrepresentation 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.

8.1.1 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.

8.1.2 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.

8.1.3 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!.

8.2 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 8.1: 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

8.2.1 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

```

1 ---
2 title: "Sesi 4 - NHMS"
3 format:
4   html:
5     embed-resources: true
6 ---

```

8.2.2 Analysis

8.2.2.1 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_cho1`)
 - 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

8 Complex Sampling Design in NHMS

```
1  ```{r}
2  #| output: false
3
4  library(tidyverse)
5  library(haven)
6
7  nhms19ds <- read_sav("nhms19ds.sav") %>%
8    as_factor()
9
10 nhms19ds
11  ```
```

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.

```
1  ```{r}
2  #| eval: false
3
4  library(skimr)
5
6  nhms19ds %>%
7    select(known_chol, u303) %>%
8    skim()
9  ```
```

Variable Name	Variable Label
state	[Final] State
strata_gp	[Final] Locality
A2101	[Final] Gender
A2104	Age (Numerical)
A2104_grp	[Final] Age Group - 16 groups
A2106_5grp	Ethnicity (5 groups)
A2107	Citizenship
A2108_3grp	[Final] Marital Status (3 groups)
A2109_4grp	[Final] Highest Education Level (5 groups)
A2221	If working, type of occupation
A2222_7grp	Employment status (7 groups)
A2222_5grp	[Final] Occupation (5 groups)
indvid	_no label_
hh_id	_no label_
state_st	PSU
ebid	EB ID - Cluster
wtfinal_ncd	Sampling Weight
c01	ever had total blood cholesterol level measured
c02	ever told have high cholesterol level
c03	when told to have high cholesterol

NHMS NCD 2019 - Cholesterol Module Dataset: Variables List

Variable Name	Variable Label
c03a	years since was told to have high cholesterol
c04a	on medication for past 2 week
c04b	advice for special low fat diet
c04c	advice to loose weight
c04d	advice to exercise
c05	treatment - herbal/TCM
c06	common place to receive treatment
u303	Total Cholesterol (mmol/L)
known_chol	_no label_
undiagnosed_chol	_no label_
total_chol	_no label_
bodyweight1	Body Weight (kg)
bodyweight2	Body Weight (kg)
bodyheight1	Body Height (cm)
bodyheight2	Body Height (cm)
wc2	Waist Circumference (cm)
wc1	Waist Circumference (cm)
weight	Body Weight (kg)
height	Body Height (cm)
wc	Waist Circumference (cm)

8 Complex Sampling Design in NHMS

Table 8.2: 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

```

1  ```{r}
2  nhms19ds <- nhms19ds %>%
3    as_factor() %>%
4    filter(!is.na(known_chol)) %>%
5    mutate(known_cholN = as.numeric(known_chol)-2,
6           u303 = as.numeric(as.character(u303)))
7  ```

```

i Note

The variable `known_chol` 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

```
1  ```{r}
2  #| eval: false
3
4  nhms19ds %>%
5    select(known_chol, known_cholN)
6  ```
```

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

```
1  ```{r}
2  library(survey)
3
4  options(survey.lonely.psu = 'adjust',
5          survey.adjust.domain.lonely = TRUE)
6  ```
```

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

```
1  ```{r}
2  nhms_unwdsg <- svydesign(id = ~1,
3                          weights = ~1,
4                          data = nhms19ds)
5  ```
```

8 Complex Sampling Design in NHMS

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

```
1 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

```
1 ```{r}
2 nhms_surdsg <- svydesign(id = ~ebid,
3                           strata = ~state_st,
4                           weights = ~wtfinal_ncd,
5                           data = nhms19ds,
6                           nest = T)
7 ```
```

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

```

1  ```{r}
2  options(width = 70) # the output width limit
3
4  summary(nhms_surdsg)
5  ```

```

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
obs	317	265	258	294	898	224	301	341	405	429	388	358	504	420
design.PSU	16	11	12	12	53	11	11	13	20	19	16	14	25	19
actual.PSU	16	11	12	12	53	11	11	13	20	19	16	14	25	19
	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).

8.2.2.2 Count the unweighted sample

1. To count the number of sample, we will use function `svytotal()` 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.

```

1  ```{r}
2  svytotal(x = ~known_chol,
3           design = nhms_unwdsg,
4           na.rm = T)
5  ```

```

	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.

```

1  ```{r}
2  svytotal(x = ~known_cholN,
3           design = nhms_unwdsg,
4           na.rm = T)
5  ```

```

	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.

8.2.2.3 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


```

1  ```{r}
2  svytotal(x = ~known_cholN,
3           design = nhms_surdsg,
4           na.rm = T)
5  ```

```

```

              total      SE
known_cholN 2868124 103013

```

8.2.2.4 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

```

1  ```{r}
2  svymean(x = ~known_cholN,
3          design = nhms_surdsg,
4          na.rm = T)
5  ```

```

```

              mean      SE
known_cholN 0.13479 0.0051

```

8.2.2.5 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 confidence 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, 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”

```

1  ```{r}
2  svyciprop(formula = ~known_cholN,
3            design = nhms_surds,
4            method = "xlogit") %>%
5    attr(, "ci")
6  ```

```

```

      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()`

8.2.2.6 Estimating the Unweighted Sample Proportion

Can you calculate the sample proportion using the same function?

Tip

Hint:

1. Sample Proportion = Unweighted Proportion.
2. Unweighted design vs. Weighted design.

8.2.2.7 Estimating by Subpopulation

1. To estimate by subpopulation, we use `svyby()` function
2. Estimating the unweighted count by locality (urban vs rural)

- Don't forget to use the unweighted design

```

1  ```{r}
2  svyby(formula = ~known_cholN,
3        by = ~strata_gp,
4        design = nhms_unwdsg,
5        FUN = svytotal,
6        na.rm.all = T)
7  ```

```

	strata_gp	known_cholN	se
Urban	Urban	1198	32.57255
Rural	Rural	817	27.44622

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

```

1  ```{r}
2  svyby(formula = ~known_cholN,
3        by = ~strata_gp,
4        design = nhms_surdsg,
5        FUN = svytotal,
6        na.rm.all = T)
7  ```

```

	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 rural)

```

1  ```{r}
2  svyby(formula = ~known_cholN,
3        by = ~strata_gp,
4        design = nhms_unwdsg,
5        FUN = svymean,
6        na.rm.all = T)
7  ```

```

	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 rural).

- unfortunately, `svyciprop()` can't be used with `svyby()` function.
- to estimate the CI, we need to subset the sample, to only the sub-population.

8 Complex Sampling Design in NHMS

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

```
1  ```{r}
2  nhms_surdsg_urban <- subset(nhms_surdsg,
3                             strata_gp = "Urban")
4
5  svyciprop(formula = ~known_cholN,
6             design = nhms_surdsg_urban,
7             method = "xl",
8             df = degf(nhms_surdsg)) %>%
9    attr(., "ci")
10  ```
```

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

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

```
1  ```{r}
2  svyciprop_by(x = ~known_cholN,
3               by = ~strata_gp,
4               design = nhms_surdsg,
5               df = degf(nhms_surdsg),
6               method = "xl")
7  ```
```

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

the custom function code

```

1  ```{r}
2  # create a svyby-like function specific for svyciprop
3  svyciprop_by <- function(x, by, design,
4                          df = NULL, method = NULL) {
5    # extract the levels in by
6    by_var <- all.vars(by)[1]
7    by_data <- model.frame(by, data = design$variables)
8    by_levels <- sort(unique(by_data[[by_var]]))
9
10   # run the svyciprop() functions on each levels in by
11   calculate_ci <- function(stratum) {
12     subset_design <-
13       subset(design,
14             design$variables[[by_var]] = stratum)
15     # Use provided df or default to subset design df
16     df_to_use <- if (is.null(df)) degf(subset_design) else df
17     result <- svyciprop(x, design = subset_design,
18                       method = method, df = df_to_use)
19     return(attr(result, "ci"))
20   }
21
22   # tabulate the result
23   ci_results <- lapply(by_levels, calculate_ci)
24   results <- data.frame(subset = by_levels,
25                        ci = do.call(rbind, ci_results))
26
27   return(results)
28 }
29 ```

```

i Note

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

8.2.2.8 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

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

8.3.1 Logistic Regression

8.3.1.1 Simple Logistic Regression

```

1  ```{r}
2  svyglm(known_chol ~ strata_gp,
3         nhms_surdsg,
4         family = quasibinomial) %>%
5    summary()
6  ```

```

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

8.3.1.2 Multiple Logistic Regression

```

1  ```{r}
2  svyglm(known_chol ~ strata_gp + A2101 + A2108_3grp,
3         nhms_surdsg,
4         family = quasibinomial) %>%
5     summary()
6  ```

```

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

8 Complex Sampling Design in NHMS

8.3.2 Linear Regression

8.3.2.1 Simple Linear Regression

```
1  ```{r}
2  svyglm(u303 ~ strata_gp,
3         nhms_surds,
4         family = gaussian) %>%
5    summary()
6  ```
```

Call:

```
svyglm(formula = u303 ~ strata_gp, design = nhms_surds, 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

8.3.2.2 Multiple Linear Regression

```

1  ```{r}
2  options(width = 70) # the output width limit
3
4  svyglm(u303 ~ strata_gp+ A2101 + A2108_3grp,
5         nhms_surdsg,
6         family = gaussian) %>%
7    summary()
8  ```

```

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

8.4 Bonus II: Mapping the Prevalence

We can map our prevalence.

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

```
1  ```{r}
2  kcprev_state <- svyby(formula = ~known_cholN,
3                        by = ~state,
4                        design = nhms_surdsg,
5                        FUN = svymean,
6                        na.rm.all = T) %>%
7    as_tibble()
8
9  kcprev_state
10  ```
```

```
# 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

```
1  ```{r}
2  #| eval: false
3
4  download.file(
5    url = "https://raw.githubusercontent.com/dosm-malaysia/data-open/main/datasets/geodata/administrative_1_state.geojson",
6    destfile = "administrative_1_state.geojson",
7    mode = "wb")
8  ```
```

! Important

dosm github link to download the map dataset: https://raw.githubusercontent.com/dosm-malaysia/data-open/main/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.

```
1  ```{r}
2  library(sf)
3  ```
```

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 dataset and our dataset is consistent.

```
1  ```{r}
2  my_state_sf <- read_sf("administrative_1_state.geojson") %>%
3    arrange(code_state) %>%
4    mutate(state = fct_recode(state,
5                              "P. Pinang" = "Pulau Pinang",
6                              "N. Sembilan" = "Negeri Sembilan",
7                              "WP KL" = "W.P. Kuala Lumpur",
8                              "WP Putrajaya" = "W.P. Putrajaya",
9                              "WP Labuan" = "W.P. Labuan"))
10  ```
```

5. Join both prevalence by state result and dosm state map.

- the combined dataset need to convert to `sf` object

```
1  ```{r}
2  #| eval: false
3
4  kcprev_state_mapds <- left_join(kcprev_state, my_state_sf) %>%
5    st_as_sf()
6
7  kcprev_state_mapds
8  ```
```

i Note

any `sf` item must have geometry column, which contain the information of the location

8 Complex Sampling Design in NHMS

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

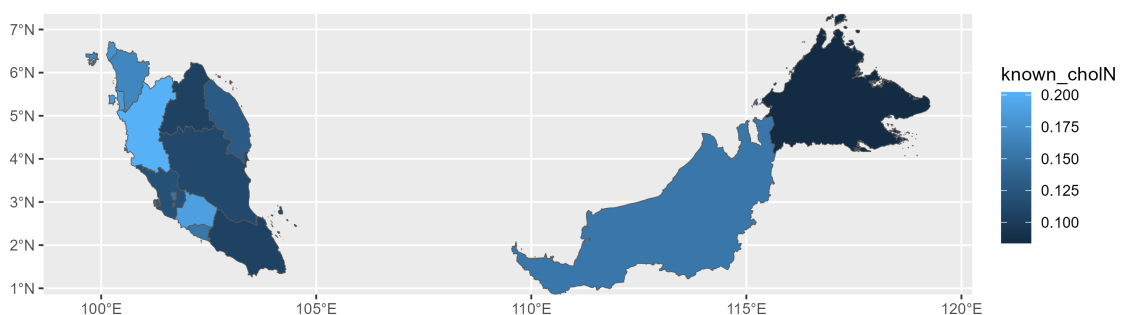
Geodetic CRS: WGS 84

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   ```
```



8.5 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

```
1  ```{r}
2  #| eval: false
3
4  download.file(
5    url = "https://storage.dosm.gov.my/population/population_malaysia.parquet",
6    destfile = "population_malaysia.parquet",
7    mode = "wb")
8  ```
```

! 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

i the data wrangling code is in the next page

```
1  ```{r}
2  dosmpop19
3  ```
```

```
# 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
```

8 Complex Sampling Design in NHMS

```
1  ```{r}
2  library(arrow)
3  dosmpop19 <- read_parquet("population_malaysia.parquet") %>%
4    filter(date = "2019-01-01",          # to get 2019 population only
5          sex != "overall_sex",          # exclude overall
6          ethnicity = "overall_ethnicity", # exclude overall
7          age != "overall_age") %>%      # exclude overall
8    rename("gender" = "sex") %>%
9    mutate(type = "dosm",
10          gender = fct_recode(gender,
11                              "Male" = "male",
12                              "Female" = "female"),
13          gender = fct_relevel(gender, "Male"),
14          age_grp = case_when(age %in% c("75-79",
15                                         "80-84",
16                                         "85+") ~ "75+",
17                               .default = age),
18          age_grp = fct_relevel(age_grp,
19                                "0-4", "5-9", "10-14",
20                                "15-19", "20-24", "25-29",
21                                "30-34", "35-39", "40-44",
22                                "45-49", "50-54", "55-59",
23                                "60-64", "65-69", "70-74",
24                                "75+"),
25          population = population * 1000) %>%
26    select(-c(date, state, ethnicity, age)) %>% # not required
27    group_by(gender, age_grp, type) %>%
28    summarise(population = sum(population, na.rm = T)) %>%
29    ungroup()
30  ```
```

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

```
1  ```{r}
2  nhms19ds_all <- nhms19ds %>%
3    mutate(cholall = 1)
4  ```
```

```

1  ```{r}
2  nhms_surdsg_all <- svydesign(id = ~ebid,
3                           strata = ~state_st,
4                           weights = ~wtfinal_ncd,
5                           data = nhms19ds_all,
6                           nest = TRUE)
7
8  nhmspop19 <- svyby(formula = ~cholall,
9                    by = ~A2104_grp+A2101,
10                   design = nhms_surdsg_all,
11                   FUN = svytotal) %>%
12    as_tibble() %>%
13    rename("gender" = "A2101",
14          "age_grp" = "A2104_grp",
15          "population" = "cholall") %>%
16    mutate(type = "nhms",
17          age_grp = fct_recode(age_grp,
18                               "75+" = "75 & above")) %>%
19    select(-se)
20
21  nhmspop19
22  ```

```

```

# 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

```

8 Complex Sampling Design in NHMS

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

```
1  ```{r}
2  join_pop19 <- full_join(dosmpop19, nhmspop19) %>%
3    arrange(gender, age_grp) %>%
4    filter(!age_grp %in% c("0-4", "5-9", "10-14")) %>%
5    mutate(population = case_when(gender == "Male" ~ population,
6                                  gender == "Female" ~ 0 - population),
7          gender = fct_relevel(gender, "Female"))
8
9  join_pop19
10  ```
```

```
# 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

```

1  ```{r}
2  join_pop19 %>%
3    ggplot(aes(x = age_grp,
4              y = population,
5              fill = interaction(gender, type))) +
6    geom_col(position = "dodge") +
7    scale_y_continuous(expand = c(0, 0),
8                      labels = function(x) scales::label_comma()(abs(x)),
9                      breaks = scales::pretty_breaks()) +
10   scale_fill_manual(values = hcl(h = c(15, 195, 15, 195),
11                                c = 100,
12                                l = 65,
13                                alpha = c(.4, .4, 1, 1)),
14                     name = "") +
15   coord_flip() +
16   facet_wrap(. ~gender,
17             scales = "free_x",
18             strip.position = "bottom") +
19   theme_bw() +
20   theme(panel.border = element_blank(),
21         axis.ticks.y = element_blank(),
22         panel.grid.major.y = element_blank(),
23         legend.position = "bottom",
24         panel.spacing.x = unit(0, "pt"),
25         strip.background = element_rect(colour = "black"))
26  ```

```

