

Table of contents

Pro	етасе		1							
		ctives								
	Way	Forward	1							
	Colla	aborators Are Welcome	1							
1	How	To Use This Book	3							
2	Why	Tidyverse?	5							
3	Why	dataset in SPSS file?	7							
4	Intro	oduction	9							
	4.1	R	9							
	4.2	RStudio	9							
	4.3	Quarto	9							
5	Data	Wrangling	11							
	5.1	What is Data Wrangling?	11							
	5.2	Tidyverse	11							
		5.2.1 Pipe & Mutate	11							
	5.3	Practical	12							
		5.3.1 Setup Project	12							
		5.3.2 Data Import	12							
		5.3.3 Data Wrangling	14							
		5.3.4 Save/Export Dataset	28							
		5.3.5 Data Exploration	28							
	5.4	Addin: questionr:: package	30							
6	Data Vizualization 33									
	6.1	Practical - Setup Project	33							
	6.2	esquisse:: RStudio Addin	34							
	6.3	Plot Examples	34							
		6.3.1 Histogram	35							
		6.3.2 Boxplot	37							
		6.3.3 Bar Chart	39							
		6.3.4 Scatter Plot	42							
		6.3.5 Line Chart	44							
		6.3.6 Spaghetti Plot	45							
	6.4	ggplot2::	46							
7	Stati	stical Tests	53							
	7 1	Practical: Setup Project	53							

Table of contents

	7.2	Bivariate Analyses of Categorical Data	
		7.2.1 Contingency Table	
		7.2.2 Pearson's Chi-Square Test	
		7.2.3 Small Sample	
		7.2.4 McNemar Test	60
	7.3	Bivariate Analyses of Numerical Data	61
		7.3.1 Independent T-test	61
		7.3.2 Paired T-test	
		7.3.3 ANOVA - Analysis of Variance	
		7.3.4 Pearson's Correlation	
	7.4	Linear Regression	
		7.4.1 Multiple Linear Regression	
	7.5	Logistic Regression	
		7.5.1 Simple Logistic Regression	
8	Com	nplex Sampling Design in NHMS	95
	8.1	Why Complex Sampling Design?	
		8.1.1 Benefits of Complex Sampling Design	
		8.1.2 Challenges in Implementing Complex Sam	npling Design 96
		8.1.3 Example: Sampling Probability of a Sabah	an
	8.2	Practical	
		8.2.1 Setup Project	
		8.2.2 Analysis	
	8.3	Bonus I: Regression (Linear Regression & Logistic	Regression) 110
		8.3.1 Logistic Regression	
		8.3.2 Linear Regression	
	8.4	Bonus II: Mapping the Prevalence	
	8.5	Bonus III: Population Pyramid	115

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:

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

5.1 What is Data Wrangling?

· Common data analysis look like this

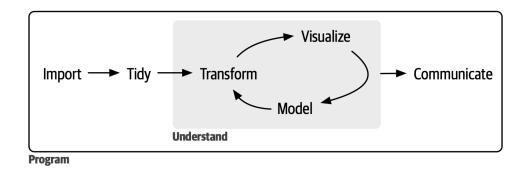


Figure 5.1: source: r4ds.hadley.nz

- · real world data commonly messy!
- 80% of time taken spend on data cleaning
- improving data quality → undefined undefin undefined improving the accuracy & ef
- data wrangling involving tidying and transforming data, from raw form to analysis-ready data.
- common data wrangling action
 - label data
 - recategorise categorical variable usually collapsing groups
 - binning continuous variable

5.2 Tidyverse

5.2.1 Pipe & Mutate

- functional programming operators, %>% called pipe, allow for chaining functions in sequence
 - result of left-hand side is passed as the first argument to the function of right-hand side
- in tidyverse, wrangling data can be done simultaneously using mutate function
 - the code can be more elegant

5.3 Practical

5.3.1 Setup Project

- 1. Setup your project
 - · Open RStudio
 - · Create New Project
- 2. Create Quarto document
 - · update the YAML metadata to make the document self-contained

```
1 ```{r}
2 #| eval: false
3
4 ---
5 title: "Sesi 1 - Data Wrangling"
6 format:
7 html:
8 embed-resources: true
9 ---
10
```

5.3.2 Data Import

- · Commonly data were collected & stored elsewhere, with various format
- Data sources includes:
 - Surveys and questionnaires
 - epidemiological studies
 - governments and health agencies record
 - electronic health records
 - Clinical trials
- Common File Formats
 - CSV (comma-separated values), text file
 - Excel file
 - SPSS, SAS and STATA files
 - JSON (javascript object notion)
 - Databases, SQL
- in R, various package can be use to import data
 - base: import csv file using read.csv(_) function
 - readxl: import excel file using read_excel(_) function
 - haven:
 - * import SAS file using read_sas(_) function
 - * import SPSS file using read_sav(_) function
 - * import STATA file uisng read_dta(_) function

- rio: simplify importing file using import(_) function
- · RStudio User: simplest method is by clicking at the file
- ensure you have the dataset asthma_ds.sav file in your working directory
 - or you can download from file repository

```
1 ```{r}
2 #| eval: false
3
4 download.file(
5 url = "https://github.com/MohdAzmiSuliman/IKU_RBook/raw/master/dataset/asthma_ds.sav",
6 destfile = "asthma_ds.sav", mode = "wb")
7
```

- 2. import the spss file
 - using read_sav(_) and as_factor(_) function from haven:: package

```
library(tidyverse)
library(haven)

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

asthma_ds
```

A tibble: 150 x 16

```
id idR
               Gender
                        Age WorkStatus Height Weight_Pre PA_HW Weight_Post Tx2
   <dbl> <chr> <fct> <dbl> <fct>
                                        <dbl>
                                                    <dbl> <dbl>
                                                                      <dbl> <fct>
       1 nXSw Female
                         34 Unemployed
                                          179
                                                     84.2
                                                                       76.7 Drug~
1
2
       2 yg2t
              Male
                         31 Unemployed
                                          169
                                                     81.8
                                                              1
                                                                       75.7 Plac~
       3 QBW4
 3
              Male
                         25 Employed
                                          164
                                                     88.5
                                                              1
                                                                       84.1 Drug~
 4
      4 2×2S Female
                         33 Unemployed
                                          136
                                                     53.2
                                                              4
                                                                       46.9 Drug~
5
      5 mOnn Male
                         28 Unemployed
                                          172
                                                     71.3
                                                              2
                                                                       63.3 Plac~
      6 D3sl Male
                         33 Unemployed
6
                                          178
                                                     87.3
                                                              0
                                                                       82.6 Drug~
7
       7 le6j Female
                         31 Unemployed
                                          140
                                                     48.8
                                                              5
                                                                       41.9 Plac~
                                                              2
       8 r3gC Female
                         34 Employed
                                          140
                                                     49.1
                                                                       43.6 Plac~
8
9
       9 3Tyt Male
                         31 Employed
                                          171
                                                     60.1
                                                              1
                                                                       54.7 Drug~
                                                                       86.2 Plac~
10
      10 cmKF Male
                         28 Employed
                                          163
                                                     93.1
                                                              1
# i 140 more rows
```

- # i 6 more variables: PEFR_Pre <dbl>, PEFR_Post <dbl>, SxWheeze_Pre <fct>,
- # SxWheeze_Post <fct>, PS_Pre <dbl>, PS_Post <dbl>

5.3.3 Data Wrangling

- All data must have context, and understanding the context of the data is important in data wrangling.
- in this practical the dataset asthma_ds is a simulated (i.e. fake) data, which can be used for 2 research
 - physical activity effect on weight loss
 - * variable: physical activity, body weight (pre & post)
 - bronchodilator effect on asthmatic patient
 - * variable: bronchodilator (placebo, drug A & drug B), PEFR (pre & post), wheezing symptom (pre & post), pain score (pre & post)

5.3.3.1 Select Variable

- · Data can be very large lots of variable (high dimension), and lots of observation (high volume)
- occasionally we may only want to select specific variable
- in this practical, we want to select the variable related to physical activity part
 - idR, Gender, Age, Height, Weight_Pre, PA_HW, Weight_Post
- · and save to a R object

```
1 ```{r}
2 asthma_pa <- asthma_ds %>%
3 select(idR, Gender, Age, Height, Weight_Pre, PA_HW,
4 Weight_Post)
5
6 asthma_pa
7
```

```
# A tibble: 150 x 7
```

```
Age Height Weight_Pre PA_HW Weight_Post
        Gender
  <chr> <fct> <dbl>
                      <dbl>
                                  <dbl> <dbl>
                                                    <dbl>
1 nXSw Female
                  34
                        179
                                  84.2
                                           5
                                                    76.7
                        169
                                  81.8
                                                    75.7
2 yg2t Male
                  31
                                           1
3 QBW4 Male
                  25
                        164
                                  88.5
                                           1
                                                    84.1
4 2×2S Female
                                                    46.9
                  33
                        136
                                  53.2
                                           4
5 mOnn Male
                  28
                        172
                                  71.3
                                           2
                                                    63.3
6 D3sl Male
                  33
                        178
                                  87.3
                                           0
                                                    82.6
7 le6j Female
                                  48.8
                                           5
                                                    41.9
                  31
                        140
8 r3gC Female
                  34
                                  49.1
                                           2
                                                    43.6
                        140
9 3Tyt Male
                  31
                                           1
                                                    54.7
                        171
                                  60.1
                                                    86.2
10 cmKF Male
                  28
                        163
                                  93.1
                                           1
# i 140 more rows
```

• we can also select several consecutive column using : symbol

```
1 ```{r}
2 asthma_tx <- asthma_ds %>%
3 select(idR, Tx2:PS_Post, Gender, Age, Height, Weight_Pre)
4
5 asthma_tx
6
```

A tibble: 150 x 12

	idR	Tx2	PEFR_Pre	PEFR_Post	${\tt SxWheeze_Pre}$	${\tt SxWheeze_Post}$	PS_Pre	PS_Post
	<chr></chr>	<fct></fct>	<dbl></dbl>	<dbl></dbl>	<fct></fct>	<fct></fct>	<dbl></dbl>	<dbl></dbl>
1	nXSw	Drug B	401	452	No	Yes	5	3
2	yg2t	Placebo	477	460	No	Yes	6	4
3	QBW4	Drug B	417	476	No	No	4	4
4	2×2S	Drug A	440	438	Yes	No	6	5
5	mOnn	Placebo	488	509	Yes	No	4	2
6	D3sl	Drug A	428	459	Yes	No	4	2
7	le6j	Placebo	450	480	No	Yes	5	5
8	r3gC	Placebo	393	431	Yes	No	4	2
9	3Tyt	Drug B	424	532	Yes	No	4	4
10	cmKF	Placebo	405	405	Yes	No	5	3
# i	140 r	nore rows	2					

i 4 more variables: Gender <fct>, Age <dbl>, Height <dbl>, Weight_Pre <dbl>

5.3.3.2 Relocate Column

- we can relocate column using relocate function

```
1 ```{r}
2 asthma_ds %>%
3 relocate(Weight_Pre, Weight_Post, PA_HW, .after = Age)
4
5 asthma_ds
6 ```
```

A tibble: 150 x 16

nt Tx2
l> <fct></fct>
79 Drug~
69 Plac~
64 Drug~
36 Drug~
72 Plac~
78 Drug~
40 Plac~
40 Plac~
71 Drug~
13 17 17 14

```
10 cmKF Male
                        28
                                 93.1
                                            86.2
                                                     1 Employed
                                                                     163 Plac~
# i 140 more rows
# i 6 more variables: PEFR_Pre <dbl>, PEFR_Post <dbl>, SxWheeze_Pre <fct>,
   SxWheeze_Post <fct>, PS_Pre <dbl>, PS_Post <dbl>
# A tibble: 150 x 16
     id idR
              Gender Age WorkStatus Height Weight_Pre PA_HW Weight_Post Tx2
   <dbl> <chr> <fct> <dbl> <fct>
                                       <dbl>
                                                 <dbl> <dbl>
                                                                   <dbl> <fct>
      1 nXSw Female
                       34 Unemployed
                                         179
                                                  84.2
                                                           5
                                                                    76.7 Drug~
      2 yg2t Male
                        31 Unemployed
                                                  81.8
                                                                    75.7 Plac~
                                         169
                                                           1
3
      3 QBW4 Male
                        25 Employed
                                        164
                                                  88.5
                                                                    84.1 Drug~
                                                           1
4
      4 2×2S Female 33 Unemployed
                                        136
                                                  53.2
                                                           4
                                                                    46.9 Drug~
                                                                    63.3 Plac~
5
      5 mOnn Male
                      28 Unemployed
                                        172
                                                  71.3
                                                           2
      6 D3sl Male
                       33 Unemployed
                                        178
                                                  87.3
                                                           0
                                                                    82.6 Drug~
7
      7 le6j Female 31 Unemployed
                                         140
                                                  48.8
                                                           5
                                                                    41.9 Plac~
                                                           2
      8 r3gC Female 34 Employed
                                        140
                                                  49.1
                                                                    43.6 Plac~
9
      9 3Tyt Male
                      31 Employed
                                                  60.1
                                                           1
                                                                    54.7 Drug~
                                        171
10
     10 cmKF Male
                        28 Employed
                                        163
                                                  93.1
                                                           1
                                                                    86.2 Plac~
# i 140 more rows
# i 6 more variables: PEFR_Pre <dbl>, PEFR_Post <dbl>, SxWheeze_Pre <fct>,
   SxWheeze_Post <fct>, PS_Pre <dbl>, PS_Post <dbl>
```

• we can also use select function to relocate

```
asthma_ds %>%
select(idR:Age, Weight_Pre, Weight_Post, PA_HW,
everything())
```

```
# A tibble: 150 x 16
```

	idR	Gender	Age	${\tt Weight_Pre}$	Weight_Post	PA_HW	id	WorkStatus	Height	Tx2
	<chr></chr>	<fct></fct>	<dbl></dbl>	<dbl></dbl>	<dbl></dbl>	<dbl></dbl>	<dbl></dbl>	<fct></fct>	<dbl></dbl>	<fct></fct>
1	nXSw	Female	34	84.2	76.7	5	1	Unemployed	179	Drug~
2	yg2t	Male	31	81.8	75.7	1	2	Unemployed	169	Plac∼
3	QBW4	Male	25	88.5	84.1	1	3	Employed	164	Drug~
4	2×2S	Female	33	53.2	46.9	4	4	Unemployed	136	Drug~
5	mOnn	Male	28	71.3	63.3	2	5	Unemployed	172	Plac∼
6	D3sl	Male	33	87.3	82.6	0	6	Unemployed	178	Drug~
7	le6j	Female	31	48.8	41.9	5	7	Unemployed	140	Plac∼
8	r3gC	Female	34	49.1	43.6	2	8	Employed	140	Plac∼
9	3Tyt	Male	31	60.1	54.7	1	9	Employed	171	Drug~
10	cmKF	Male	28	93.1	86.2	1	10	Employed	163	Plac~

[#] i 140 more rows

5.3.3.3 Sorting by Variable

We can also sort our data, using arrange(_) function

[#] i 6 more variables: PEFR_Pre <dbl>, PEFR_Post <dbl>, SxWheeze_Pre <fct>,

[#] SxWheeze_Post <fct>, PS_Pre <dbl>, PS_Post <dbl>

- arrange(_) function sort by ascending value
- to sort by descending value, add desc parameter

```
1 ```{r}
2 asthma_pa %>%
3 arrange(., Height)
4 ```
```

```
# A tibble: 150 x 7
  idR
       Gender
               Age Height Weight_Pre PA_HW Weight_Post
  <chr> <fct> <dbl> <dbl>
                              <dbl> <dbl>
                                              <dbl>
1 uMNM Female
                              52.3
                32
                      129
                                      4
                                               47.4
2 mRn4 Female
                28
                      132
                              63.9
                                      8
                                               55.3
3 ms9j Female
                     135
                              49.6
                                      8
                                              43.4
4 2×2S Female
                33 136
                              53.2
                                      4
                                              46.9
                30 136
5 PuTD Female
                             50.3
                                     1
                                              46.3
6 I79I Female
                29
                             55.7
                                              53.7
                     137
                                     1
7 BRtK Female 32
                             53.6
                                              48.2
                     138
                                     1
8 7Vye Female
                29
                     138
                              45.7
                                      1
                                              40.4
9 XLmR Female
                35
                     139
                              46.9
                                     7
                                              39.3
10 H6JD Female
                32
                      139
                              48.6
                                      1
                                              43.8
# i 140 more rows
```

```
1 ```{r}
2 asthma_pa %>%
3 arrange(., desc(Height))
4 ```
```

```
# A tibble: 150 x 7
       Gender
                Age Height Weight_Pre PA_HW Weight_Post
                              <dbl> <dbl>
  <chr> <fct> <dbl> <dbl>
1 xMFo Male
                29
                      195
                              111.
                                     4
                                               98.5
2 9dDk Male
                              103
                                       6
                                               89.3
                26
                      193
3 W8Tt Male
                25
                              129.
                      192
                                      6
                                              115.
4 jdcP Male
                30
                      190
                              128.
                                      1
                                              120.
5 LUAF Male
                26 190
                              79.8
                                      2
                                               71.2
                                      2
6 1UyJ Male
                33
                      188
                              118.
                                              110.
7 BfV4 Male
                 34
                      188
                              125
                                      0
                                              118.
                34
8 GJyq Male
                      188
                              102.
                                      1
                                               94.1
9 ITU4 Male
                 31
                      187
                              110.
                                       1
                                              101.
10 bBbT Male
                                               97.8
                 29
                      187
                              104.
                                      1
```

5.3.3.4 Filter/Subset Data

i 140 more rows

· We may also wanted to subset our data, analysing data that fulfil our criteria

5 Data Wrangling

- for example, we want to analyse among the male only.
- · filtering data using function filter

```
1 ```{r}
2 asthma_pa %>%
3 filter(., Age ≥ 15 & Age ≤ 64)
4 ```
```

```
# A tibble: 150 x 7
  idR Gender
                Age Height Weight_Pre PA_HW Weight_Post
  <chr> <fct> <dbl> <dbl>
                               <dbl> <dbl>
                                                <dbl>
1 nXSw Female
                                84.2
                 34
                       179
                                        5
                                                76.7
2 yg2t Male
                 31
                      169
                                81.8
                                        1
                                                75.7
3 QBW4 Male
                 25
                      164
                                88.5
                                        1
                                                84.1
4 2×2S Female 33
                                53.2
                                                46.9
                     136
                                        4
5 mOnn Male
                                71.3
                                        2
                                                63.3
                 28
                      172
6 D3sl Male
                 33
                    178
                                87.3
                                        0
                                                82.6
                                                41.9
7 le6j Female
                31 140
                                48.8
                                        5
8 r3gC Female
                 34
                      140
                                49.1
                                        2
                                                43.6
9 3Tyt Male
                 31
                      171
                                60.1
                                        1
                                                54.7
10 cmKF Male
                 28
                       163
                                93.1
                                        1
                                                86.2
# i 140 more rows
```

5.3.3.5 Recode Data

- function fct_recode is used to recode variable
 - for example, we want to recode Drug A and Drug B into Treat group and Placebo into Control group

```
# A tibble: 150 x 3
   idR Tx2 Tx1
   <chr> <fct> <fct> <fct>
1 nXSw Drug B Treat
2 yg2t Placebo Control
3 QBW4 Drug B Treat
4 2×2S Drug A Treat
5 mOnn Placebo Control
```

```
6 D3sl Drug A Treat
7 le6j Placebo Control
8 r3gC Placebo Control
9 3Tyt Drug B Treat
10 cmKF Placebo Control
# i 140 more rows
```

- function fct_recode(_) also used to recode numerical to categorical variable
 - fct_recode(_) can only recode factor or string. so we have to convert numerical
 variable to string using as.character(_) function
 - for example, we want to recode pain score 6 to high, 5 to moderate, 4 to tolerable

```
# A tibble: 150 x 3
         PS_Pre PS_PreCat
   idR
   <chr> <dbl> <fct>
 1 nXSw
              5 Moderate
 2 yg2t
              6 High
3 QBW4
              4 Tolerable
4 2×2S
              6 High
              4 Tolerable
5 mOnn
 6 D3sl
              4 Tolerable
              5 Moderate
7 le6j
8 r3gC
              4 Tolerable
9 3Tyt
              4 Tolerable
              5 Moderate
10 cmKF
# i 140 more rows
```

5.3.3.6 Factor Level Order

- R recognize unordered string, and ordered factor
- · commonly, we want to reorder the factor, for example we want female first, then male
- to reorder factor level, we will use function fct_relevel(_) from tidyverse:: package

```
i Note
to check level, use str(_) function or levels(_) function
```

```
```{r}
levels(asthma_ds$Gender)
```

#### [1] "Male" "Female"

```
```{r}
asthma_ds <- asthma_ds %>%
  mutate(Gender2 = fct_relevel(Gender, "Female", "Male"),
          .after = Gender)
asthma_ds
levels(asthma_ds$Gender2)
```

```
# A tibble: 150 x 17
```

```
id idR
              Gender Gender2
                               Age WorkStatus Height Weight_Pre PA_HW
  <dbl> <chr> <fct> <fct> <dbl> <fct>
                                               <dbl>
                                                         <dbl> <dbl>
      1 nXSw Female Female
                                34 Unemployed
                                                 179
                                                          84.2
                                                                   5
1
2
      2 yg2t Male
                     Male
                                31 Unemployed
                                                 169
                                                           81.8
                                                                   1
3
      3 QBW4 Male
                     Male
                                25 Employed
                                                 164
                                                          88.5
                                                                   1
4
      4 2×2S Female Female
                                33 Unemployed
                                                          53.2
                                                 136
                                                                   4
5
      5 mOnn Male
                     Male
                                                          71.3
                                                                   2
                                28 Unemployed
                                                 172
                     Male
                                33 Unemployed
                                                          87.3
6
      6 D3sl Male
                                                 178
                                                                   0
7
      7 le6j Female Female
                                31 Unemployed
                                                 140
                                                          48.8
                                                                   5
8
      8 r3gC Female Female
                                34 Employed
                                                 140
                                                          49.1
                                                                   2
9
      9 3Tyt Male
                     Male
                                31 Employed
                                                 171
                                                           60.1
                                                                   1
10
     10 cmKF Male
                     Male
                                28 Employed
                                                 163
                                                          93.1
                                                                   1
# i 140 more rows
# i 8 more variables: Weight_Post <dbl>, Tx2 <fct>, PEFR_Pre <dbl>,
```

```
PEFR_Post <dbl>, SxWheeze_Pre <fct>, SxWheeze_Post <fct>, PS_Pre <dbl>,
PS_Post <dbl>
```

[1] "Female" "Male"

5.3.3.7 Group combine (collapse factor)

- combine group using fct collapse() function
- we will recode treatment group to intervention group (tx1), in which
 - Drug A and Drug B collapse into intervention
 - placebo collapse into control

```
```{r}
asthma_tx %>%
 select(idR, Tx2) %>%
```

```
A tibble: 150 x 3
 idR
 Tx2
 Tx1
 <chr> <fct> <fct>
1 nXSw Drug B Intervention
2 yg2t Placebo Control
3 QBW4 Drug B Intervention
4 2×2S Drug A Intervention
5 mOnn Placebo Control
6 D3sl Drug A Intervention
7 le6j Placebo Control
8 r3gC Placebo Control
9 3Tyt Drug B Intervention
10 cmKF Placebo Control
i 140 more rows
```

#### 5.3.3.8 Basic Numerical Transformation

- R accept basic mathematical operator for data transformation
  - for example, to transform the height from cm to m.

```
asthma_pa <- asthma_pa %>%
mutate(Ht_m = Height/100, .after = Height)
asthma_pa
```

#### # A tibble: 150 x 8

```
idR Gender
 Age Height Ht_m Weight_Pre PA_HW Weight_Post
 <chr> <fct> <dbl> <dbl> <dbl>
 <dbl> <dbl>
 <dbl>
 76.7
1 nXSw Female
 179 1.79
 84.2
 5
 34
 31
2 yg2t Male
 169 1.69
 81.8
 1
 75.7
3 QBW4 Male
 25 164 1.64
 88.5
 1
 84.1
4 2×2S Female
 33
 136 1.36
 53.2
 4
 46.9
 71.3
 2
5 mOnn Male
 28 172 1.72
 63.3
 33 178 1.78
6 D3sl Male
 87.3
 0
 82.6
 31
 140 1.4
 48.8
 5
7 le6j Female
 41.9
8 r3gC Female 34 140 1.4
 49.1
 2
 43.6
 31
9 3Tvt Male
 171 1.71
 60.1 1
 54.7
 93.1 1
10 cmKF Male
 28
 163 1.63
 86.2
```

#### 5 Data Wrangling

#### # i 140 more rows

• we can also transform based on more than single column

```
1 ```{r}
2 asthma_pa <- asthma_pa %>%
3 mutate(BMI_Pre = Weight_Pre/(Ht_m^2), .after = Weight_Pre)
4
5 asthma_pa
6 ```
```

```
A tibble: 150 x 9
```

```
Gender
 Age Height Ht_m Weight_Pre BMI_Pre PA_HW Weight_Post
 <chr> <fct> <dbl> <dbl> <dbl>
 <dbl>
 <dbl> <dbl>
 <dbl>
1 nXSw Female
 84.2
 26.3
 5
 76.7
 34
 179 1.79
2 yg2t Male
 169 1.69
 81.8
 28.6
 1
 75.7
 31
3 QBW4 Male
 25
 164 1.64
 88.5
 32.9
 1
 84.1
4 2×2S Female
 33
 136 1.36
 53.2 28.8
 4
 46.9
5 mOnn Male
 28
 172 1.72
 71.3
 24.1
 2
 63.3
6 D3sl Male
 33
 178 1.78
 87.3 27.6
 0
 82.6
 48.8
 41.9
7 le6j Female
 140 1.4
 24.9
 5
 31
 49.1
 2
8 r3gC Female
 34
 140 1.4
 25.1
 43.6
9 3Tvt Male
 31
 171 1.71
 60.1
 20.6
 1
 54.7
10 cmKF Male
 28
 163 1.63
 93.1
 35.0
 1
 86.2
```

# i 140 more rows

· to many digits? we can round them

```
1 ```{r}
2 asthma_pa <- asthma_pa %>%
3 mutate(BMI_Pre = round(BMI_Pre, 1))
4
5 asthma_pa
6 ```
```

```
A tibble: 150 x 9
```

```
Gender
 Age Height Ht_m Weight_Pre BMI_Pre PA_HW Weight_Post
 <chr> <fct> <dbl> <dbl> <dbl>
 <dbl>
 <dbl> <dbl>
 <dbl>
1 nXSw Female
 179 1.79
 84.2
 26.3
 5
 76.7
 34
2 yg2t Male
 169 1.69
 81.8
 28.6
 1
 75.7
 31
3 QBW4 Male
 25
 164 1.64
 88.5
 32.9
 1
 84.1
4 2×2S Female
 33
 136 1.36
 53.2 28.8
 46.9
 4
5 mOnn Male
 28
 172 1.72
 71.3
 24.1
 2
 63.3
 87.3 27.6
6 D3sl Male
 33
 178 1.78
 0
 82.6
7 le6j Female
 31
 140 1.4
 48.8
 24.9
 5
 41.9
 34
 49.1
 2
8 r3gC Female
 140 1.4
 25.1
 43.6
9 3Tyt Male
 31
 171 1.71
 60.1
 20.6
 1
 54.7
10 cmKF Male
 28
 163 1.63
 93.1
 35
 1
 86.2
i 140 more rows
```

#### 5.3.3.9 Binning: Categorizing Numerical Variable

- · apart from numerical transformation, we can also bin the numerical variable
- for example, we want to categorized the BMI to asian category

```
underweight: < 18.5
normal: 18.5 - 22.9
overweight: 23.0 - 24.9
obese: >= 25.0
```

```
A tibble: 150 x 10
 Age Height Ht_m Weight_Pre BMI_PreCat PA_HW
 idR Gender
 <chr> <fct> <dbl> <dbl> <dbl>
 <dbl>
 <dbl> <fct>
 <dbl>
 84.2 26.3 Obese
1 nXSw Female
 34
 179 1.79
 5
2 yg2t Male
 31
 169 1.69
 81.8 28.6 Obese
 1
 25
 88.5
 32.9 Obese
 1
3 QBW4 Male
 164 1.64
4 2×2S Female 33 136 1.36
 53.2 28.8 Obese
 4
 24.1 Overweight
5 mOnn Male
 28
 172 1.72
 71.3
6 D3sl Male
 33 178 1.78
 87.3 27.6 Obese
7 le6j Female
 31
 48.8
 24.9 Overweight
 140 1.4
 5
 34
 25.1 Obese
 2
8 r3gC Female
 140 1.4
 49.1
 31
 20.6 Normal
9 3Tyt Male
 171 1.71
 60.1
 1
10 cmKF Male
 28
 163 1.63
 93.1
 35 Obese
i 140 more rows
```

# i 1 more variable: Weight\_Post <dbl>

- 5.3.3.10 Reshape Dataset
  - Data frame structure: Rows and columns represent observations and variables.
    - Rows: Individual records or observations.
    - Columns: Different attributes or variables.
  - Wide Format: Single rows, multiple measurement columns.
    - Each row: One subject/sample.

#### 5 Data Wrangling

- Columns: Different time points or conditions.
- Long Format: Repeated rows, time/condition column.
  - Multiple rows: Same subject/sample.
  - Specific column: Indicates time or condition.
- 1. In this example, we will use other dataset, asthma24\_wide.sav
  - download the dataset

```
1 ```{r}
2 #| eval: false
3
4 download.file(
5 url = "https://github.com/MohdAzmiSuliman/IKU_RBook/raw/master/dataset/asthma24_wide.sav",
6 destfile = "asthma24_wide.sav", mode = "wb")
7
```

- 2. Import the dataset to R
  - use read\_sav(\_) function to import SPSS file and as\_factor(\_) function to convert the code with the label.

```
1 ```{r}
2 asthma_wide <- read_sav("asthma24_wide.sav") %>%
3 as_factor()
4
5 asthma_wide
6 ```
```

```
A tibble: 24 x 14
```

```
id idR Gender
 Age WorkStatus Height Weight Tx2
 PEFR_Pre PEFR_Post
 <dbl> <chr> <fct> <dbl> <fct>
 <dbl> <dbl> <fct>
 <dbl>
 <dbl>
 82 jTPy Male
 25 Employed
 67.7 Placebo
1
 174
 463
 464
2
 41 rdbg Female
 26 Employed
 177 103
 Drug A
 398
 499
3
 58 H6JD Female
 32 Unemployed
 395
 139
 48.6 Drug A
 415
4
 54 1UyJ Male
 33 Employed
 188 118. Drug B
 497
 571
5
 61 zPxl Female
 29 Unemployed
 50.2 Drug A
 489
 146
 437
6
 134 BRPo Male
 33 Unemployed
 184
 92
 Drug B
 462
 557
7
 43 Hrnv Female
 26 Employed
 154
 45.4 Placebo
 376
 360
8
 34 Unemployed
 168
 87.9 Placebo
 456
 469
 104 muyj Male
9
 108 uMNM Female
 32 Unemployed
 129
 52.3 Placebo
 431
 447
 124 8JqP Female
 30 Employed
10
 154
 49.2 Drug A
 425
 487
```

- # i 14 more rows
- # i 4 more variables: SxWheeze\_Pre <fct>, SxWheeze\_Post <fct>, PS\_Pre <dbl>,
- # PS\_Post <dbl>
  - 3. Reshape the data frame, into long format (i.e., single column for single variable, with a specific column indicating time)

• in this example, we want to convert PEFR\_Pre and PEFR\_Post, from two separate column, into one single column, with new column indicate time.

```
A tibble: 48 x 6
 idR
 Gender
 PEFR
 Age Tx2
 Time
 <chr> <fct> <dbl> <fct>
 <chr>>
 <dbl>
1 jTPy Male
 25 Placebo PEFR_Pre
 463
2 jTPy
 Male
 25 Placebo PEFR_Post
 464
3 rdbg
 Female
 26 Drug A PEFR_Pre
 398
4 rdbg Female
 26 Drug A PEFR_Post
 499
5 H6JD Female
 32 Drug A
 PEFR_Pre
 395
6 H6JD Female
 32 Drug A PEFR_Post
 415
7 1UyJ Male
 33 Drug B PEFR_Pre
 497
8 1UyJ Male
 33 Drug B
 PEFR Post
 571
9 zPxl Female
 29 Drug A
 PEFR_Pre
 437
10 zPxl Female
 29 Drug A PEFR_Post
 489
i 38 more rows
```

- 4. We can also shape the data frame, into wide format (i.e., single row for single participant, with multiple column represent repeated variables)
  - in this example, we will use long format data in previous example

#### # A tibble: 24 x 6

```
idR
 Gender
 PEFR_Pre PEFR_Post
 Age Tx2
 <chr> <fct> <dbl> <fct>
 <dbl>
 <dbl>
 25 Placebo
1 jTPy Male
 463
 464
2 rdbg Female
 26 Drug A
 398
 499
3 H6JD Female
 32 Drug A
 395
 415
4 1UyJ Male
 33 Drug B
 497
 571
```

#### 5 Data Wrangling

```
5 zPxl Female
 29 Drug A
 437
 489
6 BRPo Male
 33 Drug B
 462
 557
7 Hrnv Female
 26 Placebo
 376
 360
8 muyj Male
 34 Placebo
 456
 469
 32 Placebo
9 uMNM Female
 431
 447
10 8JqP Female
 30 Drug A
 425
 487
i 14 more rows
```

#### 5.3.3.11 Join Dataset

- Sometimes, you may want to join several dataset
- For example, the measurements (PEFR, SxWheeze & PS) were measured twice (Pre & Post), but were stored in different file. they have anonymized identifier that can be use to matched them.
- joining dataset using \_\_join(\_)
- 1. Download the dataset in working directory

#### 2. Load the dataset and save in R objects

```
asthma_pre <- read_sav("asthma_pre.sav") %>%
as_factor() %>%
select(id, idR, PEFR_Pre, SxWheeze_Pre, PS_Pre)

asthma_pre

asthma_pre

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

asthma_post
```

```
3
 3 QBW4
 417 No
 4 2×2S
 440 Yes
4
5
 5 mOnn
 488 Yes
 6 D3sl
 428 Yes
6
 4
7
 450 No
 5
 7 le6j
 393 Yes
8
 8 r3gC
 4
9
 9 3Tyt
 424 Yes
 4
 405 Yes
10
 10 cmKF
 5
i 140 more rows
A tibble: 150 x 5
 id idR PEFR_Post SxWheeze_Post PS_Post
 <dbl> <chr>
 <dbl> <fct>
 <dbl>
1
 1 nXSw
 452 Yes
 3
 460 Yes
 4
2
 2 yg2t
 3 QBW4
 476 No
 4
3
4
 4 2×2S
 438 No
 5
5
 5 mOnn
 509 No
 2
 6 D3sl
 459 No
 2
6
7
 7 le6j
 480 Yes
 5
 2
8
 8 r3gC
 431 No
9
 9 3Tyt
 532 No
 4
 10 cmKF
 405 No
 3
10
i 140 more rows
```

3. Joining dataset using \_\_join(\_) function

```
1 ```{r}
2 left_join(asthma_pre, asthma_post, by = "idR")
3 ```
```

```
A tibble: 150 x 9
```

```
id.x idR PEFR_Pre SxWheeze_Pre PS_Pre id.y PEFR_Post SxWheeze_Post
 <dbl> <chr> <dbl> <fct> <dbl> <dbl> <dbl> <
 <dbl> <fct>
1
 1 nXSw
 401 No
 5
 1
 452 Yes
 2 yg2t
 477 No
 6
 2
 460 Yes
2
3
 3 QBW4
 417 No
 4
 3
 476 No
 4 2×2S
 440 Yes
 4
4
 6
 438 No
5
 5 mOnn
 488 Yes
 509 No
 6 D3sl
 428 Yes
6
 4
 6
 459 No
7
 450 No
 5
 7
 7 le6j
 480 Yes
8
 8 r3gC
 393 Yes
 4
 8
 431 No
9
 9 3Tyt
 424 Yes
 4
 9
 532 No
10
 10 cmKF
 405 Yes
 5 10
 405 No
i 140 more rows
```

# i 1 more variable: PS\_Post <dbl>

#### 5.3.4 Save/Export Dataset

- R allow to save/export file to various format
  - CSV file: flexible and cross platforms
  - RDS & RData file: specific to R
  - XLSX, SPSS's SAV, Stata's DTA, etc
  - JSON, XML

#### 5.3.5 Data Exploration

- · understanding our data
  - data structure and content
  - summary statistics
  - visualization
  - identifying relationship and correlation
- iterative and interactive process
  - post-import
  - during data wrangling
  - before analysis

#### 5.3.5.1 Examine Structure

• dataset structure can be view using str function

```
tibble [150 x 17] (S3: tbl_df/tbl/data.frame)
$ id
 : num [1:150] 1 2 3 4 5 6 7 8 9 10 ...
 ..- attr(*, "format.spss")= chr "F8.0"
 : chr [1:150] "nXSw" "yg2t" "QBW4" "2×2S" ...
 ..- attr(*, "format.spss")= chr "A4"
 : Factor w/ 2 levels "Male", "Female": 2 1 1 2 1 1 2 2 1 1 ...
$ Gender2
 : Factor w/ 2 levels "Female", "Male": 1 2 2 1 2 2 1 1 2 2 ...
 : num [1:150] 34 31 25 33 28 33 31 34 31 28 ...
$ Age
 .. - attr(*, "label")= chr "Age (year)"
 .. - attr(*, "format.spss")= chr "F8.2"
$ WorkStatus : Factor w/ 2 levels "Employed", "Unemployed": 2 2 1 2 2 2 2 1 1 1 ...
 .. - attr(*, "label")= chr "Employment"
 : num [1:150] 179 169 164 136 172 178 140 140 171 163 ...
$ Height
 .. - attr(*, "label")= chr "Height (cm)"
 ..- attr(*, "format.spss")= chr "F8.2"
$ Weight_Pre : num [1:150] 84.2 81.8 88.5 53.2 71.3 87.3 48.8 49.1 60.1 93.1 ...
 .. - attr(*, "label")= chr "Weight (kg) - before"
```

```
.. - attr(*, "format.spss")= chr "F8.2"
 : num [1:150] 5 1 1 4 2 0 5 2 1 1 ...
 $ PA HW
 .. - attr(*, "label")= chr "Physical Activity (total hour per week)"
 .. - attr(*, "format.spss")= chr "F8.2"
$ Weight_Post : num [1:150] 76.7 75.7 84.1 46.9 63.3 82.6 41.9 43.6 54.7 86.2 ...
 .. - attr(*, "label")= chr "Weight (kg) - after"
 .. - attr(*, "format.spss")= chr "F8.2"
 $ Tx2
 : Factor w/ 3 levels "Placebo", "Drug A", ..: 3 1 3 2 1 2 1 1 3 1 ...
 .. - attr(*, "label")= chr "Treatment Group"
 $ PEFR_Pre
 : num [1:150] 401 477 417 440 488 428 450 393 424 405 ...
 ..- attr(*, "label")= chr "PEFR (L/min) - before"
 ..- attr(*, "format.spss")= chr "F8.2"
 $ PEFR_Post : num [1:150] 452 460 476 438 509 459 480 431 532 405 ...
 .. - attr(*, "label")= chr "PEFR (L/min) - after"
 ..- attr(*, "format.spss")= chr "F8.2"
 $ SxWheeze_Pre : Factor w/ 2 levels "No", "Yes": 1 1 1 2 2 2 1 2 2 2 ...
 .. - attr(*, "label")= chr "Symptom Wheezing - before"
$ SxWheeze_Post: Factor w/ 2 levels "No", "Yes": 2 2 1 1 1 1 2 1 1 1 ...
 ..- attr(*, "label")= chr "Symptom Wheezing - after"
 : num [1:150] 5 6 4 6 4 4 5 4 4 5 ...
 ..- attr(*, "label")= chr "Pains Score - before"
 ..- attr(*, "format.spss")= chr "F8.0"
 : num [1:150] 3 4 4 5 2 2 5 2 4 3 ...
$ PS_Post
 ..- attr(*, "label")= chr "Pains Score - after"
 .. - attr(*, "format.spss")= chr "F8.0"
 ```{r}
str(asthma_ds$Gender)
```

Factor w/ 2 levels "Male", "Female": 2 1 1 2 1 1 2 2 1 1 ...

5.3.5.2 Summarise data

simplest way to have summary of the data, is use summary(_) function

```
1 ```{r}
2 summary(asthma_ds)
3 ```
```

id	idR	Gender	Gender2	Age
Min. : 1.00	Length:150	Male :86	Female:64	Min. :25.00
1st Qu.: 38.25	Class :character	Female:64	Male :86	1st Qu.:27.00
Median : 75.50	Mode :character			Median :30.00
Mean : 75.50				Mean :30.15
3rd Qu.:112.75				3rd Qu.:33.00

```
:150.00
                                                               :35.00
Max.
                                                         Max.
                              Weight Pre
    WorkStatus
                   Height
                                                  PA HW
Employed :70
               Min. :129.0
                              Min. : 36.60 Min. : 0.000
               1st Qu.:151.0
                              1st Qu.: 58.25
                                              1st Qu.: 1.000
Unemployed:80
               Median :167.0
                              Median : 77.70
                                              Median : 2.000
               Mean
                    :164.0
                              Mean : 75.89
                                              Mean : 2.807
               3rd Qu.:175.8
                              3rd Qu.: 89.45
                                             3rd Qu.: 4.000
               Max. :195.0
                              Max. :129.10 Max. :10.000
Weight_Post
                     Tx2
                               PEFR_Pre
                                             PEFR_Post
                                                           SxWheeze_Pre
Min. : 31.20
                Placebo:72
                            Min. :364.0 Min. :344.0
                                                          No :62
1st Qu.: 52.08
                Drug A:35
                            1st Qu.:401.2
                                           1st Qu.:433.2
                                                          Yes:88
Median : 70.75
                            Median :431.5 Median :463.0
                Drug B:43
Mean : 68.55
                            Mean :433.2 Mean :465.6
3rd Qu.: 82.08
                            3rd Qu.:461.0
                                           3rd Qu.:505.5
Max. :119.60
                            Max. :516.0
                                           Max. :580.0
SxWheeze_Post
                               PS_Post
                 PS_Pre
                   :4.000
No :99
                            Min.
                                   :2.000
             Min.
Yes:51
             1st Qu.:4.000
                            1st Qu.:3.000
             Median :5.000
                            Median :4.000
             Mean
                  :4.867
                            Mean :3.993
             3rd Qu.:6.000
                            3rd Qu.:5.000
                   :6.000
                            Max. :6.000
             Max.
 ```{r}
 summary(asthma_ds$Gender)
Male Female
 86
 64
```{r}
 summary(asthma ds$Age)
                        Mean 3rd Qu.
 Min. 1st Qu.
               Median
                                        Max.
 25.00
        27.00
                30.00
                       30.15
                               33.00
                                       35.00
```

5.4 Addin: questionr:: package

questionr :: package is a rstudio addin that is useful for data wrangling

```
i ```{r}
if (!require("questionr", character.only = TRUE)) {
   install.packages("questionr")
}
```

. . . .

after install the addin, it will appear in the addin list

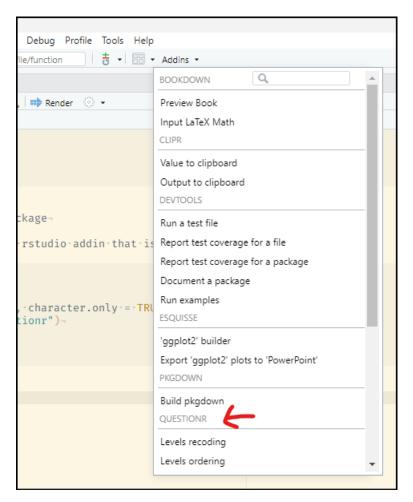


Figure 5.2: questionr:: addin

the usage of addin is intuitive and self-explanatory

One of the strengths of R is its capability for data visualization. While base R itself offers robust visualization and graphing capabilities, the open-source nature of R encourages the development and use of various packages to extend its visualization potential. Among these, the ggplot2:: package stands out for its powerful and versatile graphing functions, making it a popular choice among data scientists and statisticians.

Although R is a programming language with a syntax that can be complex for beginners, the esquisse:: package provides a user-friendly interface to facilitate graph plotting. By simplifying the process of creating visualizations, esquisse:: allows users to generate plots more intuitively, making data visualization accessible to those less familiar with R's syntax.

In this chapter, we will demonstrate examples of several types of graphs, showcasing the breadth of R's visualization capabilities. Through practical examples, readers will learn how to leverage packages like ggplot2:: and tools like esquisse:: to transform data into compelling visual narratives. Whether you're visualizing trends, distributions, relationships, or comparisons, R provides the tools necessary to produce informative and aesthetically pleasing graphics.

Some of the commong graph here are

- Single Variable
 - histogram
 - boxplot
 - bar chart
 - scatter plot
 - line chart

6.1 Practical - 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

```
title: "Sesi 2 - Data Vizualization"
format:
html:
embed-resources: true
```

3. Download the dataset

• we will use asthmads_clean.sav data set

```
```{r}
 # | eval: false
 download.file(
 url = "https://github.com/MohdAzmiSuliman/IKU_RBook/raw/master/dataset/asthmads_clean.sav",
 destfile = "asthmads_clean.sav", mode = "wb")
 library(tidyverse)
 library(haven)
 asthmads <- read_sav("asthmads_clean.sav") %>%
 as_factor()
 asthmads
A tibble: 150 x 22
 id idR Gender
 Age WorkStatus Height Ht_m Weight_Pre BMI_Pre
 <dbl> <chr> <fct> <dbl> <fct>
 <dbl> <dbl>
 <dbl>
 <dbl>
 1 nXSw Female 34 Unemployed
 179 1.79
 84.2
 26.3
 28.6
2
 2 yg2t Male
 31 Unemployed
 169 1.69
 81.8
 3 QBW4 Male
 25 Employed
 164 1.64
 88.5
 32.9
 4 2×2S Female 33 Unemployed
 136 1.36
 53.2 28.8
 28 Unemployed
5
 5 mOnn Male
 172 1.72
 71.3
 24.1
 33 Unemployed
 27.6
6
 6 D3sl Male
 178 1.78
 87.3
7
 7 le6j Female 31 Unemployed
 140 1.4
 48.8 24.9
 8 r3gC Female 34 Employed
8
 140 1.4
 49.1
 25.0
9
 9 3Tyt Male
 31 Employed
 171 1.71
 60.1 20.6
 10 cmKF Male
 28 Employed
 163 1.63
 93.1
 35.0
10
i 140 more rows
i 13 more variables: BMI_PreCat <fct>, PA_HW <dbl>, Weight_Post <dbl>,
 BMI_Post <dbl>, BMI_PostCat <fct>, Tx1 <fct>, Tx2 <fct>, PEFR_Pre <dbl>,
```

# 6.2 esquisse:: RStudio Addin

# 6.3 Plot Examples

PS Post <dbl>

Example of plot/graph will be shown first, then when we have rough idea of how ggplot is plotted.

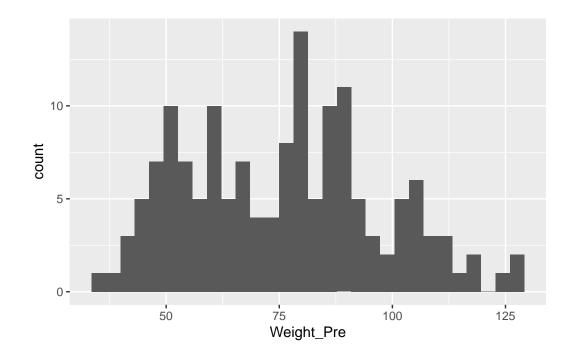
PEFR\_Post <dbl>, SxWheeze\_Pre <fct>, SxWheeze\_Post <fct>, PS\_Pre <dbl>,

#

# 6.3.1 Histogram

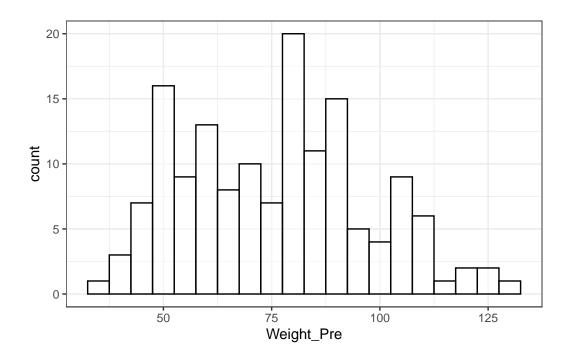
Histogram suitable to show distribution of single numerical variable, such as the body weight distribution

```
1 ```{r}
2 asthmads %>%
3 ggplot(aes(Weight_Pre)) +
4 geom_histogram()
```



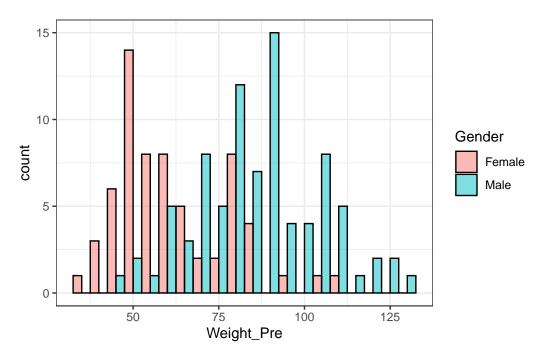
we can customise it to make it prettier

```
asthmads %>%
ggplot(aes(Weight_Pre)) +
geom_histogram(binwidth = 5, alpha = 0.5,
fill = "white", colour = "black") +
theme_bw()
```



we can also stratified the graph, for example, by gender

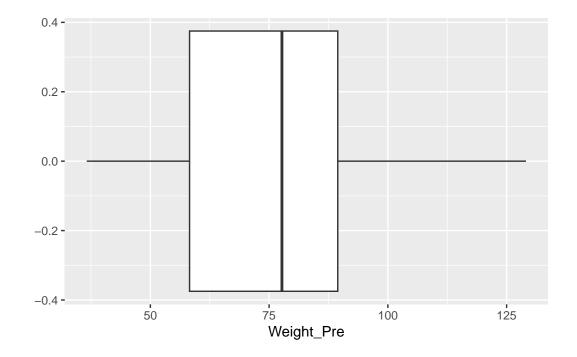
```
asthmads %>%
ggplot(aes(Weight_Pre)) +
geom_histogram(aes(fill = Gender),
position = position_dodge(),
binwidth = 5, alpha = 0.5,
colour = "black") +
theme_bw()
```



# 6.3.2 Boxplot

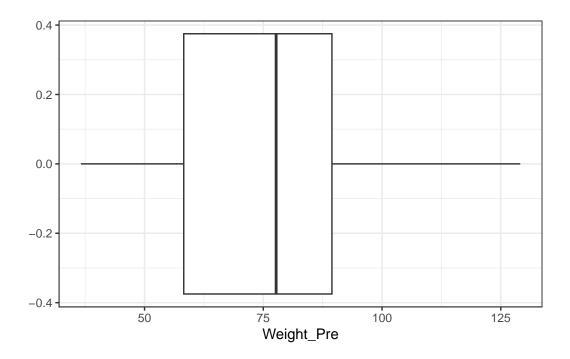
Boxplot also a common graph use to visualize single numerical variable

```
1 ```{r}
2 asthmads %>%
3 ggplot(aes(Weight_Pre)) +
4 geom_boxplot()
5
```



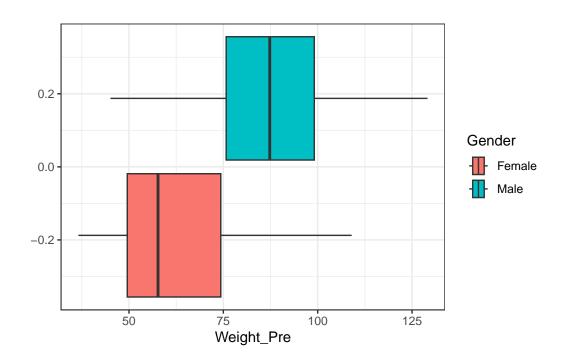
we can customise it to make it prettier

```
asthmads %>%
ggplot(aes(Weight_Pre)) +
geom_boxplot() +
theme_bw()
```



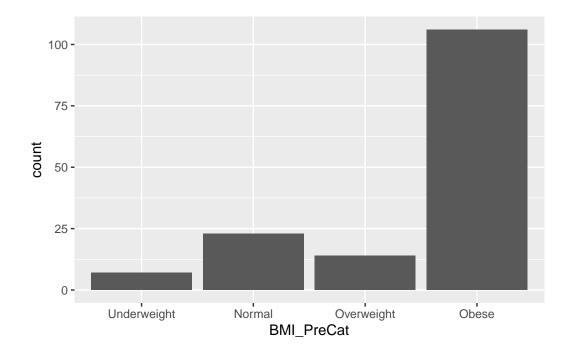
we can also stratified the graph, for example, by gender

```
1 asthmads %>%
2 ggplot(aes(Weight_Pre)) +
3 geom_boxplot(aes(fill = Gender)) +
4 theme_bw()
```



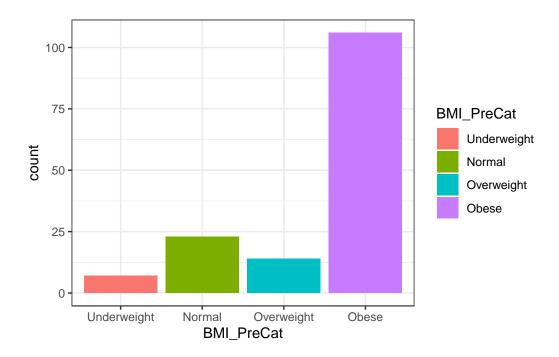
### 6.3.3 Bar Chart

Similar to histogram, bar chart suitable to show distribution of single categorical variable, such as BMI categories. the bar in histogram is continuous, but in bar chart, has gap.



we can customise it to make it prettier

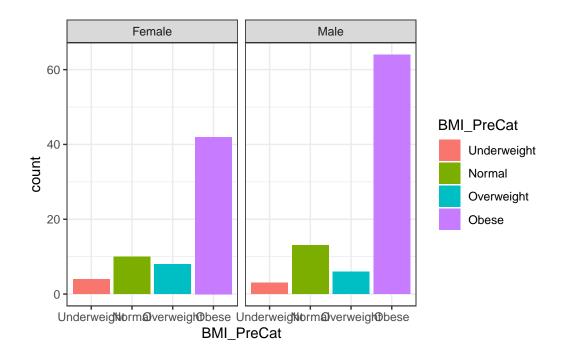
```
asthmads %>%
ggplot(aes(BMI_PreCat)) +
geom_bar(aes(fill = BMI_PreCat)) +
theme_bw()
```



there are various way if we want to stratified this.

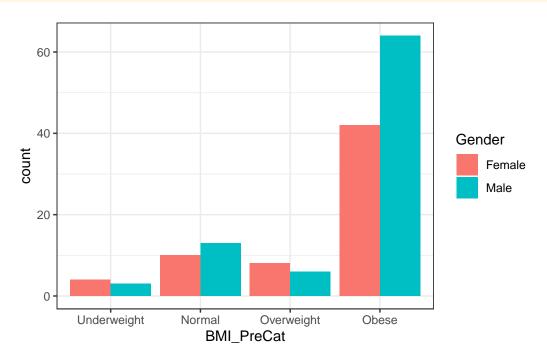
- we can stratified by facet $\_$ wrap function

```
1 ```{r}
2 asthmads %>%
3 ggplot(aes(BMI_PreCat)) +
4 geom_bar(aes(fill = BMI_PreCat)) +
5 facet_wrap(~ Gender) +
6 theme_bw()
7
```



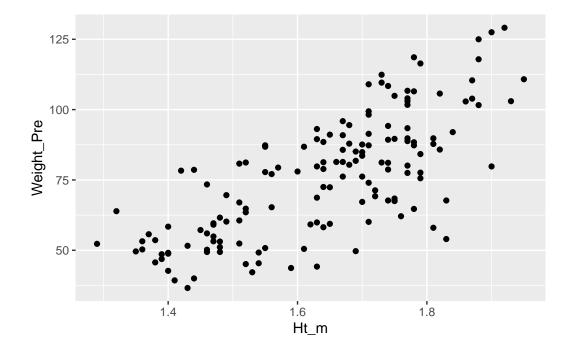
 $\cdot$  we can stratified by position dodge

```
1 ```{r}
2 asthmads %>%
3 ggplot(aes(BMI_PreCat)) +
4 geom_bar(aes(fill = Gender), position = position_dodge()) +
5 theme_bw()
6
```

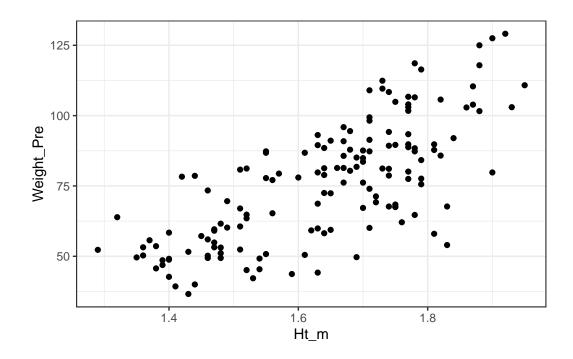


# 6.3.4 Scatter Plot

Scatter plot suitable to visualize two numerical variables together, for example, we want to see the relationship between body weight and height.

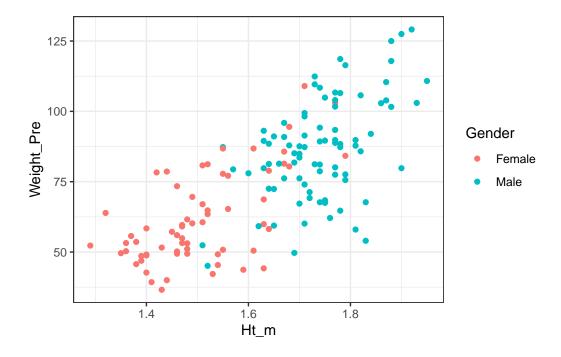


we can customise it so make it pretty

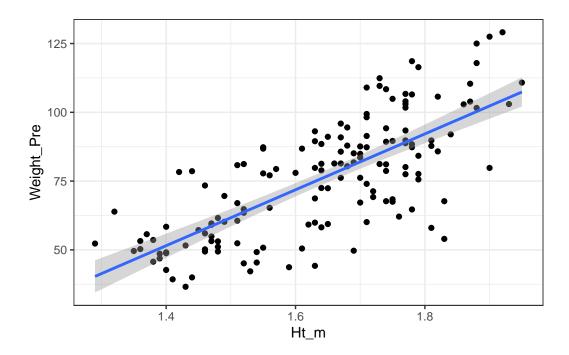


# we can also stratified by gender

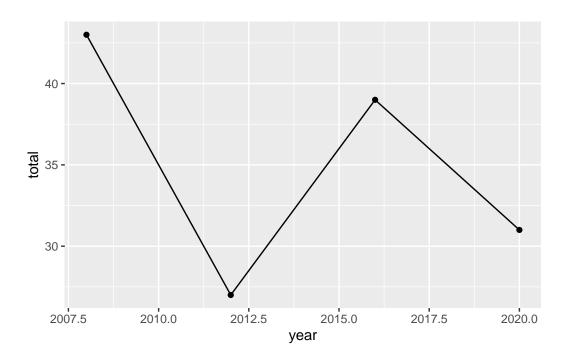
```
1 ```{r}
2 asthmads %>%
3 ggplot(aes(x = Ht_m, y = Weight_Pre)) +
4 geom_point(aes(colour = Gender)) +
5 theme_bw()
6 ```
```



and even add best fit line (pearson correlation)

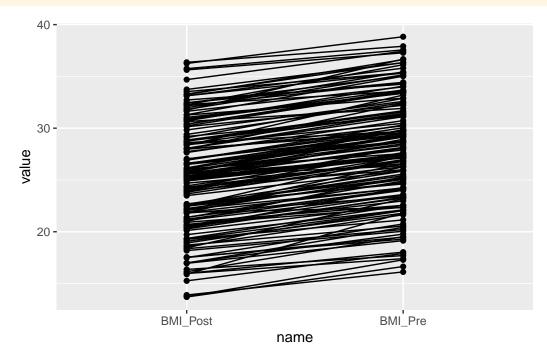


### 6.3.5 Line Chart



# 6.3.6 Spaghetti Plot

```
asthmads %>%
select(id, BMI_Pre, BMI_Post) %>%
pivot_longer(cols = c(BMI_Pre, BMI_Post)) %>%
ggplot(aes(x = name, y = value)) +
geom_point() +
geom_line(aes(group = id))
```



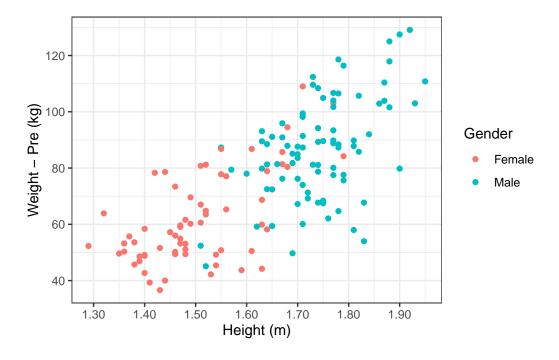
# **6.4** ggplot2::

ggplot2:: implements the grammar of graphics, a coherent system for describing and building graph. We will use scatterplot as example, in understanding ggplot.

Source: https://r4ds.hadley.nz/data-visualize

Example: What is the relationship between body weight and height?

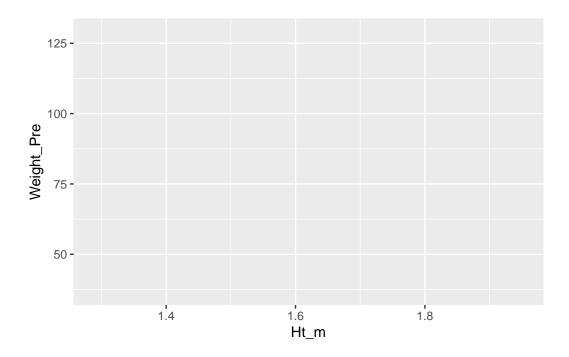
The Ultimate Goal



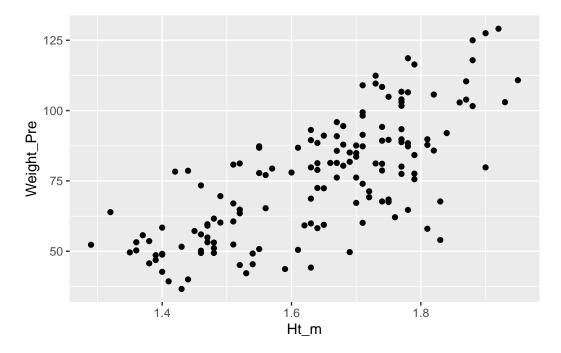
1. ggplot function i.e, creating the "canvas"

```
1 ```{r}
2 ggplot(data = asthmads)
3
```

- 2. map the visual properties
  - specifying the x axis and y axis in the respective arguement



# 3. Define the geometrical object to represent the data



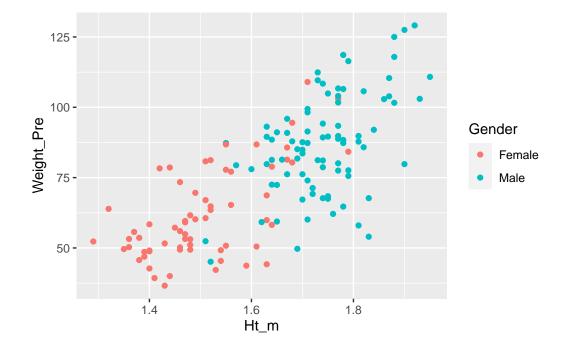
# Important

do note that in ggplot, the plotting functions (e.g., ggplot, geom\_point etc) were combined with + symbol rather than % pipe.

# 4. adding aesthetics and layers

• we can add gender as the third variable, by adding the colour

```
asthmads %>%
ggplot(data = . ,
mapping = aes(x = Ht_m, y = Weight_Pre,
colour = Gender)) +
geom_point()
```



# 5. Adding labels

```
asthmads %>%

ggplot(data = .,

mapping = aes(x = Ht_m, y = Weight_Pre,

colour = Gender)) +

geom_point() +

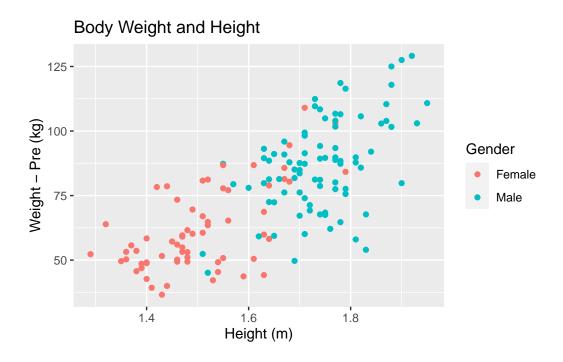
labs(title = "Body Weight and Height",

x = "Height (m)",

y = "Weight - Pre (kg)",

colour = "Gender")
```

11



# 7. Change the scale

```
asthmads %>%

ggplot(data = . ,

mapping = aes(x = Ht_m, y = Weight_Pre,

colour = Gender)) +

geom_point() +

scale_x_continuous(breaks = seq(1, 2.2, 0.1)) +

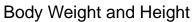
scale_y_continuous(breaks = seq(0, 200, 20)) +

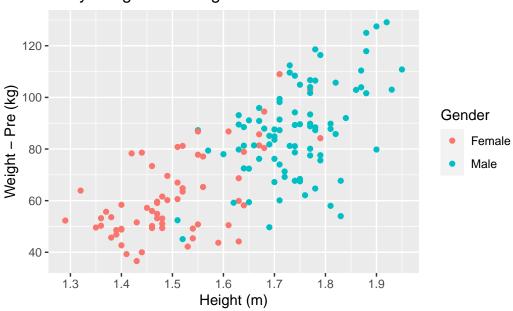
labs(title = "Body Weight and Height",

x = "Height (m)",

y = "Weight - Pre (kg)",

colour = "Gender")
```





# 8. Adding theme to the plot

```
asthmads %>%

ggplot(data = .,

mapping = aes(x = Ht_m, y = Weight_Pre,

colour = Gender)) +

geom_point() +

scale_x_continuous(breaks = seq(1, 2.2, 0.1)) +

scale_y_continuous(breaks = seq(0, 200, 20)) +

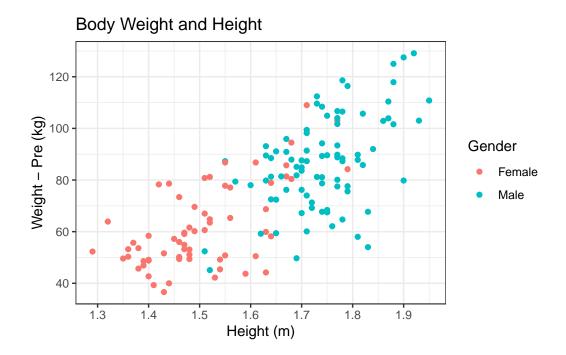
labs(title = "Body Weight and Height",

x = "Height (m)",

y = "Weight - Pre (kg)",

colour = "Gender") +

theme_bw()
```



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.

```
Note

dataset used in this example can be downloaded from github site:

https://github.com/MohdAzmiSuliman/IKU_RBook/tree/master/dataset
```

# 7.1 Practical: 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

```
title: "Sesi 3 - Basic Statistical Analysis"
format:
html:
embed-resources: true
```

# 7.2 Bivariate Analyses of Categorical Data

Bivariate analysis involves examining the relationship or association between to 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 know as cross-tabulation or crosstab.

### 7.2.1 Contingency Table

Contigency table, or crosstab, is a simple matrix that display the frequency of occurence of combination of two categories for two categorical variable.

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

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

Table 7.1: Contigency Table Structure

### 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.

# Example: Gender vs Employment Status

### 0. Download dataset

- we will be using the asthmads clean.sav dataset
- this code below download directly from github package into your working directory
- otherwise, you may download from the link: https://github.com/MohdAzmiSuliman/IKU\_RBook/ raw/master/dataset/asthmads\_clean.sav and copy to the working directory

```
```{r}
# | eval: false
download.file(
 url = "https://github.com/MohdAzmiSuliman/IKU_RBook/raw/master/dataset/asthmads_clean.sav",
  destfile = "asthmads_clean.sav", mode = "wb")
```

1. Import dataset

```
```{r}
library(tidyverse)
library(haven)
asthmads <- read_sav("asthmads_clean.sav") %>%
 as_factor()
asthmads
```

### # A tibble: 150 x 22

```
id idR
 Gender
 Age WorkStatus Height Ht m Weight Pre BMI Pre
 <dbl> <chr> <fct> <dbl> <fct>
 <dbl> <dbl>
 <dbl>
 <dbl>
 34 Unemployed
 1
 1 nXSw Female
 179 1.79
 84.2
 26.3
2
 2 yg2t
 Male
 31 Unemployed
 169 1.69
 81.8
 28.6
 25 Employed
 32.9
3
 3 QBW4
 Male
 164 1.64
 88.5
 4
 4 2×2S Female
 33 Unemployed
 136 1.36
 53.2
 28.8
 5
 5 mOnn Male
 28 Unemployed
 24.1
 172 1.72
 71.3
 6
 6 D3sl Male
 33 Unemployed
 178 1.78
 87.3
 27.6
 31 Unemployed
 24.9
 7
 7 le6j Female
 140 1.4
 48.8
8
 8 r3gC Female
 34 Employed
 140 1.4
 49.1
 25.0
 9
 20.6
 9 3Tyt Male
 31 Employed
 171 1.71
 60.1
10
 10 cmKF Male
 28 Employed
 163 1.63
 93.1
 35.0
i 140 more rows
```

```
i 13 more variables: BMI_PreCat <fct>, PA_HW <dbl>, Weight_Post <dbl>,
```

- BMI\_Post <dbl>, BMI\_PostCat <fct>, Tx1 <fct>, Tx2 <fct>, PEFR\_Pre <dbl>,
- PEFR\_Post <dbl>, SxWheeze\_Pre <fct>, SxWheeze\_Post <fct>, PS\_Pre <dbl>, #
- PS\_Post <dbl> #

### 2. Create Contingency Table

```
1 ```{r}
2 with(asthmads, table(Gender, WorkStatus))
3 ```
```

### WorkStatus

```
Gender Unemployed Employed
Female 47 17
Male 33 53
```

3. **BONUS 1**: We can also use tidyverse::, janitor:: and gt:: package to create nice looking table with total row and total column

```
1 ```{r}
2 #| tbl-cap: "2×2 Contigency Table for Gender by Employment Status, using tidyverse, janitor and gt"
3 asthmads %>%
5 count(Gender, WorkStatus) %>%
6 pivot_wider(names_from = WorkStatus, values_from = n,
7 values_fill = list(n = 0)) %>%
8 janitor::adorn_totals(c("row", "col")) %>%
9 gt()
```

Table 7.2: 2x2 Contigency Table for Gender by Employment Status, using tidyverse, janitor and gt

Gender	Unemployed	Employed	Total
Female	47	17	64
Male	33	53	86
Total	80	70	150

3. **BONUS 2**: We can also use gtsummary :: package to create nice looking table with total row and total column

Table 7.3: 2x2 Contigency Table for Gender by Employment Status, using gtsummary

Characteristic	Unemployed, N = 80	Employed, N = 70
Gender		
Female	47 (58.8%)	17 (24.3%)
Male	33 (41.3%)	53 (75.7%)

# 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 variable.

Pearson Chi-square Test is conducted with these assumptions:

- both variables must be **categorical** (either nominal or ordinal)
- the observation must be **independence**. This mean that the outcome of one observation is not influency by the outcome of another observation
- the groupings are mutually exclusive
- <20% of celss have expected frequency of <5</li>
- minimum expected frequency if >1

Example: Gender vs Employment Status

- 1. Refer previous part to create contingency table
- 2. Calculate the chi-square test

```
1 ```{r}
2 with(asthmads, table(Gender, WorkStatus)) %>%
3 chisq.test(., correct = F)
4 ```
```

Pearson's Chi-squared test

```
data: .
X-squared = 18.128, df = 1, p-value = 2.066e-05
```

3. Check the test's assumptions

```
1 ```{r}
2 with(asthmads, table(Gender, WorkStatus)) %>%
3 chisq.test(., correct = F) %>%
4 .$expected
5
```

```
WorkStatus
Gender Unemployed Employed
Female 34.13333 29.86667
Male 45.86667 40.13333
```

```
Tip
```

If the assumptions for chi-square is violated, R will show warning message In chisq.test(.): Chi-squared approximation may be incorrect which is most likely due to violation of small sample assumptions.

4. **Bonus**: we can create a nice looking table with gtsummary

```
"" \{r\}
"" | tbl-cap: "Association between Gender and Employment Status"

library(gtsummary)

asthmads %>%
tbl_summary(include = Gender,
by = WorkStatus,
digits = all_categorical() ~ c(0,1)) %>%
add_p(test = all_categorical() ~ "chisq.test",
test.args = all_tests("chisq.test") ~ list(correct = F))
```

Table 7.4: Association between Gender and Employment Status

Characteristic	<b>Unemployed</b> , N = 80	Employed, N = 70	p-value	
Gender			<0.001	
Female	47 (58.8%)	17 (24.3%)		
Male	33 (41.3%)	53 (75.7%)		

# 7.2.3 Small Sample

Pearson's Chi-square is only reliable with medium to large datasets. There are two (2) assumptions in the test that might be violated for a small sample, which were (1) <20% of cells have an expected frequency of >5 and (2) minimum expected frequency >1.

There are two alternatives for a small sample, namely:

- · Yates' Correction for Continuity
- · Fisher Exact Test

### 7.2.3.1 Yates' Correction for Continuity

Yates' correction adjust the Chi-square statistics to account for the overestimation of significance due to the continuity assumption of the chi-square distribution.

In R, Yates' correction is the default option for Pearson Chi-square

# Note

While Yates' correction is used for small sample (especially when it violate the chi-square assumptions), Yates' correction is not preferable for moderate or large sample, as it can be overly conservative, increasing the risk of Type II error

```
1 ```{r}
2 with(asthmads, table(Gender, WorkStatus)) %>%
3 chisq.test(.)
4 ```
```

Pearson's Chi-squared test with Yates' continuity correction

```
data: .
X-squared = 16.746, df = 1, p-value = 4.273e-05
```

```
1 ```{r}
2 asthmads %>%
3 tbl_summary(include = Gender,
4 by = WorkStatus,
5 digits = all_categorical() ~ c(0,1)) %>%
6 add_p()
7
```

Characteristic	Unemployed, N = 80	Employed, N = 70	p-value	
Gender			<0.001	
Female	47 (58.8%)	17 (24.3%)		
Male	33 (41.3%)	53 (75.7%)		

### 7.2.3.2 Fisher's Exact Test

Pearson's Chi-square test (and the Yates' Correction) is an approximation test, while Fisher Exact Test was based on calculating the exact probability of observing the data under the null hypothesis. Thus it is preferable compared to Pearson's Chi-square Test especially in small sample.

```
1 ```{r}
2 with(asthmads, table(Gender, WorkStatus)) %>%
3 fisher.test(.)
4
```

Fisher's Exact Test for Count Data

```
data: .
p-value = 2.96e-05
alternative hypothesis: true odds ratio is not equal to 1
95 percent confidence interval:
 2.08176 9.60658
sample estimates:
odds ratio
 4.392789
```

```
asthmads %>%
tbl_summary(include = Gender,
by = WorkStatus,
digits = all_categorical() ~ c(0,1)) %>%
add_p(test = all_categorical() ~ "fisher.test")
```

Characteristic	Unemployed, N = 80	Employed, N = 70	p-value	
Gender			<0.001	
Female	47 (58.8%)	17 (24.3%)		
Male	33 (41.3%)	53 (75.7%)		

### 7.2.4 McNemar Test

McNemar test is Pearson's Chi-square Test equivalence for paired data (e.g., pre-post data).

Example: Wheezing Symptom Pre and Post Intervention

1. Create Contingency Table

```
1 ```{r}
2 with(asthmads, table(SxWheeze_Pre, SxWheeze_Post))
3
```

```
SxWheeze_Post
SxWheeze_Pre No Yes
No 43 19
Yes 56 32
```

### 2. Calcualte the McNemar Test statistic

```
```{r}
with(asthmads, table(SxWheeze_Pre, SxWheeze_Post)) %>%
  mcnemar.test(correct = F)
```

```
McNemar's Chi-squared test
data:
McNemar's chi-squared = 18.253, df = 1, p-value = 1.934e-05
```



Caution

Supposedly gtsummary:: package can create table for paired data, unfortunately I unable to replicate the code, despite converted to long format.

7.3 Bivariate Analyses of Numerical Data

Bivariate analysis involves examining the relationship or association between two variables, and bivariate analyses of numerical data is said when involving continuous dependent variables. Example of bivariate analyses of numerical data include

- · Parametric Tests
 - Independent t-test
 - Paired t-test
 - Analysis of Variance (ANOVA)
 - Pearson' Correlation
- · Non-Parametric Tests
 - Mann-Whitney U Test
 - Wilcoxon Signed Rank Test
 - Spearman's Correlation

Many of parametric tests were based on normal distribution and analyses of the variances, while non-parametric tests were based on rank.

7.3.1 Independent T-test

Independent t-test is a parametric test, commonly used to compare the mean of two independent sample, more specifically, compare the difference of the two means, in relation to the variation (i.e., the variance) of the data.

Independent t-test is conducted with these assumptions:

- samples was taken **randomly** (i.e., the sample representative of the population)
- the groups and measurements were **independents**.
- the outcome (or dependent variable) is **numerical** data.
- the outcome is **normally distributed** in each group.
- the variance outcome between groups is approximately equal (homogeneity of variance)

Note

- 1. In large sample size, T-test can be robust to violation of normal distribution assumption, based on Central Limit Theorem.
- 2. The homogeneity of variance can be ignore if we use Welch T-test as default.

Example: Compare Height between Gender

- 1. We will use the same dataset as previous asthmads.
 - · below is the code as recap

```
# | eval: false

download.file(
    url = "https://github.com/MohdAzmiSuliman/IKU_RBook/raw/master/dataset/asthmads_clean.sav",
    destfile = "asthmads_clean.sav", mode = "wb")

asthmads <- read_sav("asthmads_clean.sav") %>%
    as_factor()

"``{r}
    asthmads
"``
```

```
# A tibble: 150 x 22
```

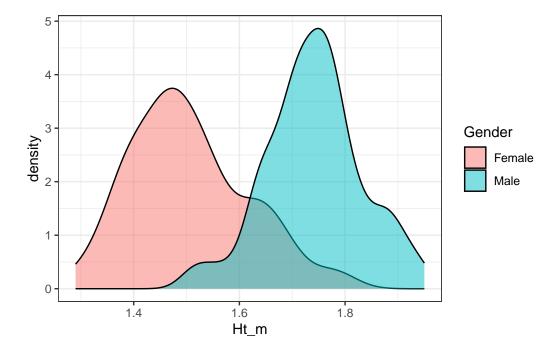
```
id idR
              Gender
                       Age WorkStatus Height Ht_m Weight_Pre BMI_Pre
   <dbl> <chr> <fct> <dbl> <fct>
                                       <dbl> <dbl>
                                                        <dbl>
                                                                <dbl>
                                                         84.2
                                                                26.3
1
      1 nXSw Female
                        34 Unemployed
                                         179 1.79
                                                                28.6
2
      2 yg2t Male
                        31 Unemployed
                                         169 1.69
                                                         81.8
3
      3 QBW4 Male
                        25 Employed
                                         164 1.64
                                                         88.5
                                                                32.9
      4 2×2S Female
                        33 Unemployed
                                         136 1.36
                                                                28.8
                                                         53.2
5
      5 mOnn Male
                        28 Unemployed
                                         172 1.72
                                                         71.3
                                                                 24.1
6
      6 D3sl Male
                        33 Unemployed
                                         178 1.78
                                                        87.3
                                                                27.6
7
      7 le6j Female
                        31 Unemployed
                                         140 1.4
                                                         48.8
                                                                 24.9
                                                                25.0
8
      8 r3gC Female
                        34 Employed
                                         140 1.4
                                                         49.1
9
      9 3Tyt Male
                        31 Employed
                                         171 1.71
                                                                 20.6
                                                         60.1
     10 cmKF Male
                        28 Employed
                                         163 1.63
                                                         93.1
                                                                 35.0
10
# i 140 more rows
```

```
# i 13 more variables: BMI_PreCat <fct>, PA_HW <dbl>, Weight_Post <dbl>,
# BMI_Post <dbl>, BMI_PostCat <fct>, Tx1 <fct>, Tx2 <fct>, PEFR_Pre <dbl>,
# PEFR_Post <dbl>, SxWheeze_Pre <fct>, SxWheeze_Post <fct>, PS_Pre <dbl>,
# PS_Post <dbl>
```

2. Confirm data distribution

• assumption for normal distributed in each group

```
1 ```{r}
2 asthmads %>%
3 ggplot(aes(x = Ht_m, fill = Gender)) +
4 geom_density(alpha = .5) +
5 theme_bw()
6
```



3. Calculate mean and SD of height for each gender

```
# A tibble: 2 x 3
  Gender mean sd
  <fct> <dbl> <dbl> <dbl>
1 Female 1.50 0.110
2 Male 1.74 0.0898
```

4. Conduct the Welch's T-test

```
1 ```{r}
2 t.test(Ht_m ~ Gender, asthmads)
3
```

Important

In R, many functions require a formula parameter, especially in statistical modelling.

- The general form of a formula is outcome ~ predictors, data,
 - outcome = dependent variable
 - predictors = independent variables.
- This formula structure is used in various functions, such as linear modelling.
- For a t-test, which compares means across groups,
 - the formula formed by outcome ~ group, data
 - group = categorical variable = groups

5. Bonus 1: We can create a nice looking table with gtsummary

Table 7.7: Height Differences between Gender table, using gtsummary

Characteristic	Female , N = 64	Male , N = 86	Difference	95% CI	p-value
Height (m)	1.50 (0.11)	1.74 (0.09)	-0.24	-0.27, -0.20	<0.001

6. Bonus 2: the classical Student T-test

As mention previously, by default, R will use Welch's T-test, regardless of the assumption of homogeneity of variance. But if you still want to use the Student T-test, change the var.equal = TRUE parameter

```
1 ```{r}
2 t.test(Ht_m ~ Gender, asthmads, var.equal = T)
3 ```
```

```
Two Sample t-test
```

```
data: Ht_m by Gender
t = -14.539, df = 148, p-value < 2.2e-16
```

alternative hypothesis: true difference in means between group Female and group Male is not equal to 0 95 percent confidence interval:

```
-0.2693719 -0.2049086
sample estimates:
mean in group Female mean in
```

mean in group Female mean in group Male
1.503906 1.741047

Important

Interested to know why Welch's T-test is preferable? In short:

- Student T-test result biased when assumption for normality and homogeneity of variance were not met
- Welch's T-test provides better control of Type 1 error rates when assumption of homoegeneity of variance was not met.
- Using Welch's T-test as default skip the need to test for homogeneity of variance (i.e., the Levene's test)
- Real data commonly not normally distributed and it is reasonable to assume that variance is unequal in many studies

Source: https://doi.org/10.5334/irsp.82

7.3.2 Paired T-test

Paired T-test is a parametric test, used to compare the mean of two dependent sample, e.g., comparing the mean for pre and post measurement.

Independent t-test is conducted with these assumptions:

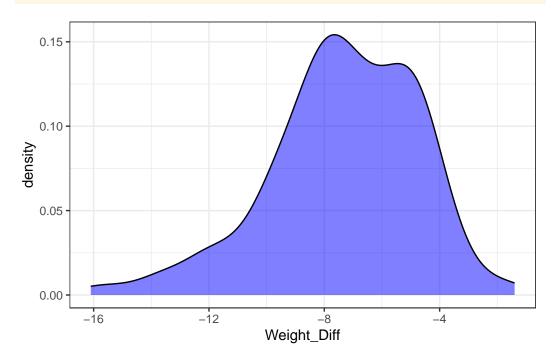
- The samples were **random**
- The measurement were **dependent** (i.e., paired, e.g., pre-post)
- The outcome variable is numerical data
- The outcome differences is normally distributed

Example: Weight differences from pre to post measurement

- 1. We will use the same dataset as previous: asthmads
- 2. Check the differences data distribution
 - assumption for outcome difference is normally distributed
 - we need to calculate the differences

• then plot the differences to check the distribution

```
1 ```{r}
2 asthmads %>%
3 mutate(Weight_Diff = Weight_Post - Weight_Pre) %>%
4 ggplot(aes(x = Weight_Diff)) +
5 geom_density(alpha = .5, fill = "blue") +
6 theme_bw()
```



- we can see that the data is approximately normally distributed
- 3. Calculate mean and SD of weight pre & post

```
···{r}
asthmads %>%
  summarise(Weight_Pre_Mean = mean(Weight_Pre, na.rm = T),
            Weight_Pre_SD = sd(Weight_Pre, na.rm = T),
            Weight_Post_Mean = mean(Weight_Post, na.rm = T),
           Weight_Post_SD = sd(Weight_Post, na.rm = T))
```

A tibble: 1 x 4

```
Weight_Pre_Mean Weight_Pre_SD Weight_Post_Mean Weight_Post_SD
           <dbl>
                         <dbl>
                                         <dbl>
                                                         <dbl>
1
            75.9
                          21.3
                                          68.6
                                                          20.5
```

4. Conduct the Paired T-test

```
```{r}
t.test(Pair(Weight_Post, Weight_Pre) ~ 1, asthmads)
```

### Paired t-test

```
data: Pair(Weight_Post, Weight_Pre)
t = -34.658, df = 149, p-value < 2.2e-16
alternative hypothesis: true mean difference is not equal to 0
95 percent confidence interval:
-7.755671 -6.918996
sample estimates:
mean difference
 -7.337333
```

# Caution

Since this is pre-post test, we need to write the post column first. In this example, the samples had reduction in body weight, thus the differences is in negative

5. **Bonus**: we can create a nice looking table with gtsummary

```
```{r}
asthmads %>%
  select(id, Weight_Post, Weight_Pre) %>%
  pivot_longer(cols = starts_with("Weight"),
               names_to = "Type",
```

Characteristic	Pre , N = 150	Post , N = 150	Difference	95% CI	p-value
Weight	75.89 (21.27)	68.55 (20.46)	-7.3	-7.8, -6.9	<0.001

7.3.3 ANOVA - Analysis of Variance

Analysis of Variance (ANOVA) is a parametric test, commonly used to compare the means of three or more independent (unrelated) groups, more specifically, in relation to the variation (i.e. the variance) of the data.

Note

There are severals test in ANOVA, which include one-way ANOVA, two-way ANOVA, N-way ANOVA, ANCOVA, Repeated Measure ANOVA, RM ANCOVA, MANOVA etc.

In this example, we will focus only to one-way ANOVA

One-way ANOVA is conducted with these assumptions:

- The samples were taken randomly
- The groups and measurements were independents
- The outcome is **numerical** data
- The outcome in approximately **normal distribution** in each group
- The variances of outcome in each group have approximately equal, i.e. homogeneity of variance.



note that almost all parametric tests have similar assumptions.

Example: PEFR improvement between Treatment Group

- 1. We will use the same dataset as previous, i.e., the asthmads
 - we will need to wrangle the dataset, to calculate the PEFR improvement

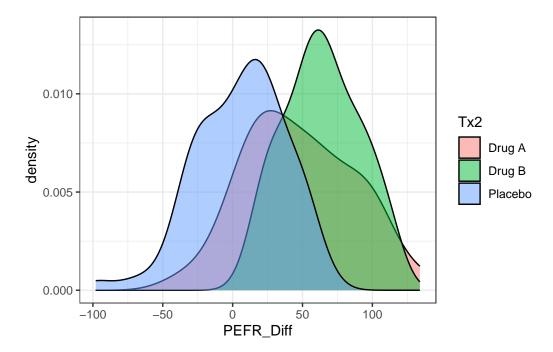
```
1 ```{r}
2 asthmads <- asthmads %>%
3 mutate(PEFR_Diff = PEFR_Post-PEFR_Pre)
```

4

2. Confirm data distribution

• assumption for normal distributed in each group

```
1 ```{r}
2 asthmads %>%
3 ggplot(aes(x = PEFR_Diff, fill = Tx2)) +
4 geom_density(alpha = .5) +
5 theme_bw()
6
```



3. Calculate mean and SD of PEFR_Diff for each treatment group

```
# A tibble: 3 x 3
  Tx2   mean   sd
  <fct>  <dbl>  <dbl>  38.7
2 Drug A 47.9   38.7
2 Drug B 64   26.9
3 Placebo 5.92  31.9
```

7 Statistical Tests

4. Conduct the ANOVA

```
1 ```{r}
2 oneway.test(PEFR_Diff ~ Tx2, asthmads)
3 ```
```

One-way analysis of means (not assuming equal variances)

```
data: PEFR_Diff and Tx2 F = 56.384, num df = 2.000, denom df = 78.046, p-value = 7.064e-16
```

Note

In R, there are three related functions, but used differently, for ANOVA.

- Function oneway.test(_) is for Welch's ANOVA
- 2. Function aov(_) is for ANOVA assuming equal variance
- 3. Function anova(_) is to compare between models.
- 5. Pair-wise post-hoc test
 - since the comparison by ANOVA is significant, we need to do post-hoc test to check which pair have significant difference.
 - since there are three pairwise comparison (Drug A vs B, Drug A vs Placebo and Drug B vs Placebo), in Bonferroni, the alpha level need to divide by 3 (i.e., 0.017)

Important

Type 1 error increase with the number of comparison being made. Thus adjustment to the alpha level are necessary.

Common post-hoc adjustment include Bonferroni Correction, in which the original alpha level (\mathbf{Z}) divided by the number of comparison.

```
pairwise.t.test(asthmads$PEFR_Diff, asthmads$Tx2,
p.adjust.method = "b")
```

Pairwise comparisons using t tests with pooled SD

data: asthmads\$PEFR_Diff and asthmads\$Tx2

```
Drug A Drug B
Drug B 0.09 -
Placebo 9.8e-09 4.8e-16
```

P value adjustment method: bonferroni

we can see that only Drug A vs B is not significant. other pairs were significant.

6. **Bonus**: we can create nice looking table with gtsummary

```
```{r}
asthmads %>%
 tbl_summary(include = PEFR_Diff,
 by = Tx2,
 label = PEFR_Diff ~ "PEFR Difference (Post - Pre)",
 statistic = all_continuous() ~ "{mean} ({sd})",
 digits = all_continuous() ~ 2) %>%
 add_p(test = all_continuous() ~ "oneway.test")
```

Characteristic	<b>Drug A</b> , N = 35	<b>Drug B</b> , N = 43	Placebo, N = 72	p-value
PEFR Difference (Post - Pre)	47.89 (38.71)	64.00 (26.86)	5.92 (31.94)	<0.001

#### 7.3.4 Pearson's Correlation

Pearson's correlation is a parametric test, commonly used to test correlation between two numerical data.

The Pearson's Correlation assumptions were:

- each observation/measurement were **independent**
- both variables were numerical data
- both variables should follow approximately **normal distribution**.
- the relationship between the two variables should be **linear**.
- · the scatterplot of the two variables should show a roughly constant spread, the homoscedasticity



homogeneity and homoscedasticity is similar concept but mean differently. homogeneity means the variance between different groups were approximately equal, but homoscedasticity means the variances of residuals or best fit line were approximately constant.

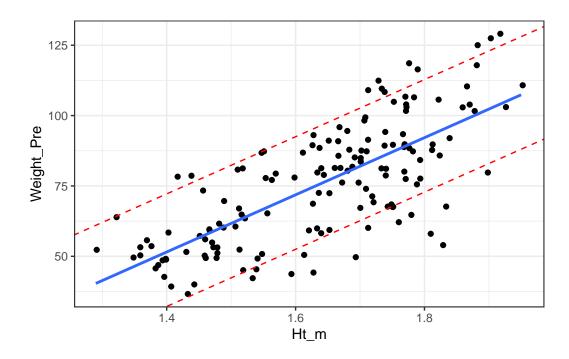
Example: Correlation between height and body weight

- 1. We will use the same dataset as previous, i.e. the asthmads.
- 2. Confirm data distribution
- · assumptions for normal distribution

```
```{r}
  # | layout-ncol: 2
  asthmads %>%
     ggplot(aes(x = Ht_m)) +
     geom_density(fill = "lightblue", alpha = .5) +
     theme_bw()
  asthmads %>%
     ggplot(aes(x = Weight_Pre)) +
     geom_density(fill = "lightgreen", alpha = .5) +
     theme_bw()
                                                  0.015
                                                 density density
density
                                                  0.005
                                                  0.000
                       1.6
Ht_m
                                                                        75
Weight_Pre
```

· assumption for linearity & homoscedasciticity

```
···{r}
   modslope <- lm(Weight_Pre ~ Ht_m, data = asthmads) %>%
     broom::tidy() %>%
     filter(term = "Ht_m") %>%
     pull(estimate)
   asthmads %>%
     ggplot(aes(x = Ht_m, y = Weight_Pre)) +
     geom_point(position = position_jitter()) +
     geom_smooth(method = "lm", se = F) +
10
     geom_abline(slope = modslope, intercept = -70,
11
                 colour = "red", linetype = 2) +
     geom_abline(slope = modslope, intercept = -110,
13
                 colour = "red", linetype = 2) +
14
     theme_bw()
15
```



4. Calculate the pearson correlation

```
cor.test(~ Ht_m + Weight_Pre, asthmads, method = "pearson")
```

Pearson's product-moment correlation

```
data: Ht_m and Weight_Pre
t = 13.09, df = 148, p-value < 2.2e-16
alternative hypothesis: true correlation is not equal to 0
95 percent confidence interval:
    0.6483493   0.7989693
sample estimates:
        cor
0.7325001</pre>
```

5. **Bonus 1**: Create nice table with apaTables :: package

```
1 ```{r}
2 library(apaTables)
3
4 asthmads %>%
5 select(Ht_m, Weight_Pre) %>%
6 apa.cor.table(.)
7
```

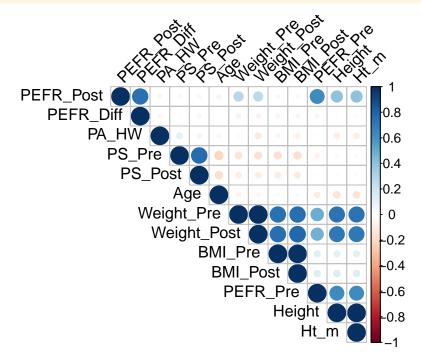
7 Statistical Tests

Means, standard deviations, and correlations with confidence intervals

Note. M and SD are used to represent mean and standard deviation, respectively. Values in square brackets indicate the 95% confidence interval. The confidence interval is a plausible range of population correlations that could have caused the sample correlation (Cumming, 2014).

* indicates p < .05. ** indicates p < .01.

6. Bonus 2: Correlation Plot



7.4 Linear Regression

Linear regression is a statistical method used to model the relationship between a dependent variable and one or more independent variables by fitting a linear equation to observed data.

The formula for linear regression is

$$y = \beta_0 + \beta_1 x + \epsilon$$

where

- *y* is the dependent variable (outcome)
- *x* is the independent variable (predictor)
- β_0 is the y-intercept
- β_1 is the slope of the line, representing the change in y for one-unit change in x
- ϵ is the error term

For linear regression to provide reliable estimates, several key assumptions must be met

- each observation/measurement were independent
- the outcome variable is numerical data
- the residuals follow approximately **normal distribution**.
- the relationship between the predictor and the outcome should be **linear**.
- the residuals have constants variance at every level of the predictor, i.e., the **homoscedasticity**
- in multiple linear regression, there will be no or little multicollinearity, which independent variables are not too highly correlated with each other



note the similarity between pearson correlation and linear regression

Example: Finding factors associated with BMI changes

7.4.0.1 Simple Linear Regression

- 1. We will use the same dataset as previous, i.e. the asthmads
 - · we need to do some data wrangling to calculate the BMI changes

```
```{r}
asthmads <- asthmads %>%
 mutate(BMI_Diff = BMI_Post-BMI_Pre)
```

- 2. Conduct the simple linear regression
  - in this example, the simple linear regression model is saved in R object, for used later

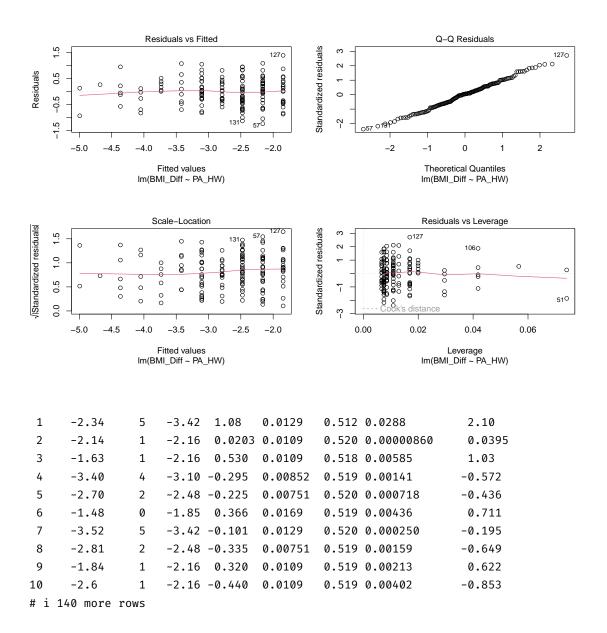
```
```{r}
 slinm <- lm(BMI_Diff ~ PA_HW, asthmads)</pre>
summary(slinm)
Call:
lm(formula = BMI_Diff ~ PA_HW, data = asthmads)
Residuals:
           1Q Median 3Q
    Min
                                     Max
-1.22966 -0.35884 0.02248 0.35711 1.39559
Coefficients:
          Estimate Std. Error t value Pr(>|t|)
(Intercept) -1.84559 0.06743 -27.37
                                    <2e-16 ***
PA_HW
         Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
Residual standard error: 0.5182 on 148 degrees of freedom
Multiple R-squared: 0.6567, Adjusted R-squared: 0.6544
F-statistic: 283.1 on 1 and 148 DF, p-value: < 2.2e-16
  3. Test assumptions
```

the simplest way for assumptions testing, is to use plot(_) function

```
1 ```{r}
2 #| layout-ncol: 2
3
4 plot(slinm)
5
```

 otherwise, we call also use augment(_) function from broom:: package to extract the residuals and check the assumptions

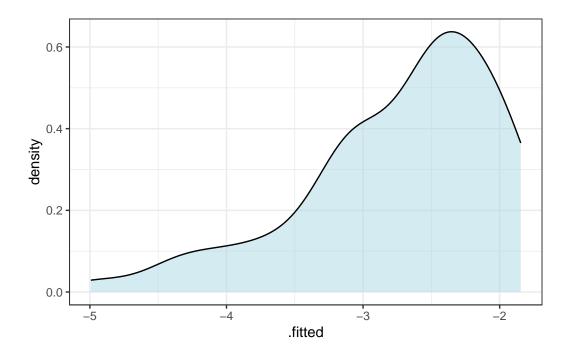
```
1 ```{r}
2 library(broom)
3
4 augment(slinm)
5
```



· using the dataset from augment, to plot the residual to check for normality

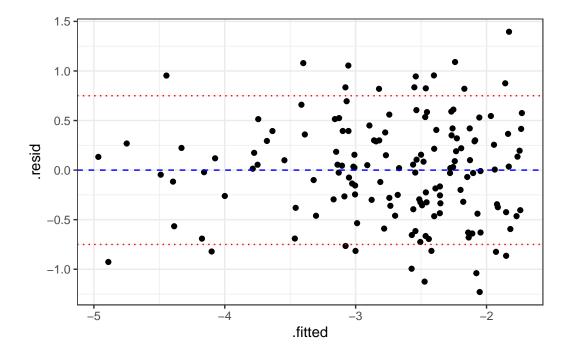
```
1 ```{r}
2 augment(slinm) %>%
3 ggplot(aes(.fitted)) +
4 geom_density(fill = "lightblue", alpha = .5) +
5 theme_bw()
6
```

7 Statistical Tests



- plot residual values vs predicted values (aka fitted values) for linearity and homoscedasticity
 - linearity: abscence of systematic patterns like curves
 - homoscedasticity: residuals randomly scttered around the horizontal line with no funnel-shapped patern

```
augment(slinm) %>%
ggplot(aes(x = .fitted, y = .resid)) +
geom_point(position = position_jitter()) +
geom_hline(yintercept = 0, linetype = 2, colour = "blue") +
geom_hline(yintercept = .75, linetype = 3, colour = "red") +
geom_hline(yintercept = -.75, linetype = 3, colour = "red") +
theme_bw()
```



4. Check model fitness

- one of parameter to check our model fitness is by looking at the coefficient of determination $\it r^2$ from the model summary

```
1 ```{r}
2 summary(slinm)
3 ```
```

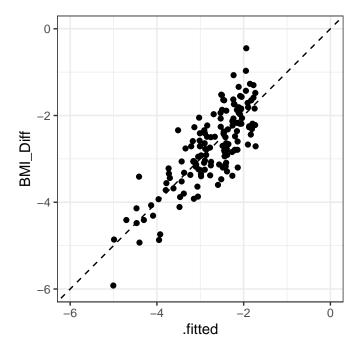
```
Call:
lm(formula = BMI_Diff ~ PA_HW, data = asthmads)
Residuals:
     Min
               1Q
                   Median
                                 3Q
                                         Max
-1.22966 -0.35884 0.02248 0.35711 1.39559
Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) -1.84559
                        0.06743 -27.37
                                          <2e-16 ***
PA_HW
            -0.31476
                        0.01871 -16.82
                                          <2e-16 ***
Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
Residual standard error: 0.5182 on 148 degrees of freedom
Multiple R-squared: 0.6567,
                              Adjusted R-squared: 0.6544
F-statistic: 283.1 on 1 and 148 DF, p-value: < 2.2e-16
```

• we can also plot our observed vs predicted value to visualize the model fitness

7 Statistical Tests

- in this plot, the point should be close at the diagonal, i.e. observed = predicted

```
1    ```{r}
2    slinm %>%
3    augment() %>%
4    ggplot(aes(x = .fitted, y = BMI_Diff)) +
5    scale_x_continuous(limits = c(-6, 0)) +
6    scale_y_continuous(limits = c(-6, 0)) +
7    geom_point(position = position_jitter()) +
8    geom_abline(intercept = 0, slope = 1, linetype = 2) +
9    theme_bw() +
10    theme(aspect.ratio = 1)
```



- 5. **Bonus**: nice table with gtsummary
- tbl_uvregression(_) functions from gtsummary not only can create a nice table

Characteristic	N	Beta	95% CI	p-value
Physical Activity (total hour per week)	150	-0.31	-0.35, -0.28	<0.001

· but also can simplified our simple linear regression involving several predictors

Characteristic	N	Beta	95% CI	p-value
Gender	150			
Female		_	_	
Male		0.23	-0.05, 0.52	0.108
Age (year)	150	0.01	-0.04, 0.06	0.609
Employment	150			
Unemployed		_	_	
Employed		0.02	-0.26, 0.31	0.884
Height (cm)	150	0.01	0.00, 0.02	0.070
Physical Activity (total hour per week)	150	-0.31	-0.35, -0.28	<0.001

7.4.1 Multiple Linear Regression

Real-world relationships between variables rarely occur in isolation; they are often influenced by third variables. These third variables can be categorized as:

- Confounders
- Mediators
- Moderators

Controlling for these third variables is crucial for accurately understanding the relationships among variables. While the optimal approach is to control for these variables at the study design level (e.g., through randomization, stratification, and setting proper inclusion and exclusion criteria), the presence of third variables can be inevitable and may significantly impact the observed relationships.

When third variables cannot be controlled through design alone, statistical methods can be employed to adjust for their influence, including (not limited to) Stratification, Multivariable Analysis or Multivariate Analysis

Example: Find factors associated with BMI changes

- 1. Conduct multiple linear regression
 - and check for model fitness (R^2)

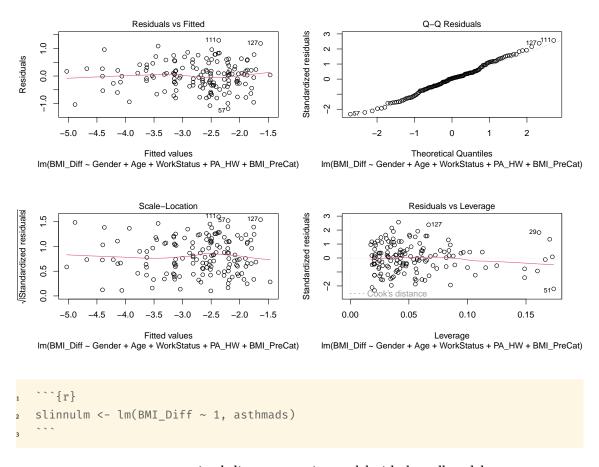
```
```{r}
 mlinm <- lm(BMI_Diff ~ Gender + Age + WorkStatus + PA_HW + BMI_PreCat,
 asthmads)
 summary(mlinm)
Call:
lm(formula = BMI_Diff ~ Gender + Age + WorkStatus + PA_HW + BMI_PreCat,
 data = asthmads)
Residuals:
 Min
 1Q
 Median
 3Q
 Max
-1.18613 -0.32674 0.02218 0.28579 1.29659
Coefficients:
 Estimate Std. Error t value Pr(>|t|)
(Intercept)
 -1.800884 0.479161 -3.758 0.000249 ***
GenderMale
 0.157402 0.091767 1.715 0.088483 .
 0.005324 0.014421 0.369 0.712517
Age
WorkStatusEmployed -0.159447 0.093687 -1.702 0.090962 .
PA_HW
 BMI_PreCatNormal
 -0.140436 0.223807 -0.627 0.531347
BMI_PreCatObese -0.248582 0.205849 -1.208 0.229211
Signif. codes: 0 '***' 0.001 '**' 0.05 '.' 0.1 ' ' 1
Residual standard error: 0.5156 on 142 degrees of freedom
Multiple R-squared: 0.6739, Adjusted R-squared: 0.6578
F-statistic: 41.93 on 7 and 142 DF, p-value: < 2.2e-16
 2. Test for assumptions
```

```
···{r}
| layout-ncol: 2
plot(mlinm)
```

# 3. Bonus 1: Model Comparison

we can compare multiple model with anova(\_) function

• for example we have null model



- we want to compare our simple linear regression model with the null model  $% \left( 1\right) =\left( 1\right) \left( 1\right$ 

```
1 ```{r}
2 anova(slinnulm, slinm)
3 ```
```

Analysis of Variance Table

```
Model 1: BMI_Diff ~ 1

Model 2: BMI_Diff ~ PA_HW

Res.Df RSS Df Sum of Sq F Pr(>F)

1 149 115.774

2 148 39.748 1 76.026 283.08 < 2.2e-16 ***

Signif. codes: 0 '***' 0.001 '**' 0.05 '.' 0.1 ' ' 1
```

• we can also compare multiple models

```
1 ```{r}
2 anova(slinnulm, slinm, mlinm)
3 ```
```

Analysis of Variance Table

- we can see that our multiple linear regression model is not significantly different with simple linear regression model. thus we can choose the simpler model i.e., the simple linear regression model.
- 4. Bonus 2: Variable Selection

# Important

variable selection should start with plausible relationship i.e., back up with prior knowledge. this should start even at study design, in which unlikely relationship should not be included as variable collected in the study. this is to avoid spurious significant association further read: https://doi.org/10.1002%2Fbimj.201700067

While variable selection should be made purposely, i.e., the PI specifically select variable to be included in final model, variable selection by statistically software is still available to conduct.

· create null model

```
1 ```{r}
2 mlinnulm <- lm(BMI_Diff ~ 1, asthmads)
3 ```</pre>
```

· create full model

```
1 ```{r}
2 mlinfulm <- lm(BMI_Diff ~ Gender + Age + WorkStatus + PA_HW + BMI_PreCat,
3 asthmads)
4 ```</pre>
```

conduct forward selection

```
step(mlinfulm,
scope = list(lower = mlinnulm, upper = mlinfulm),
direction = "forward",
trace = 0)
```

```
Call:
lm(formula = BMI_Diff ~ Gender + Age + WorkStatus + PA_HW + BMI_PreCat,
 data = asthmads)
Coefficients:
 GenderMale
 (Intercept)
 Age
 -1.800884
 0.157402
 0.005324
 WorkStatusEmployed
 PA_HW
 BMI_PreCatNormal
 -0.140436
 -0.159447
 -0.312087
BMI_PreCatOverweight
 BMI_PreCatObese
 -0.336183
 -0.248582

 conduct backward selection

 ···{r}
 step(mlinfulm,
 scope = list(lower = mlinnulm, upper = mlinfulm),
 direction = "backward",
 trace = 0)
Call:
lm(formula = BMI_Diff ~ Gender + WorkStatus + PA_HW, data = asthmads)
Coefficients:
 (Intercept)
 GenderMale WorkStatusEmployed
 PA_HW
 -1.8527
 0.1531
 -0.1706
 -0.3151

 conduct both selection

 ···{r}
 step(mlinfulm,
 scope = list(lower = mlinnulm, upper = mlinfulm),
 direction = "both",
 trace = 0)
lm(formula = BMI_Diff ~ Gender + WorkStatus + PA_HW, data = asthmads)
Coefficients:
```

5. **Bonus 3**: Nice table with gtsummary

(Intercept)

-1.8527

· final model table based on variable selection, direction "both"

0.1531

GenderMale WorkStatusEmployed

-0.1706

PA\_HW

-0.3151

#### 7 Statistical Tests

```
1 ```{r}
2 lm(formula = BMI_Diff ~ Gender + WorkStatus + PA_HW,
3 data = asthmads) %>%
4 tbl_regression() %>%
5 bold_p()
6 ```
```

Characteristic	Beta	95% CI	p-value
Gender			
Female	_	_	
Male	0.15	-0.03, 0.33	0.093
Employment			
Unemployed	_	_	
Employed	-0.17	-0.35, 0.01	0.059
Physical Activity (total hour per week)	-0.32	-0.35, -0.28	<0.001

# 6. Bonus 4: Another package that also create nice table, sjPlot

```
1 ```{r}
2 library(sjPlot)
3
4 lm(formula = BMI_Diff ~ Gender + WorkStatus + PA_HW,
5 data = asthmads) %>%
6 tab_model()
7
```

		BMI Diff	
Predictors	Estimates	CI	p
(Intercept)	-1.85	-2.031.67	<0.001
Gender: Male	0.15	-0.03 - 0.33	0.093
Employment:	-0.17	-0.35 - 0.01	0.059
Employed			
Physical Activity(total	-0.32	-0.350.28	<0.001
hour per week)			
Observations	150		
R <sup>2</sup> / R <sup>2</sup> adjusted	0.668 / 0.661		

# 7.5 Logistic Regression

Similar to linear regression, logistic regression models the relationship between a dependent variable and one or more independent variables. However, while linear regression predicts continuous outcomes, binary logistic regression is used when the outcome is binary in nature, with two possible outcomes (0/1, yes/no, true/false). Logistic regression estimates the **probabilities** of the binary outcome

the formula is

$$\log\left(\frac{P}{1-P}\right) = \beta_0 + \beta_1 X$$

- $\cdot P$  is the probability of outcome
- $\frac{P}{1-P}$  is the odd of event occurring
- $\log\left(\frac{P}{1-P}\right)$  the log odds or logit
- X is the predictor
- $\beta_0$  is the y-intercept
- $\beta_1 X$  is the slope of the line, representing the change of probability in y for one-unit change in X

The assumptions for logistic regression are:

- each observation were **indepedent**.
- the outcome variable is binary categorical data
- The log odds of the outcome should have a linear relationship with the independent variables
- abscence (or minimal) multicollinearity
- large sample size. general rule of thumn, at least 10 cases with the least frequent outcome for each independent variable in the model
- · no extreme outlier
- · model should adequately fit the model, i.e., test for goodness-of-fit



note the similarities and differences between linear regression and logistic regression.

Example: Finding factors associated with BMI changes category

#### 7.5.1 Simple Logistic Regression

- 1. We will use the same dataset as previous, i.e., the asthmads
  - we need to do some data wrangling to categorised the BMI changes.
  - · assumed, if the BMI changes (reduced) by 2.5 kg/m<sup>2</sup>, the changes is effective

2. Conduct the simple logistic regression

```
1 ```{r}
2 slogm <- glm(BMI_DiffCat ~ PA_HW, family = binomial, asthmads)
3
4 summary(slogm)</pre>
```

3. Exponentiate the estimate and confident interval

```
(Intercept) PA_HW
0.5282878 2.9566236
2.5 % 97.5 %
(Intercept) 0.2459518 1.079780
PA_HW 1.9692051 4.872002
```

#### Note

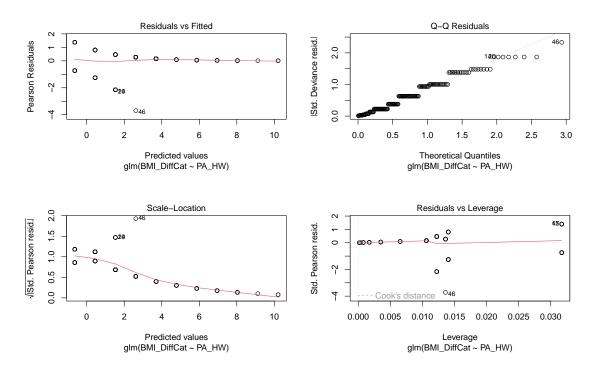
exponentiating a coefficient gives you the odds ratio associated with a one-unit increase in the corresponding predictor variable, assuming other variables are held constant

# 4. Test assumption

• similar to linear regression, simplest way for assumption testing is use plot(\_) function

```
1 ```{r}
2 #| layout-ncol: 2
```

```
plot(slogm)
5
```



#### 5. Check model fitness

AIC: 115.3

• in R, the model fitness measured with AIC, which is from the summary(\_) function

```
1 ```{r}
2 summary(slogm)
3 ```
```

Residual deviance: 111.3 on 148 degrees of freedom

Number of Fisher Scoring iterations: 6

- · we can also conduct hosmer-lemeshow test.
- the outcome however need to change to binary numeric 0/1 first.

```
asthmads <- asthmads %>%

mutate(BMI_DiffCatN = case_when(BMI_DiffCat = "Not Effective" ~ 0,

BMI_DiffCat = "Effective" ~ 1))

library(ResourceSelection)

hoslem.test(asthmads$BMI_DiffCatN,

fitted(slogm), g = 5)
```

Hosmer and Lemeshow goodness of fit (GOF) test

```
data: asthmads$BMI_DiffCatN, fitted(slogm)
X-squared = 0.86611, df = 3, p-value = 0.8336
```

- 6. Check model performance
- · model performance with accuracy, sensitivity and specificity
  - need to calculate the predicted outcome, using broom:: package

Confusion Matrix and Statistics

```
predcat

BMI_DiffCat Not Effective Effective

Not Effective 11 21

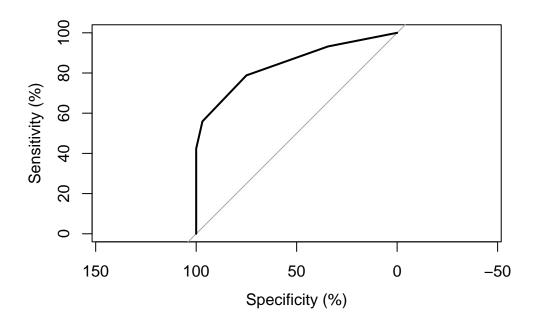
Effective 8 110
```

#### • model performance with AUROC

```
Call:
roc.data.frame(data = ., response = BMI_DiffCat, predictor = .fitted, levels = c("Not Effective",

Data: .fitted in 32 controls (BMI_DiffCat Not Effective) < 118 cases (BMI_DiffCat Effective).

Area under the curve: 84.43%
95% CI: 78.29%-90.57% (DeLong)
```



· of course if you want to customise the plot with ggplot, you can extract the value

```
threshold specificity sensitivity
 -Inf
 0.00000 1.00000000
1
2
 0.4776725
 0.34375 0.93220339
 0.78813559
3
 0.7158379
 0.75000
 0.8768814
 0.96875
 0.55932203
4
 0.9537934
 1.00000
 0.42372881
 0.9837596
 1.00000
 0.22881356
7
 0.9944329
 1.00000
 0.16101695
 0.9981084
 1.00000
 0.11016949
 0.9993592
 1.00000
 0.06779661
10 0.9997832
 1.00000
 0.02542373
11 0.9999266
 1.00000
 0.01694915
 Inf
 1.00000
 0.00000000
12
```

```
augment(slogm, type.predict = "response") %>%
mutate(predcat = cut(.fitted,
breaks = c(0, .5, 1),
```

```
labels = c("Not Effective", "Effective")),
predcat = fct_relevel(predcat, "Not Effective")) %>%

roc(BMI_DiffCat, .fitted,
levels = c("Not Effective", "Effective")) %>%

coords() %>%

arrange(sensitivity, specificity) %>%

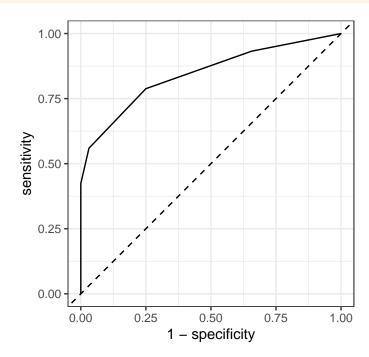
ggplot(aes(x = 1-specificity, y = sensitivity)) +

geom_line(group = 1) +

geom_abline(intercept = 0, slope = 1, linetype = 2) +

theme_bw() +

theme(aspect.ratio = 1)
```



# 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 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.

#### 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:

- 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.
- 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

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

### 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\_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

# 8 Complex Sampling Design in NHMS

# 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



some packages and functions that offer a quick data exploration:

- skimr:: package: skim(\_) function.
- summarytools:: package: dfSummary(\_) function.

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	Employement 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 - (	Cho	lestero	l Modu	le Dataset:	Variab	les 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)

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	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



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

```
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)

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"
                                            "A2107"
                        "A2106_5grp"
                                                               "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"
                        "bodyheight1"
                                            "bodyheight2"
                                                               "wc2"
[33] "bodyweight2"
[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

• 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:
                          4
                              5
                                      7
                                           8
                                                 10
                                                      11 12
                 2
                      3
                                  6
                                               9
obs
           584 274 281 263 281 307 331 319 294 245 302 338 307 333
design.PSU 27
                         11
                             12
                                     12
                                          12
                                              12
                                                  12
                13
                     13
                                 12
actual.PSU 27
                13
                     13
                         11
                             12
                                 12
                                      12
                                          12
                                              12
                                                  12
                                                      12
                                                          12
                                                               14
                                                                   12
                                                  24
            15
                16
                     17
                         18
                            19
                                 20
                                     21
                                         22
                                              23
                                                      25
                                                          26
                                                              27
                                                                   28 29
           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
actual.PSU
                     12
                         12 53 11 11
            16
                11
                                         13
                                              20
                                                  19
                                                      16
                                                         14
                                                                   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"
                                             "A2109_4grp"
                         "A2108_3grp"
[10] "A2221"
                         "A2222_7grp"
                                             "A2222_5grp"
[13] "indvid"
                         "hh_id"
                                             "state_st"
                                             "c01"
[16] "ebid"
                         "wtfinal_ncd"
[19] "c02"
                         "c03"
                                             "c03a"
[22] "c04a"
                         "c04b"
                                             "c04c"
[25] "c04d"
                         "c05"
                                             "c06"
[28] "u303"
                                             "undiagnosed_chol"
                         "known_chol"
                         "bodyweight1"
                                             "bodyweight2"
[31] "total_chol"
                                             "wc2"
[34] "bodyheight1"
                         "bodyheight2"
[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 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.

```
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.

```
total SE known_cholN 2015 40.339
```

i 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.
- Using function svymean(_) to calculate

mean SE known_cholN 0.13479 0.0051

8.2.2.5 Estimating Confidence Interval for Prevalence

- 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"

```
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(_)

8.2.2.6 Estimating the Unweighted Sample Proportion

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



Hint:

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

8.2.2.7 Estimating by Subpopulation

- 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

```
        strata_gp
        known_cholN
        se

        Urban
        1198
        32.57255

        Rural
        817
        27.44622
```

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

```
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)

```
        strata_gp
        known_cholN
        se

        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.
 - \cdot 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,</pre>
 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")
```

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

the custom function code

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

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

Number of Fisher Scoring iterations: 4

```
1 ```{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|)
             -1.83690 0.05134 -35.779 <2e-16 ***
(Intercept)
Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
(Dispersion parameter for quasibinomial family taken to be 1.000096)
```

8.3.1.2 Multiple Logistic Regression

```
__.
```

Survey design:

```
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 '**' 0.05 '.' 0.1 ' ' 1
```

svydesign(id = ~ebid, strata = ~state_st, weights = ~wtfinal_ncd,

(Dispersion parameter for quasibinomial family taken to be 1.00007)

Number of Fisher Scoring iterations: 5

data = nhms19ds, nest = T)

8.3.2 Linear Regression

8.3.2.1 Simple Linear Regression

```
1 ```{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 ***
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

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

Coefficients:

Survey design:

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

```
kcprev_state <- svyby(formula = ~known_cholN,

by = ~state,

design = nhms_surdsg,

FUN = svymean,

na.rm.all = T) %>%

as_tibble()

kcprev_state

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
                 0.106 0.00732
 3 Kelantan
 4 Melaka
                 0.154 0.0189
5 N. Sembilan 0.188 0.0293
6 Pahang 0.112 0.0138
                0.185 0.0270
7 P. Pinang
                 0.202 0.0248
8 Perak
 9 Perlis
                 0.177 0.0190
                0.120 0.0137
0.130 0.0138
10 Selangor
11 Terengganu
                 0.0836 0.0116
12 Sabah
13 Sarawak
                  0.154 0.0156
14 WP Kl
                  0.154 0.0177
                0.149 0.0154
15 WP Labuan
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/administrations
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/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.

```
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 dataaset and our dataset is consistent.

- 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
```

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

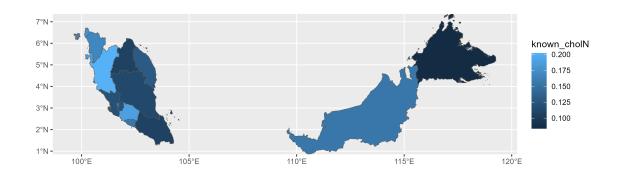
Bounding box: xmin: 99.6409 ymin: 0.85539 xmax: 119.269 ymax: 7.36098

Geodetic CRS: WGS 84
A tibble: 16 x 5

	state	known_cholN	se	code_state	geometry
	<fct></fct>	<dbl></dbl>	<dbl></dbl>	<int></int>	<multipolygon [°]=""></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
- the data wrangling code is in the next page

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

```
# A tibble: 32 x 4
  gender age_grp
```

```
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,
10
 "Male" = "male",
11
 "Female" = "female"),
12
 gender = fct_relevel(gender, "Male"),
 age_grp = case_when(age \%in\% c("75-79",
 "80-84",
 "85+") ~ "75+",
 .default = age),
17
 age_grp = fct_relevel(age_grp,
 "0-4", "5-9", "10-14",
19
 "15-19", "20-24", "25-29",
 "30-34", "35-39", "40-44",
 "45-49", "50-54", "55-59",
22
 "60-64", "65-69", "70-74",
23
 "75+"),
24
 population = population * 1000) %>%
 select(-c(date, state, ethnicity, age)) %>% # not required
 group_by(gender, age_grp, type) %>%
27
 summarise(population = sum(population, na.rm = T)) %>%
28
 ungroup()
29
```

#### 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}
 nhms_surdsg_all <- svydesign(id = ~ebid,</pre>
 strata = ~state_st,
 weights = ~wtfinal_ncd,
 data = nhms19ds_all,
 nest = TRUE)
 nhmspop19 <- svyby(formula = ~cholall,</pre>
 by = \sim A2104_grp + A2101,
 design = nhms_surdsg_all,
 FUN = svytotal) %>%
11
 as_tibble() %>%
12
 rename("gender" = "A2101",
 "age_grp" = "A2104_grp",
 "population" = "cholall") %>%
 mutate(type = "nhms",
16
 age_grp = fct_recode(age_grp,
17
 "75+" = "75 & above")) %>%
18
 select(-se)
19
 nhmspop19
```

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

## 3. Join DOSM population and NHMS population

- ullet 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

```
A tibble: 52 x 4
 gender age_grp type population
 <fct> <fct>
 <chr>>
 <dbl>
1 Male
 15-19
 1480100
 dosm
2 Male
 15-19
 nhms
 497478.
3 Male
 1675900
 20-24
 dosm
4 Male
 20-24
 nhms
 1496318.
5 Male
 25-29
 1746200
 dosm
6 Male
 25-29
 nhms
 1585427.
7 Male
 30-34
 dosm
 1502400
8 Male
 30-34
 nhms
 1412422.
9 Male
 35-39
 1304000
 dosm
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,
 1 = 65.
12
 alpha = c(.4, .4, 1, 1)),
13
 name = "") +
14
 coord_flip() +
 facet_wrap(. ~gender,
 scales = "free_x",
17
 strip.position = "bottom") +
18
 theme_bw() +
19
 theme(panel.border = element_blank(),
20
 axis.ticks.y = element_blank(),
 panel.grid.major.y = element_blank(),
22
 legend.position = "bottom",
 panel.spacing.x = unit(0, "pt"),
24
 strip.background = element_rect(colour = "black"))
25
26
```

