Data Wrangling in R

Kursus R: Pengenalan dan Praktikal (Sesi 2)

Mohd Azmi Bin Suliman

azmi.suliman@moh.gov.my

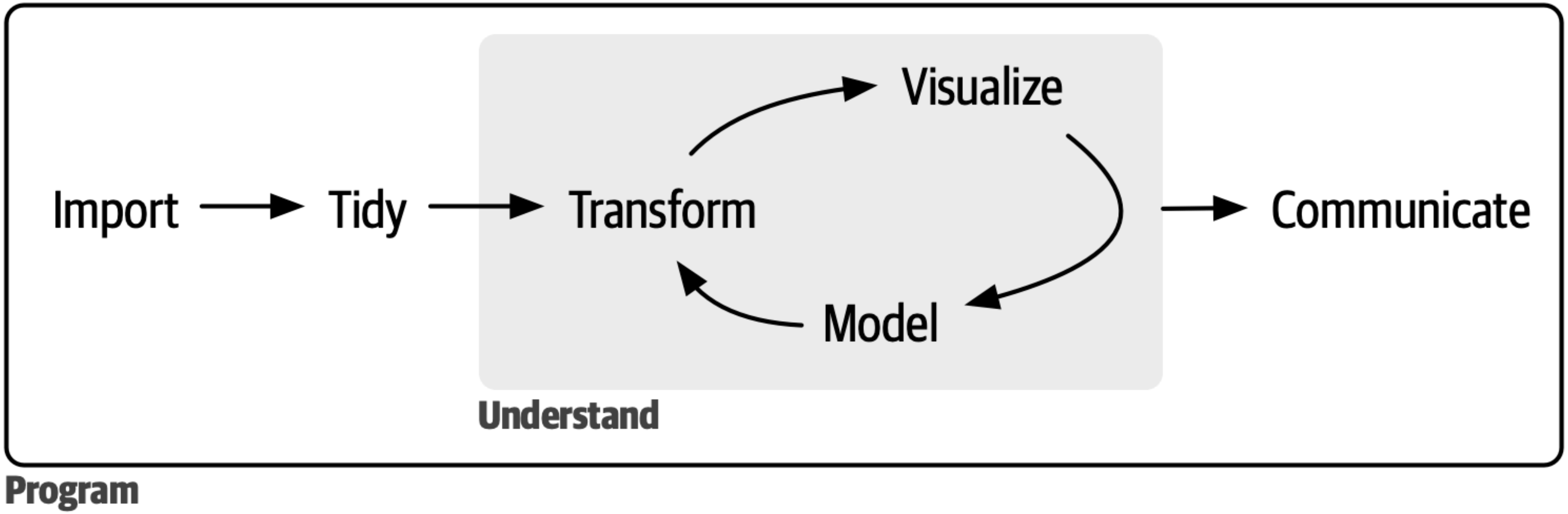
Pusat Penyelidikan Penyakit Tak Berjangkit, Institut Kesihatan Umum

Wednesday, 02 October 2024

Data Wrangling

What is Data Wrangling?

- Common data analysis look like this



source: r4ds.hadley.nz

What is Data Wrangling?

- Real world data commonly messy
- 80% of time taken spend on data cleaning
- Improving data quality > improving the accracy and efficiency
- 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

Let’s Try!

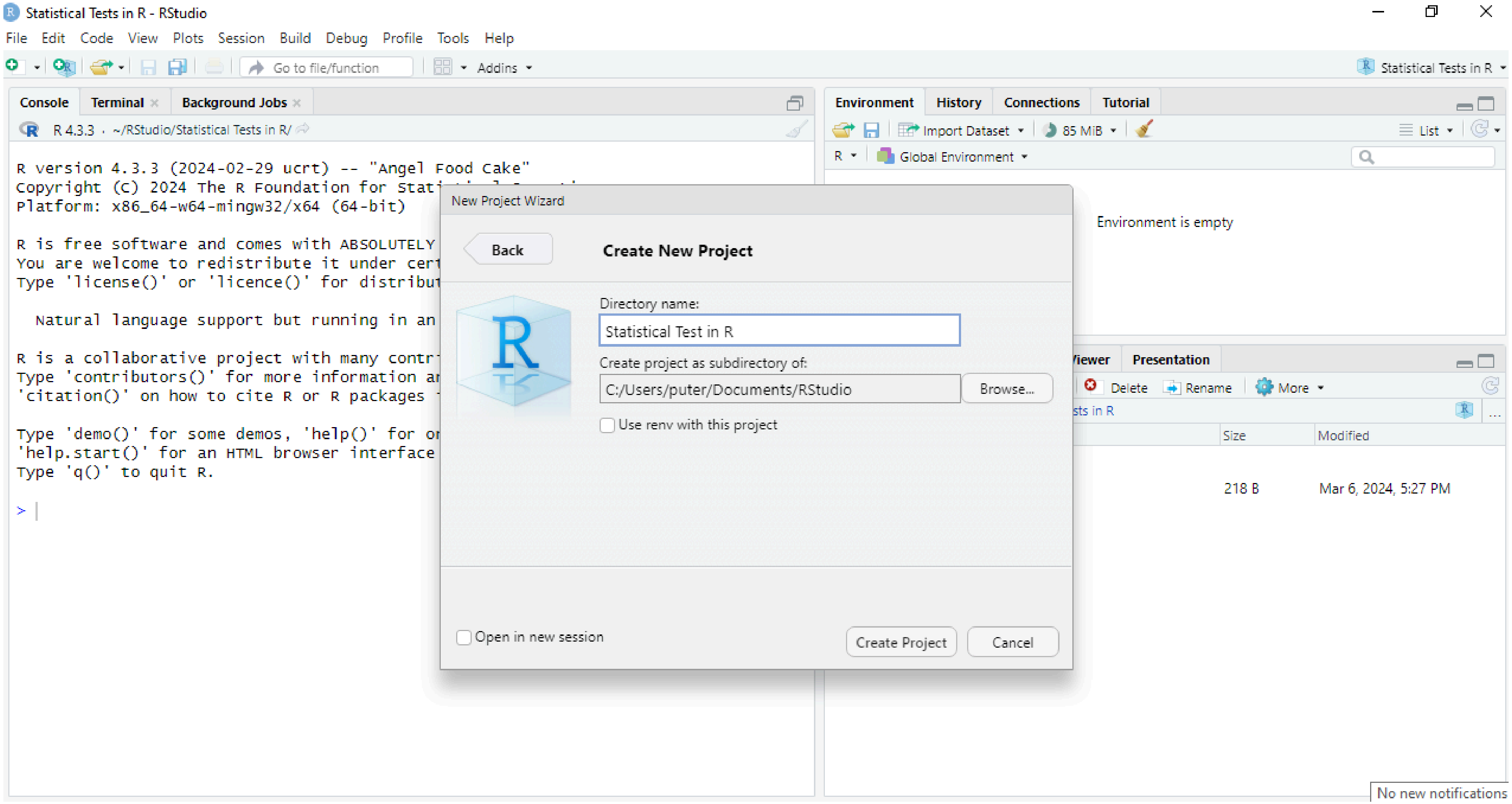
Setup your project & quarto document.

Create New Project

- RStudio allows for project management.
  - → Project as a 'container' for our work.

1. Open RStudio.
2. Create a new project.
   - `File` > `New Project` > `New Directory` > `New Project`
3. Set the name and directory.
   - Name: `Statistical Tests in R`
   - Directory: `.../Documents/RStudio`
4. Click `Create Project`

Create New Project



```
R version 4.3.3 (2024-02-29 ucrt) -- "Angel Food Cake"
Copyright (C) 2024 The R Foundation for Stati
Platform: x86_64-w64-mingw32/x64 (64-bit)

R is free software and comes with ABSOLUTELY
You are welcome to redistribute it under cert
Type 'license()' or 'licence()' for distribut

  Natural language support but running in an

R is a collaborative project with many contr
Type 'contributors()' for more information ar
'citation()' on how to cite R or R packages

Type 'demo()' for some demos, 'help()' for or
'help.start()' for an HTML browser interface
Type 'q()' to quit R.

> 
```

Directory name: Statistical Test in R

Create project as subdirectory of: C:/Users/puter/Documents/RStudio

□ Use renv with this project

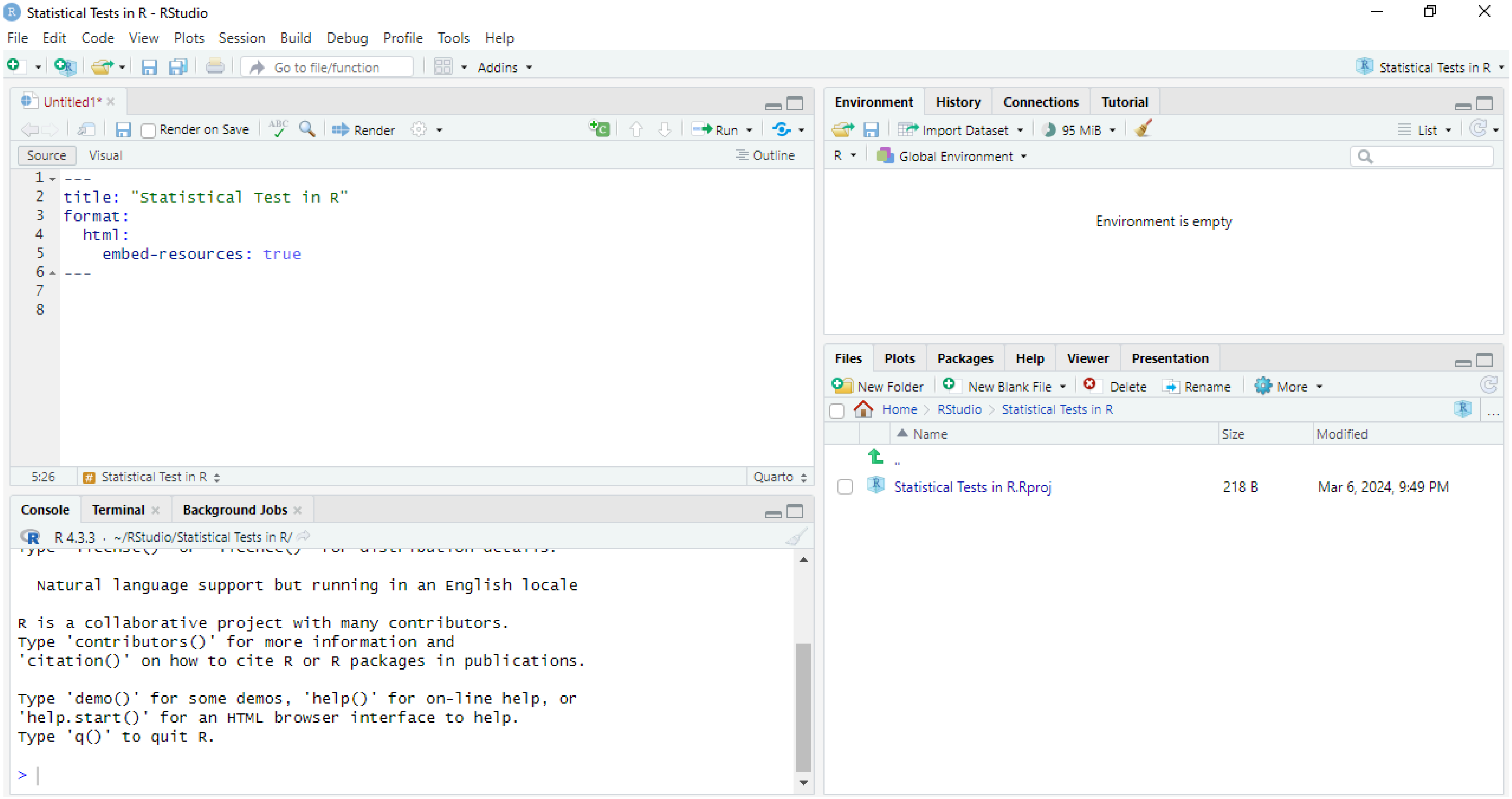
□ Open in new session

Create New Quarto Document

Quarto as R Notebook.

1. Create a new Quarto document.
    - `File` > `New File` > `Quarto Document...`
2. Set the title
    - Title: `Data Wrangling in R`
    - Untick `Use the visual editor.`
3. Click `Create Empty Document`
4. Edit the YAML
    - Add the `embed-resources: true` parameter.

Create a New Quarto Document



```
---
title: "Statistical Test in R"
format:
  html:
    embed-resources: true
---
```

```
Type 'license()' or 'licence()' for distribution details.

  Natural language support but running in an English locale

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

> |
```

Environment is empty

Name	Size	Modified
..		
Statistical Tests in R.Rproj	218 B	Mar 6, 2024, 9:49 PM

Notebook Setup

- In Quarto document, we can set one of the code chunk as our setup chunk
- This setup chunk will be run first before any other code chunk
- We can set the global options, load the packages, etc.
- Only one setup chunk is allowed in a document
- Add Level 1 header

    → `# Setup, Dataset and Library`

- Add setup code chunk

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

library(tidyverse)
```
~~~

Data Import

- SPSS
- CSV
- Online

Practical: Data Import

- Create New Level 1 Header

  → `# Data Import`

Data Import

- Commonly data were collected & stored elsewhere
- The data can be in various formats: Excel, SPSS, CSV, etc.
- The data sources includes
  - → Surveys
  - → Clinical Trials
  - → Records: Clinical, Health, Returns, Governments
- In `base` R, R can read text file with delimiter, e.g., CSV file
- To read other format, we can use additional packages
  - → `readxl` package for Excel file
  - → `haven` package for SPSS, SAS, Stata, etc.
  - → `rio` package have single function to simplify data import

Data Import (SPSS)

- In IKU, we commonly deal with SPSS data files, with `.sav` file extension.
- One of SPSS `.sav` advantage as data storage is the ability to store metadata
  - → variable labels
  - → value labels
  - → missing values
- In R, `haven` package can read SPSS data file

Data Import (SPSS)

- In R, **haven** package can read SPSS data file
  - → Add `as_factor()` function to extract the label and applied to the data frame

```r
library(haven)

asthmads_spss <- read_sav("Dataset/asthmads_spss.sav") %>%
  as_factor()
head(asthmads_spss)
```

```
# A tibble: 6 × 18
     id idR   Gender   Age WorkStatus Height Weight_Pre WC_Pre PA_HW Weight_Post
  <dbl> <chr> <fct>  <dbl> <fct>       <dbl>      <dbl>  <dbl> <dbl>       <dbl>
1     1 ejGs  Female    34 Unemployed    179       84.2     77     3        75.5
2     2 A4pG  Male      31 Unemployed    169       81.8     94     1        75.4
3     3 qkCO  Male      25 Employed      164       88.5     95     4        80.2
4     4 jcFZ  Female    33 Unemployed    136       53.2     85     2        47.7
5     5 qVSA  Male      28 Unemployed    172       71.3     90     3        61.8
6     6 wDAR  Male      33 Unemployed    178       87.3     92     2        81.8
# ℹ 8 more variables: WC_Post <dbl>, Tx2 <fct>, PEFR_Pre <dbl>,
#   PEFR_Post <dbl>, SxWheeze_Pre <fct>, SxWheeze_Post <fct>, PS_Pre <dbl>,
#   PS_Post <dbl>
```

Data Import (CSV)

- Comma separated value, CSV file is a plain text file that store tabular data
- In R, we can use `read.csv()` function to read CSV file
- We can also use `read_csv()` function from `readr` package, which offer more granular options

  → Since we already load `tidyverse` package in the setup code chunk, no need to load `readr` package

```r
asthmads_csvraw <- read_csv("Dataset/asthmads_nolab.csv")
asthmads_csvraw
```

Data Import (CSV)

```r
asthmads_csvraw <- read_csv("Dataset/asthmads_nolab.csv")
asthmads_csvraw
```

```
# A tibble: 150 × 19
    ...1    id idR   Gender   Age WorkStatus Height Weight_Pre WC_Pre PA_HW
   <dbl> <dbl> <chr>  <dbl> <dbl>      <dbl>  <dbl>      <dbl>  <dbl> <dbl>
 1     1     1 ejGs       2    34          2    179       84.2     77     3
 2     2     2 A4pG       1    31          2    169       81.8     94     1
 3     3     3 qkCO       1    25          1    164       88.5     95     4
 4     4     4 jcFZ       2    33          2    136       53.2     85     2
 5     5     5 qVSA       1    28          2    172       71.3     90     3
 6     6     6 wDAR       1    33          2    178       87.3     92     2
 7     7     7 FuAU       2    31          2    140       48.8     80     4
 8     8     8 fnKz       2    34          1    140       49.1     82     2
 9     9     9 0YTi       1    31          1    171       60.1     85     3
10    10    10 pfMa       1    28          1    163       93.1    101     5
# ℹ 140 more rows
# ℹ 9 more variables: Weight_Post <dbl>, WC_Post <dbl>, Tx2 <dbl>,
#   PEFR_Pre <dbl>, PEFR_Post <dbl>, SxWheeze_Pre <dbl>, SxWheeze_Post <dbl>,
#   PS_Pre <dbl>, PS_Post <dbl>
```

Data Import (CSV)

- Unlike SPSS SAV file, CSV file does not store metadata
- Some CSV file store the ‘*raw*’ data, without any label or value label.

```r
asthmads_csvraw <- read_csv("Dataset/asthmads_nolab.csv")
asthmads_csvraw
```

```
# A tibble: 150 × 19
    ...1    id idR   Gender   Age WorkStatus Height Weight_Pre WC_Pre PA_HW
   <dbl> <dbl> <chr>  <dbl> <dbl>      <dbl>  <dbl>      <dbl>  <dbl> <dbl>
 1     1     1 ejGs       2    34          2    179       84.2     77     3
 2     2     2 A4pG       1    31          2    169       81.8     94     1
 3     3     3 qkCO       1    25          1    164       88.5     95     4
 4     4     4 jcFZ       2    33          2    136       53.2     85     2
 5     5     5 qVSA       1    28          2    172       71.3     90     3
 6     6     6 wDAR       1    33          2    178       87.3     92     2
 7     7     7 FuAU       2    31          2    140       48.8     80     4
 8     8     8 fnKz       2    34          1    140       49.1     82     2
 9     9     9 0YTi       1    31          1    171       60.1     85     3
10    10    10 pfMa       1    28          1    163       93.1    101     5
# ℹ 140 more rows
# ℹ 9 more variables: Weight_Post <dbl>, WC_Post <dbl>, Tx2 <dbl>,
#   PEFR_Pre <dbl>, PEFR_Post <dbl>, SxWheeze_Pre <dbl>, SxWheeze_Post <dbl>,
#   PS_Pre <dbl>, PS_Post <dbl>
```

Data Import (CSV)

- Unlike SPSS SAV file, CSV file does not store metadata
- Some CSV file store the '*label*' data, with string as value - but may not contain the underlying level for factors column.

```r
asthmads_csvlab <- read_csv("Dataset/asthmads_csv.csv")
head(asthmads_csvlab)
```

```
# A tibble: 6 × 19
   ...1    id idR   Gender   Age WorkStatus Height Weight_Pre WC_Pre PA_HW
  <dbl> <dbl> <chr> <chr>  <dbl> <chr>       <dbl>      <dbl>  <dbl> <dbl>
1     1     1 ejGs  Female    34 Unemployed    179       84.2     77     3
2     2     2 A4pG  Male      31 Unemployed    169       81.8     94     1
3     3     3 qkCO  Male      25 Employed      164       88.5     95     4
4     4     4 jcFZ  Female    33 Unemployed    136       53.2     85     2
5     5     5 qVSA  Male      28 Unemployed    172       71.3     90     3
6     6     6 wDAR  Male      33 Unemployed    178       87.3     92     2
# ℹ 9 more variables: Weight_Post <dbl>, WC_Post <dbl>, Tx2 <chr>,
#   PEFR_Pre <dbl>, PEFR_Post <dbl>, SxWheeze_Pre <chr>, SxWheeze_Post <chr>,
#   PS_Pre <dbl>, PS_Post <dbl>
```

```r
levels(asthmads_csvlab$Gender)
```

```
NULL
```

Data Import (Online)

- We can also import data from online source
- For example, we can import data from DOSM Open Data Portal
  - → https://open.dosm.gov.my/data-catalogue
  - → Population by State

```r
my_state_pop <- read_csv("https://storage.dosm.gov.my/population/population_st
head(my_state_pop)
```

```
# A tibble: 6 × 6
  state date       sex   age     ethnicity population
  <chr> <date>     <chr> <chr>   <chr>          <dbl>
1 Johor 1970-01-01 both  overall overall        1326.
2 Johor 1970-01-01 both  0-4     overall         210.
3 Johor 1970-01-01 both  5-9     overall         216.
4 Johor 1970-01-01 both  10-14   overall         192.
5 Johor 1970-01-01 both  15-19   overall         153.
6 Johor 1970-01-01 both  20-24   overall         111.
```

Data Import (Online)

- We can also import data from online source
- Or if you prefer to save the data on your disk, you can download directly in R
  - → using `download.file()` function
  - → add `mode = "wb"` parameter to download binary file (maximum compatibility)

```r
download.file(url = "https://storage.dosm.gov.my/population/population_state.c
              destfile = "my_state_pop.csv",
              mode = "wb")
my_state_pop <- read_csv("my_state_pop.csv")
my_state_pop
```

Basic Data Exploration

Practical: Data Exploration Setup

- Add Level 1 Header

  → `# Basic Data Exploration`

Basic Data Exploration

- Usually done after data import, and maybe repeated several times during data wrangling process
- Common data exploration includes
  - → `summary()`
  - → `str()`
  - → `head()` / `tail()`
  - → `glimpse()` (from `tidyverse` package)
  - → `dfSummary()` (from `summarytools` package)

Practical: `summary()`

- `summary()` function is a generic function used to summarised R object.
- For data frame, `summary()` will return the summary statistics for each column

```r
summary(asthmads_spss)
```

```
      id              idR                 Gender          Age       
Min.   :  1.00   Length:150          Male  :86   Min.   :25.00  
1st Qu.: 38.25   Class :character    Female:64   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  
Max.   :150.00                                   Max.   :35.00  
     WorkStatus      Height        Weight_Pre          WC_Pre      
Employed  :70   Min.   :129.0   Min.   : 36.60   Min.   : 70.00  
Unemployed:80   1st Qu.:151.0   1st Qu.: 58.25   1st Qu.: 83.25  
                Median :167.0   Median : 77.70   Median : 89.00  
                Mean   :164.0   Mean   : 75.89   Mean   : 88.58  
                3rd Qu.:175.8   3rd Qu.: 89.45   3rd Qu.: 95.00  
                Max.   :195.0   Max.   :129.10   Max.   :104.00  
    PA_HW         Weight_Post        WC_Post            Tx2        PEFR_Pre    
Min.   : 0.000   Min.   : 31.40   Min.   :59.00   Placebo:75   Min.   :349.0  
1st Qu.: 1.000   1st Qu.: 51.95   1st Qu.:75.00   Drug A :46   1st Qu.:412.0  
Median : 2.000   Median : 68.95   Median :81.00   Drug B :29   Median :431.0  
Mean   : 2.633   Mean   : 68.64   Mean   :80.53                Mean   :434.6  
```

Practical: summary()

- summary() can also be done to specific column/s

```
asthmads_spss %>%
  select(Age) %>%
  summary()
```

```
     Age       
Min.   :25.00  
1st Qu.:27.00  
Median :30.00  
Mean   :30.15  
3rd Qu.:33.00  
Max.   :35.00  
```

```
asthmads_spss %>%
  select(Gender:Height) %>%
  summary()
```

```
   Gender         Age             WorkStatus        Height     
Male  :86   Min.   :25.00   Employed  :70   Min.   :129.0  
Female:64   1st Qu.:27.00   Unemployed:80   1st Qu.:151.0  
            Median :30.00                   Median :167.0  
            Mean   :30.15                   Mean   :164.0  
            3rd Qu.:33.00                   3rd Qu.:175.8  
            Max.   :35.00                   Max.   :195.0  
```

Practical: `str()`

- `str()` function is used to display the internal structure of an R object

```r
str(asthmads_spss)
```

```
tibble [150 × 18] (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"
 $ idR          : chr [1:150] "ejGs" "A4pG" "qkCO" "jcFZ" ...
  ..- attr(*, "format.spss")= chr "A4"
 $ Gender       : Factor w/ 2 levels "Male","Female": 2 1 1 2 1 1 2 2 1 1 ...
 $ Age          : num [1:150] 34 31 25 33 28 33 31 34 31 28 ...
  ..- 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"
 $ Height       : num [1:150] 179 169 164 136 172 178 140 140 171 163 ...
  ..- 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"
 $ WC_Pre       : num [1:150] 77 94 95 85 90 92 80 82 85 101 ...
  ..- attr(*, "label")= chr "Waist Circumference (cm) - before"
```

Practical: `str()`

- `str()` function can also be done to specific column/s

```
asthmads_spss %>%
  select(Age) %>%
  str()
```

```
tibble [150 × 1] (S3: tbl_df/tbl/data.frame)
 $ Age: num [1:150] 34 31 25 33 28 33 31 34 31 28 ...
  ..- attr(*, "label")= chr "Age (year)"
  ..- attr(*, "format.spss")= chr "F8.2"
```

```
asthmads_spss %>%
  select(Gender:Height) %>%
  str()
```

```
tibble [150 × 4] (S3: tbl_df/tbl/data.frame)
 $ Gender    : Factor w/ 2 levels "Male","Female": 2 1 1 2 1 1 2 2 1 1 ...
 $ Age       : num [1:150] 34 31 25 33 28 33 31 34 31 28 ...
  ..- 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"
 $ Height    : num [1:150] 179 169 164 136 172 178 140 140 171 163 ...
  ..- attr(*, "label")= chr "Height (cm)"
  ..- attr(*, "format.spss")= chr "F8.2"
```

Practical: `head()` and `tail()`

- `head()` and `tail()` function is used to display the first and last few rows of the data frame

```r
head(asthmads_spss)
```

```r
tail(asthmads_spss)
```

```
# A tibble: 6 × 18
     id idR   Gender   Age WorkStatus Height Weight_Pre WC_Pre PA_HW Weight_Post
  <dbl> <chr> <fct>  <dbl> <fct>       <dbl>      <dbl>  <dbl> <dbl>       <dbl>
1   145 JOCA  Male      26 Unemployed    170       83.6     95     4        72.8
2   146 byQi  Male      30 Unemployed    166       81.4     91     0        76.1
3   147 7y8o  Female    32 Unemployed    149       60.2     81     2        52.2
4   148 mH0E  Female    31 Employed      156       65.3     80     0        62.7
5   149 22pz  Female    32 Unemployed    144       40       73     1        36.4
6   150 m0bj  Male      34 Unemployed    172       69.2     88     1        63.2
# ℹ 8 more variables: WC_Post <dbl>, Tx2 <fct>, PEFR_Pre <dbl>,
#   PEFR_Post <dbl>, SxWheeze_Pre <fct>, SxWheeze_Post <fct>, PS_Pre <dbl>,
#   PS_Post <dbl>
```

Practical: `glimpse()` from `tidyverse` package

- `glimpse()` function is used to display the internal structure of an R object, similar to `str()`

```r
glimpse(asthmads_spss)
```

```
Rows: 150
Columns: 18
$ id            <dbl> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 1…
$ idR           <chr> "ejGs", "A4pG", "qkCO", "jcFZ", "qVSA", "wDAR", "FuAU", …
$ Gender        <fct> Female, Male, Male, Female, Male, Male, Female, Female, …
$ Age           <dbl> 34, 31, 25, 33, 28, 33, 31, 34, 31, 28, 34, 25, 31, 35, …
$ WorkStatus    <fct> Unemployed, Unemployed, Employed, Unemployed, Unemployed…
$ Height        <dbl> 179, 169, 164, 136, 172, 178, 140, 140, 171, 163, 164, 1…
$ Weight_Pre    <dbl> 84.2, 81.8, 88.5, 53.2, 71.3, 87.3, 48.8, 49.1, 60.1, 93…
$ WC_Pre        <dbl> 77, 94, 95, 85, 90, 92, 80, 82, 85, 101, 93, 89, 99, 95,…
$ PA_HW         <dbl> 3, 1, 4, 2, 3, 2, 4, 2, 3, 5, 5, 6, 3, 3, 4, 4, 1, 1, 0,…
$ Weight_Post   <dbl> 75.5, 75.4, 80.2, 47.7, 61.8, 81.8, 41.9, 44.0, 52.1, 83…
$ WC_Post       <dbl> 72, 89, 90, 78, 83, 84, 73, 71, 75, 94, 81, 80, 86, 93, …
$ Tx2           <fct> Placebo, Placebo, Placebo, Placebo, Drug A, Placebo, Dru…
$ PEFR_Pre      <dbl> 397, 472, 476, 416, 452, 484, 366, 435, 425, 437, 433, 4…
$ PEFR_Post     <dbl> 355, 445, 481, 382, 475, 497, 336, 413, 434, 413, 422, 4…
$ SxWheeze_Pre  <fct> No, Yes, Yes, No, No, Yes, Yes, No, No, No, Yes, No, Yes…
$ SxWheeze_Post <fct> No, Yes, Yes, No, No, No, No, No, No, No, No, No, Yes, N…
$ PS_Pre        <dbl> 5, 4, 4, 5, 4, 6, 5, 5, 5, 5, 4, 6, 6, 5, 5, 5, 5, 6, 6,…
```

Practical: `dfSummary()` from `summarytools` package

- `dfSummary()` function is used to display the summary statistics for each column

```r
library(summarytools)

asthmads_spss %>%
  dfSummary() %>%
  stview()
```

Basic Data Wrangling

Practical: Data Wrangling Setup

- Add Level 1 Header

  → `# Data Transformation`

Basic Data Wrangling, Row Operations

Row Operations

- `filter()`
- `arrange()`
- `distinct()`

Row Operation

- Common row operation includes
  - → `filter()`, which subset the data based on condition, without changing the order of the data
  - → `arrange()`, which sort the data based on condition
  - → `distinct()`, which find rows with unique value

Row Operation

- Create new level 2 header

  → `## Data Wrangling: Row Operations`

Row Operation: Filter

- `filter()` function is used to subset the data based on condition
- The order of the data will not be changed

```r
asthmads_spss %>%
  filter(Gender == "Female") %>%
  head()
```

```
# A tibble: 6 × 18
     id idR   Gender   Age WorkStatus Height Weight_Pre WC_Pre PA_HW Weight_Post
  <dbl> <chr> <fct>  <dbl> <fct>       <dbl>      <dbl>  <dbl> <dbl>       <dbl>
1     1 ejGs  Female    34 Unemployed    179       84.2     77     3        75.5
2     4 jcFZ  Female    33 Unemployed    136       53.2     85     2        47.7
3     7 FuAU  Female    31 Unemployed    140       48.8     80     4        41.9
4     8 fnKz  Female    34 Employed      140       49.1     82     2        44
5    12 nxm1  Female    25 Unemployed    149       69.6     89     6        61.8
6    15 tMX1  Female    32 Employed      147       59.1     84     4        51.9
# ℹ 8 more variables: WC_Post <dbl>, Tx2 <fct>, PEFR_Pre <dbl>,
#   PEFR_Post <dbl>, SxWheeze_Pre <fct>, SxWheeze_Post <fct>, PS_Pre <dbl>,
#   PS_Post <dbl>
```

Row Operation: Filter

- `filter()` function is used to subset the data based on condition
- Use relational operator to set the condition

```r
asthmads_spss %>%
  filter(Gender != "Female") %>%
  head()
```

```
# A tibble: 6 × 18
     id idR   Gender   Age WorkStatus Height Weight_Pre WC_Pre PA_HW Weight_Post
  <dbl> <chr> <fct>  <dbl> <fct>       <dbl>      <dbl>  <dbl> <dbl>       <dbl>
1     2 A4pG  Male      31 Unemployed    169       81.8     94     1        75.4
2     3 qkCO  Male      25 Employed      164       88.5     95     4        80.2
3     5 qVSA  Male      28 Unemployed    172       71.3     90     3        61.8
4     6 wDAR  Male      33 Unemployed    178       87.3     92     2        81.8
5     9 0YTi  Male      31 Employed      171       60.1     85     3        52.1
6    10 pfMa  Male      28 Employed      163       93.1    101     5        83  
# ℹ 8 more variables: WC_Post <dbl>, Tx2 <fct>, PEFR_Pre <dbl>,
#   PEFR_Post <dbl>, SxWheeze_Pre <fct>, SxWheeze_Post <fct>, PS_Pre <dbl>,
#   PS_Post <dbl>
```

Row Operation: Filter

- `filter()` function is used to subset the data based on condition
- Can apply multiple condition

```
asthmads_spss %>%
  filter(Gender == "Female",
         Age <= 45) %>%
  head()
```

```
# A tibble: 6 × 18
     id idR   Gender   Age WorkStatus Height Weight_Pre WC_Pre PA_HW Weight_Post
  <dbl> <chr> <fct>  <dbl> <fct>       <dbl>      <dbl>  <dbl> <dbl>       <dbl>
1     1 ejGs  Female    34 Unemployed    179       84.2     77     3        75.5
2     4 jcFZ  Female    33 Unemployed    136       53.2     85     2        47.7
3     7 FuAU  Female    31 Unemployed    140       48.8     80     4        41.9
4     8 fnKz  Female    34 Employed      140       49.1     82     2        44  
5    12 nxm1  Female    25 Unemployed    149       69.6     89     6        61.8
6    15 tMX1  Female    32 Employed      147       59.1     84     4        51.9
# ℹ 8 more variables: WC_Post <dbl>, Tx2 <fct>, PEFR_Pre <dbl>,
#   PEFR_Post <dbl>, SxWheeze_Pre <fct>, SxWheeze_Post <fct>, PS_Pre <dbl>,
#   PS_Post <dbl>
```

Row Operation: Filter

- `filter()` function is used to subset the data based on condition
- Can use logical operator

```r
asthmads_spss %>%
  filter(Gender == "Female" & Age <= 45) %>%
  head()
```

```
# A tibble: 6 × 18
     id idR   Gender   Age WorkStatus Height Weight_Pre WC_Pre PA_HW Weight_Post
  <dbl> <chr> <fct>  <dbl> <fct>       <dbl>      <dbl>  <dbl> <dbl>       <dbl>
1     1 ejGs  Female    34 Unemployed    179       84.2     77     3        75.5
2     4 jcFZ  Female    33 Unemployed    136       53.2     85     2        47.7
3     7 FuAU  Female    31 Unemployed    140       48.8     80     4        41.9
4     8 fnKz  Female    34 Employed      140       49.1     82     2        44  
5    12 nxm1  Female    25 Unemployed    149       69.6     89     6        61.8
6    15 tMX1  Female    32 Employed      147       59.1     84     4        51.9
# ℹ 8 more variables: WC_Post <dbl>, Tx2 <fct>, PEFR_Pre <dbl>,
#   PEFR_Post <dbl>, SxWheeze_Pre <fct>, SxWheeze_Post <fct>, PS_Pre <dbl>,
#   PS_Post <dbl>
```

Row Operation: Filter

- `filter()` function is used to subset the data based on condition
- Can use logical operator

```r
asthmads_spss %>%
  filter(Gender == "Female" | Age <= 45) %>%
  head()
```

```
# A tibble: 6 × 18
     id idR   Gender   Age WorkStatus Height Weight_Pre WC_Pre PA_HW Weight_Post
  <dbl> <chr> <fct>  <dbl> <fct>       <dbl>      <dbl>  <dbl> <dbl>       <dbl>
1     1 ejGs  Female    34 Unemployed    179       84.2     77     3        75.5
2     2 A4pG  Male      31 Unemployed    169       81.8     94     1        75.4
3     3 qkCO  Male      25 Employed      164       88.5     95     4        80.2
4     4 jcFZ  Female    33 Unemployed    136       53.2     85     2        47.7
5     5 qVSA  Male      28 Unemployed    172       71.3     90     3        61.8
6     6 wDAR  Male      33 Unemployed    178       87.3     92     2        81.8
# ℹ 8 more variables: WC_Post <dbl>, Tx2 <fct>, PEFR_Pre <dbl>,
#   PEFR_Post <dbl>, SxWheeze_Pre <fct>, SxWheeze_Post <fct>, PS_Pre <dbl>,
#   PS_Post <dbl>
```

Row Operation: Filer

- `filter()` function is used to subset the data based on condition
- to select several character string, use `%in%` operator

```r
asthmads_spss %>%
  filter(Tx2 %in% c("Drug A", "Drug B")) %>%
  head()
```

```
# A tibble: 6 × 18
     id idR   Gender   Age WorkStatus Height Weight_Pre WC_Pre PA_HW Weight_Post
  <dbl> <chr> <fct>  <dbl> <fct>       <dbl>      <dbl>  <dbl> <dbl>       <dbl>
1     5 qVSA  Male      28 Unemployed    172       71.3     90     3        61.8
2     7 FuAU  Female    31 Unemployed    140       48.8     80     4        41.9
3    12 nxm1  Female    25 Unemployed    149       69.6     89     6        61.8
4    14 qYJb  Male      35 Employed      175      105.      95     3        93.2
5    16 BuPu  Female    34 Unemployed    147       54.9     80     4        48.8
6    17 Cdab  Male      30 Unemployed    179       77.6     89     1        73.4
# ℹ 8 more variables: WC_Post <dbl>, Tx2 <fct>, PEFR_Pre <dbl>,
#   PEFR_Post <dbl>, SxWheeze_Pre <fct>, SxWheeze_Post <fct>, PS_Pre <dbl>,
#   PS_Post <dbl>
```

Row Operation: Arrange

- `arrange()` function is used to sort the data based on condition

```r
asthmads_spss %>%
  arrange(Age) %>%
  head()
```

```
# A tibble: 6 × 18
     id idR   Gender   Age WorkStatus Height Weight_Pre WC_Pre PA_HW Weight_Post
  <dbl> <chr> <fct>  <dbl> <fct>       <dbl>      <dbl>  <dbl> <dbl>       <dbl>
1     3 qkCO  Male      25 Employed      164       88.5     95     4        80.2
2    12 nxm1  Female    25 Unemployed    149       69.6     89     6        61.8
3    25 GfAA  Male      25 Unemployed    183       67.7     85     2        60.2
4    63 pIXb  Male      25 Unemployed    192      129.      99     2       120. 
5    82 7znc  Male      25 Employed      174       67.7     85     2        60.7
6    86 9q3d  Female    25 Employed      151       80.8     94     6        72.7
# ℹ 8 more variables: WC_Post <dbl>, Tx2 <fct>, PEFR_Pre <dbl>,
#   PEFR_Post <dbl>, SxWheeze_Pre <fct>, SxWheeze_Post <fct>, PS_Pre <dbl>,
#   PS_Post <dbl>
```

Row Operation: Arrange

- `arrange()` function is used to sort the data based on condition
- add `desc()` function to sort in descending order

```r
asthmads_spss %>%
  arrange(desc(Age)) %>%
  head()
```

```
# A tibble: 6 × 18
     id idR   Gender   Age WorkStatus Height Weight_Pre WC_Pre PA_HW Weight_Post
  <dbl> <chr> <fct>  <dbl> <fct>       <dbl>      <dbl>  <dbl> <dbl>       <dbl>
1    14 qYJb  Male      35 Employed      175       105.     95     3        93.2
2    31 plm0  Male      35 Employed      182       106.     95     0        96.9
3    47 O99x  Female    35 Unemployed    140        42.7    81     2        39.5
4    52 mFdG  Female    35 Unemployed    139        46.9    79     2        43.1
5    72 R9Ht  Male      35 Unemployed    164        81.3    94     2        75.4
6    79 XWDs  Female    35 Employed      168        80.4    85     4        71.4
# ℹ 8 more variables: WC_Post <dbl>, Tx2 <fct>, PEFR_Pre <dbl>,
#   PEFR_Post <dbl>, SxWheeze_Pre <fct>, SxWheeze_Post <fct>, PS_Pre <dbl>,
#   PS_Post <dbl>
```

Row Operation: Arrange

- `arrange()` function is used to sort the data based on condition
- can use multiple column

```r
asthmads_spss %>%
  arrange(Age, Weight_Pre) %>%
  head()
```

```
# A tibble: 6 × 18
     id idR   Gender   Age WorkStatus Height Weight_Pre WC_Pre PA_HW Weight_Post
  <dbl> <chr> <fct>  <dbl> <fct>       <dbl>      <dbl>  <dbl> <dbl>       <dbl>
1    25 GfAA  Male      25 Unemployed    183       67.7     85     2        60.2
2    82 7znc  Male      25 Employed      174       67.7     85     2        60.7
3    12 nxml  Female    25 Unemployed    149       69.6     89     6        61.8
4   121 hJ5a  Male      25 Employed      177       77.5     92     1        70  
5    91 iWos  Male      25 Employed      160       78       94     3        69.9
6    86 9q3d  Female    25 Employed      151       80.8     94     6        72.7
# ℹ 8 more variables: WC_Post <dbl>, Tx2 <fct>, PEFR_Pre <dbl>,
#   PEFR_Post <dbl>, SxWheeze_Pre <fct>, SxWheeze_Post <fct>, PS_Pre <dbl>,
#   PS_Post <dbl>
```

Row Operation: Distinct

- `distinct()` function is used to find rows with unique value
  → set parameter `.keep_all = TRUE` to keep all columns

```r
asthmads_spss %>%
  distinct(Age, .keep_all = TRUE) %>%
  head()
```

```
# A tibble: 6 × 18
     id idR   Gender   Age WorkStatus Height Weight_Pre WC_Pre PA_HW Weight_Post
  <dbl> <chr> <fct>  <dbl> <fct>       <dbl>      <dbl>  <dbl> <dbl>       <dbl>
1     1 ejGs  Female    34 Unemployed    179       84.2     77     3        75.5
2     2 A4pG  Male      31 Unemployed    169       81.8     94     1        75.4
3     3 qkCO  Male      25 Employed      164       88.5     95     4        80.2
4     4 jcFZ  Female    33 Unemployed    136       53.2     85     2        47.7
5     5 qVSA  Male      28 Unemployed    172       71.3     90     3        61.8
6    14 qYJb  Male      35 Employed      175      105.      95     3        93.2
# ℹ 8 more variables: WC_Post <dbl>, Tx2 <fct>, PEFR_Pre <dbl>,
#   PEFR_Post <dbl>, SxWheeze_Pre <fct>, SxWheeze_Post <fct>, PS_Pre <dbl>,
#   PS_Post <dbl>
```

Row Operation: Distinct

- `distinct()` function is used to find rows with unique value
  - → If column is specified, only the specified column will be used to find unique value
  - → Multiple column can be specified

```r
asthmads_spss %>%
  distinct(Gender, WorkStatus, Tx2) %>%
  head()
```

```
# A tibble: 6 × 3
  Gender WorkStatus Tx2
  <fct>  <fct>      <fct>
1 Female Unemployed Placebo
2 Male   Unemployed Placebo
3 Male   Employed   Placebo
4 Male   Unemployed Drug A
5 Female Unemployed Drug A
6 Female Employed   Placebo
```

Basic Data Wrangling, Column Operations

Column Operations

- `mutate()`
- `select()`
- `rename()`
- `relocate`()`

Column Operation

- Common column operation includes
  - → `mutate()`, which add new column or modify existing column
  - → `select()`, which select column
  - → `rename()`, which rename column
  - → `relocate()`, which change the order of the column

Column Operation

- Create new level 2 header

  → `## Data Wrangling: Column Operations`

Column Operation: Mutate

- `mutate()` function is used to add new column or modify existing column
    - → if the variable exist, it will be overwritten, and placed at current column

```r
asthmads_spss %>%
  mutate(Height = Height/100) %>%
  head()
```

```
# A tibble: 6 × 18
     id idR   Gender   Age WorkStatus Height Weight_Pre WC_Pre PA_HW Weight_Post
  <dbl> <chr> <fct>  <dbl> <fct>       <dbl>      <dbl>  <dbl> <dbl>       <dbl>
1     1 ejGs  Female    34 Unemployed   1.79       84.2     77     3        75.5
2     2 A4pG  Male      31 Unemployed   1.69       81.8     94     1        75.4
3     3 qkCO  Male      25 Employed     1.64       88.5     95     4        80.2
4     4 jcFZ  Female    33 Unemployed   1.36       53.2     85     2        47.7
5     5 qVSA  Male      28 Unemployed   1.72       71.3     90     3        61.8
6     6 wDAR  Male      33 Unemployed   1.78       87.3     92     2        81.8
# ℹ 8 more variables: WC_Post <dbl>, Tx2 <fct>, PEFR_Pre <dbl>,
#   PEFR_Post <dbl>, SxWheeze_Pre <fct>, SxWheeze_Post <fct>, PS_Pre <dbl>,
#   PS_Post <dbl>
```

Column Operation: Mutate

- `mutate()` function is used to add new column or modify existing column
  - → if the variable not exist, it will be created, and placed at the final column

```r
asthmads_spss %>%
  mutate(BMI_Pre = Weight_Pre/(Height/100)^2) %>%
  head()
```

```
# A tibble: 6 × 19
     id idR   Gender   Age WorkStatus Height Weight_Pre WC_Pre PA_HW Weight_Post
  <dbl> <chr> <fct>  <dbl> <fct>       <dbl>      <dbl>  <dbl> <dbl>       <dbl>
1     1 ejGs  Female    34 Unemployed    179       84.2     77     3        75.5
2     2 A4pG  Male      31 Unemployed    169       81.8     94     1        75.4
3     3 qkCO  Male      25 Employed      164       88.5     95     4        80.2
4     4 jcFZ  Female    33 Unemployed    136       53.2     85     2        47.7
5     5 qVSA  Male      28 Unemployed    172       71.3     90     3        61.8
6     6 wDAR  Male      33 Unemployed    178       87.3     92     2        81.8
# ℹ 9 more variables: WC_Post <dbl>, Tx2 <fct>, PEFR_Pre <dbl>,
#   PEFR_Post <dbl>, SxWheeze_Pre <fct>, SxWheeze_Post <fct>, PS_Pre <dbl>,
#   PS_Post <dbl>, BMI_Pre <dbl>
```

Column Operation: Mutate

- `mutate()` function is used to add new column or modify existing column
- Several parameters can be specified
  - → `.keep` parameter to control which columns are retained in the output
  - → `.before` and `.after` parameter to control the position of the new column

Column Operation: Select

- `select()` function is used to select column
  - → Use `:` operator to select range of column

```
asthmads_spss %>%
  select(Gender:Weight_Pre) %>%
  head()
```

```
# A tibble: 6 × 5
  Gender    Age WorkStatus Height Weight_Pre
  <fct>   <dbl> <fct>       <dbl>      <dbl>
1 Female     34 Unemployed    179       84.2
2 Male       31 Unemployed    169       81.8
3 Male       25 Employed      164       88.5
4 Female     33 Unemployed    136       53.2
5 Male       28 Unemployed    172       71.3
6 Male       33 Unemployed    178       87.3
```

Column Operation: Select

- `select()` function is used to select column
  - → Use `-` operator to exclude column

```r
asthmads_spss %>%
  select(-c(Weight_Pre:Weight_Post)) %>%
  head()
```

```
# A tibble: 6 × 14
     id idR   Gender   Age WorkStatus Height WC_Post Tx2     PEFR_Pre PEFR_Post
  <dbl> <chr> <fct>  <dbl> <fct>       <dbl>   <dbl> <fct>      <dbl>     <dbl>
1     1 ejGs  Female    34 Unemployed    179      72 Placebo      397       355
2     2 A4pG  Male      31 Unemployed    169      89 Placebo      472       445
3     3 qkCO  Male      25 Employed      164      90 Placebo      476       481
4     4 jcFZ  Female    33 Unemployed    136      78 Placebo      416       382
5     5 qVSA  Male      28 Unemployed    172      83 Drug A       452       475
6     6 wDAR  Male      33 Unemployed    178      84 Placebo      484       497
# ℹ 4 more variables: SxWheeze_Pre <fct>, SxWheeze_Post <fct>, PS_Pre <dbl>,
#   PS_Post <dbl>
```

Column Operation: Rename

- `rename()` function is used to rename column

```
asthmads_spss %>%
  rename(Ht_cm = Height) %>%
  head()
```

```
# A tibble: 6 × 18
     id idR   Gender   Age WorkStatus Ht_cm Weight_Pre WC_Pre PA_HW Weight_Post
  <dbl> <chr> <fct>  <dbl> <fct>      <dbl>      <dbl>  <dbl> <dbl>       <dbl>
1     1 ejGs  Female    34 Unemployed   179       84.2     77     3        75.5
2     2 A4pG  Male      31 Unemployed   169       81.8     94     1        75.4
3     3 qkCO  Male      25 Employed     164       88.5     95     4        80.2
4     4 jcFZ  Female    33 Unemployed   136       53.2     85     2        47.7
5     5 qVSA  Male      28 Unemployed   172       71.3     90     3        61.8
6     6 wDAR  Male      33 Unemployed   178       87.3     92     2        81.8
# ℹ 8 more variables: WC_Post <dbl>, Tx2 <fct>, PEFR_Pre <dbl>,
#   PEFR_Post <dbl>, SxWheeze_Pre <fct>, SxWheeze_Post <fct>, PS_Pre <dbl>,
#   PS_Post <dbl>
```

Column Operation: Relocate

- `relocate()` function is used to change the order of the column
  - → can be single column or multiple columns

```r
asthmads_spss %>%
  relocate(Weight_Post, .after = Weight_Pre) %>%
  head()
```

```
# A tibble: 6 × 18
     id idR   Gender   Age WorkStatus Height Weight_Pre Weight_Post WC_Pre PA_HW
  <dbl> <chr> <fct>  <dbl> <fct>       <dbl>      <dbl>       <dbl>  <dbl> <dbl>
1     1 ejGs  Female    34 Unemployed    179       84.2        75.5     77     3
2     2 A4pG  Male      31 Unemployed    169       81.8        75.4     94     1
3     3 qkCO  Male      25 Employed      164       88.5        80.2     95     4
4     4 jcFZ  Female    33 Unemployed    136       53.2        47.7     85     2
5     5 qVSA  Male      28 Unemployed    172       71.3        61.8     90     3
6     6 wDAR  Male      33 Unemployed    178       87.3        81.8     92     2
# ℹ 8 more variables: WC_Post <dbl>, Tx2 <fct>, PEFR_Pre <dbl>,
#   PEFR_Post <dbl>, SxWheeze_Pre <fct>, SxWheeze_Post <fct>, PS_Pre <dbl>,
#   PS_Post <dbl>
```

Column Operation: Relocate

- `relocate()` function is used to change the order of the column
  - → note that only one target column per line

```
asthmads_spss %>%
  relocate(Weight_Post, .after = Weight_Pre) %>%
  relocate(Height, .after = Weight_Post) %>%
  head()
```

```
# A tibble: 6 × 18
     id idR   Gender   Age WorkStatus Weight_Pre Weight_Post Height WC_Pre PA_HW
  <dbl> <chr> <fct>  <dbl> <fct>           <dbl>       <dbl>  <dbl>  <dbl> <dbl>
1     1 ejGs  Female    34 Unemployed       84.2        75.5    179     77     3
2     2 A4pG  Male      31 Unemployed       81.8        75.4    169     94     1
3     3 qkCO  Male      25 Employed         88.5        80.2    164     95     4
4     4 jcFZ  Female    33 Unemployed       53.2        47.7    136     85     2
5     5 qVSA  Male      28 Unemployed       71.3        61.8    172     90     3
6     6 wDAR  Male      33 Unemployed       87.3        81.8    178     92     2
# ℹ 8 more variables: WC_Post <dbl>, Tx2 <fct>, PEFR_Pre <dbl>,
#   PEFR_Post <dbl>, SxWheeze_Pre <fct>, SxWheeze_Post <fct>, PS_Pre <dbl>,
#   PS_Post <dbl>
```

Column Operation: Relocate

- `relocate()` function is used to change the order of the column
- Hint: if you want to move column after creating new column, you can specify the location directly from the mutate function
- rather than this

```
asthmads_spss %>%
  mutate(BMI_Pre = Weight_Pre/(Height/100)^2) %>%
  relocate(BMI_Pre, .after = Weight_Pre)
```

- you can do this

```
asthmads_spss %>%
  mutate(BMI_Pre = Weight_Pre/(Height/100)^2, .after = Weight_Pre)
```