Harmonisation Template for Cohort B

My Name

2025-03-10

Table of contents

# Preface

Here is the documentation of the data harmonisation step generated using [Quarto](https://quarto.org/). To learn more about Quarto books visit <https://quarto.org/docs/books>.

## File Structure

Here is the file structure of the project used to generate the document.

harmonisation/ # Root of the project template.
|
├── .quarto/ (not in repository) # Folder to keep intermediate files/folders
| # generated when Quarto renders the files.
|
├── archive/ # Folder to keep previous books and harmonised data.
| |
│ ├── reports/ # Folder to keep previous versions of
| | | # data harmonisation documentation.
| | |
| | ├── {some\_date}\_batch/ # Folder to keep {some\_date} version of
| | | # data harmonisation documentation.
| | |
| | └── Flowchart.xlsx # Flowchart sheet to record version control.
| |
| └── harmonised/ # Folder to keep previous version of harmonised data.
| |
| ├── {some\_date}\_batch/ # Folder to keep {some\_date} version of
| | # harmonised data.
| |
| └── Flowchart.xlsx # Flowchart sheet to record version control.
|
├── codes/ # Folder to keep R/Quarto scripts
| | # to run data harmonisation.
| |
│ ├── {cohort name}/ # Folder to keep Quarto scripts to run
| | | # data cleaning, harmonisation
| | | # and output them for each cohort.
| | |
| | └── preprocessed\_data/ # Folder to keep preprocessed data.
| |
│ ├── harmonisation\_summary/ # Folder to keep Quarto scripts to create
| | # data harmonisation summary report.
| |
│ ├── output/ # Folder to keep harmonised data.
| |
| ├── cohort\_harmonisation\_script.R # R script to render each {cohort name}/ folder.
| | # folder into html, pdf and word document.
| |
| └── harmonisation\_summary\_script.R # R script to render the {harmonisation\_summary}/
| # folder into word document.
│
├── data-raw/ # Folder to keep cohort raw data (.csv, .xlsx, etc.)
| |
│ ├── {cohort name}/ # Folder to keep cohort raw data.
| | |
| | ├── {data\_dictionary} # Data dictionary file that correspond to the
| | | # cohort raw data. Can be one from the
| | | # collaborator provide or provided by us.
| | |
| | └── Flowchart.xlsx # Flowchart sheet to record version control.
| |
| ├── data-dictionary/ # Folder to keep data dictionary
| | | # used for harmonising data.
| | |
| | └── Flowchart.xlsx # Flowchart sheet to record version control.
| |
| └── data-input/ # Folder to keep data input file
| | # for collaborators to fill in.
| |
| └── Flowchart.xlsx # Flowchart sheet to record version control.
|
├── docs/ # Folder to keep R functions documentation
| # generated using pkgdown:::build\_site\_external().
|
├── inst/ # Folder to keep arbitrary additional files
| | # to include in the project.
| |
| └── WORDLIST # File generated by spelling::update\_wordlist()
|
├── man/ # Folder to keep R functions documentation
| | # generated using devtools::document().
| |
│ ├── {fun-demo}.Rd # Documentation of the demo R function.
| |
│ └── harmonisation-template.Rd # High-level documentation.
|
├── R/ # Folder to keep R functions.
| |
│ ├── {fun-demo}.R # Script with R functions.
| |
│ └── harmonisation-package.R # Dummy R file for high-level documentation.
│
├── renv/ (not in repository) # Folder to keep all packages
| # installed in the renv environment.
|
├── reports/ # Folder to keep the most recent data harmonisation
| # documentation.
|
├── templates/ # Folder to keep template files needed to generate
| | # data harmonisation documentation efficiently.
| |
| ├── quarto-yaml/ # Folder to keep template files to generate
| | | # data harmonisation documentation structure
| | | # in Quarto.
| | |
│ | ├── \_quarto\_{cohort name}.yml # Quarto book template data harmonisation documentation
| | | # for {cohort name}.
| | |
| | └── \_quarto\_summary.yml # Quarto book template data harmonisation summary.
| |
| └── index-qmd/ # Folder to keep template files to generate
| | # the preface page of the data harmonisation
| | # documentation.
| |
| ├── \_index\_report.qmd # Preface template for each cohort data harmonisation
| | # report.
| |
| └── \_index\_summary.qmd # Preface template for data harmonisation
| # summary report.
|
├── tests/ # Folder to keep test unit files.
| # Files will be used by R package testhat.
|
├── .Rbuildignore # List of files/folders to be ignored while
│ # checking/installing the package.
|
├── .Renviron (not in repository) # File to set environment variables.
|
├── .Rprofile (not in repository) # R code to be run when R starts up.
| # It is run after the .Renviron file is sourced.
|
├── .Rhistory (not in repository) # File containing R command history.
|
├── .gitignore # List of files/folders to be ignored while
│ # using the git workflow.
|
├── .lintr # Configuration for linting
| # R projects and packages using linter.
|
├── .renvignore # List of files/folders to be ignored when
│ # renv is doing its snapshot.
|
├── DESCRIPTION[\*] # Overall metadata of the project.
|
├── LICENSE # Content of the MIT license generated via
| # usethis::use\_mit\_license().
|
├── LICENSE.md # Content of the MIT license generated via
| # usethis::use\_mit\_license().
|
├── NAMESPACE # List of functions users can use or imported
| # from other R packages. It is generated
| # by devtools::document().
│
├── README.md # GitHub README markdown file generated by Quarto.
|
├── README.qmd # GitHub README quarto file used to generate README.md.
|
├── \_pkgdown.yml # Configuration for R package documentation
| # using pkgdown:::build\_site\_external().
|
├── \_quarto.yml # Configuration for Quarto book generation.
| # It is also the project configuration file.
|
├── csl\_file.csl # Citation Style Language (CSL) file to ensure
| # citations follows the Lancet journal.
|
├── custom-reference.docx # Microsoft word template for data harmonisation
| # documentation to Word.
|
├── harmonisation\_template.Rproj # RStudio project file.
|
├── index.qmd # Preface page of Quarto book content.
|
├── references.bib # Bibtex file for Quarto book.
|
└── renv.lock # Metadata of R packages installed generated
 # using renv::snapshot().

[\*] These files are automatically created but user needs to manually add some information.

# 1. R Package And Environment

## 1.1 R Packages Used

Here are the R packages used in this analysis.

harmonisation::get\_r\_package\_info() |>
 knitr::kable()

| package | version | date | source |
| --- | --- | --- | --- |
| dplyr | 1.1.4 | 2023-11-17 | RSPM (R 4.5.0) |
| fontawesome | 0.5.3 | 2024-11-16 | RSPM (R 4.5.0) |
| forcats | 1.0.0 | 2023-01-29 | RSPM (R 4.5.0) |
| glue | 1.8.0 | 2024-09-30 | RSPM (R 4.5.0) |
| harmonisation | 1.0.0.0 | 2025-05-20 | local |
| here | 1.0.1 | 2020-12-13 | RSPM (R 4.5.0) |
| htmltools | 0.5.8.1 | 2024-04-04 | RSPM (R 4.5.0) |
| lubridate | 1.9.4 | 2024-12-08 | RSPM (R 4.5.0) |
| magrittr | 2.0.3 | 2022-03-30 | RSPM (R 4.5.0) |
| openxlsx | 4.2.8 | 2025-01-25 | RSPM (R 4.5.0) |
| pointblank | 0.12.2 | 2024-10-23 | RSPM (R 4.5.0) |
| purrr | 1.0.4 | 2025-02-05 | RSPM (R 4.5.0) |
| quarto | 1.4.4 | 2024-07-20 | RSPM (R 4.5.0) |
| reactable | 0.4.4 | 2023-03-12 | RSPM (R 4.5.0) |
| sessioninfo | 1.2.2 | 2021-12-06 | CRAN (R 4.5.0) |
| stringr | 1.5.1 | 2023-11-14 | RSPM (R 4.5.0) |
| testthat | 3.2.3 | 2025-01-13 | RSPM (R 4.5.0) |
| tibble | 3.2.1 | 2023-03-20 | RSPM (R 4.5.0) |
| tidyr | 1.3.1 | 2024-01-24 | RSPM (R 4.5.0) |
| vroom | 1.6.5 | 2023-12-05 | RSPM (R 4.5.0) |

## 1.2 R Platform Information

Here are the R platform environment used in this analysis.

harmonisation::get\_r\_platform\_info() |>
 knitr::kable()

| setting | value |
| --- | --- |
| version | R version 4.5.0 (2025-04-11 ucrt) |
| os | Windows 11 x64 (build 26100) |
| system | x86\_64, mingw32 |
| ui | RTerm |
| language | (EN) |
| collate | English\_Singapore.utf8 |
| ctype | English\_Singapore.utf8 |
| tz | Asia/Singapore |
| date | 2025-06-11 |
| pandoc | 3.4 @ C:/Program Files/RStudio/resources/app/bin/quarto/bin/tools/ (via rmarkdown) |
| quarto | 1.7.30 @ C:/Program Files/Quarto/bin/quarto.exe/ (via quarto) |
| knitr | 1.49 from RSPM (R 4.5.0) |

## 1.3 Data dictionary

Check to see if the data dictionary 20250310\_data\_dictionary.xlsx exists.

dict\_relative\_path <- fs::path(
 "data-raw",
 "data\_dictionary",
 params$data\_dictionary
)

dict\_path <- here::here(dict\_relative\_path)

if (!file.exists(dict\_path)) {
 stop(glue::glue("Input data dictionary {dict\_path} cannot be found"))
}

# 2. Read Cohort B Data

## 2.1 Read Data

We read the file data\_to\_harmonise\_age\_issue.csv using [vroom::vroom](https://vroom.r-lib.org/reference/vroom.html)

cohort\_B\_data <- vroom::vroom(
 file = here::here("data-raw",
 "Cohort\_B",
 "data\_to\_harmonise\_age\_issue.csv"),
 delim = ",",
 col\_select = 1:2,
 show\_col\_types = FALSE,
 col\_types = list(
 ID = vroom::col\_character(),
 Age = vroom::col\_integer()
 )
 ) |>
 dplyr::rename(cohort\_unique\_id = "ID") |>
 # Remove rows when the ID value is NA
 dplyr::filter(!is.na(.data[["cohort\_unique\_id"]])) |>
 # Remove white spaces in column names
 dplyr::rename\_all(stringr::str\_trim) |>
 # Check if cohort id is unique
 pointblank::rows\_distinct(
 columns = "cohort\_unique\_id",
 )

To safeguard a csv file with issues, we can use the function [vroom::problems](https://vroom.r-lib.org/reference/problems.html)

If there are issues with the data, the output of [vroom::problems](https://vroom.r-lib.org/reference/problems.html) will be a [tibble](https://tibble.tidyverse.org/).

cohort\_B\_data |>
 vroom::problems()

# A tibble: 3 × 5
 row col expected actual file
 <int> <int> <chr> <chr> <chr>
1 4 2 an integer missing D:/Jeremy/PortableR/RPortableWorkDirectory/har…
2 10 2 an integer missing D:/Jeremy/PortableR/RPortableWorkDirectory/har…
3 17 2 an integer missing D:/Jeremy/PortableR/RPortableWorkDirectory/har…

To check for this in an automatically, we can use [pointblank::expect\_row\_count\_match](https://rstudio.github.io/pointblank/reference/row_count_match.html)

cohort\_B\_data |>
 vroom::problems() |>
 pointblank::expect\_row\_count\_match(count = 0)

Error: Row counts for the two tables did not match.
The `expect\_row\_count\_match()` validation failed beyond the absolute threshold level (1).
\* failure level (1) >= failure threshold (1)

Suppose we have a csv file with no issues, we can safeguard it with the following code.

cohort\_B\_data <- vroom::vroom(
 file = here::here("data-raw",
 "Cohort\_B",
 "data\_to\_harmonise.csv"),
 delim = ",",
 col\_select = 1:8,
 show\_col\_types = FALSE,
 col\_types = list(
 ID = vroom::col\_character(),
 Age = vroom::col\_integer(),
 Sex = vroom::col\_character(),
 Height = vroom::col\_double(),
 Weight = vroom::col\_double(),
 `Smoke History` = vroom::col\_character(),
 `Chest Pain Character` = vroom::col\_character(),
 Dyspnea = vroom::col\_character()
 )
 ) |>
 dplyr::rename(cohort\_unique\_id = "ID") |>
 # Remove rows when the ID value is NA
 dplyr::filter(!is.na(.data[["cohort\_unique\_id"]])) |>
 # Remove white spaces in column names
 dplyr::rename\_all(stringr::str\_trim) |>
 # Check if cohort id is unique
 pointblank::rows\_distinct(
 columns = "cohort\_unique\_id",
 )

cohort\_B\_data |>
 vroom::problems() |>
 pointblank::expect\_row\_count\_match(count = 0)

## 2.2 Write Preprocessed File

We output data to be used for the next session.

cohort\_B\_data |>
 fst::write\_fst(
 path = here::here(params$analysis\_folder,
 params$harmonisation\_folder,
 params$preprocessing\_folder,
 "01\_Cohort\_B\_cleaned.fst")
)

# 3. Extract Demographic

## 3.1 Read Preprocessed File

We read output data from the previous section.

## 3.2 Demographics and Behavioral parameters

### 3.2.1 Age and Sex

age\_years will be mapped from the column Age.

sex is grouped as follows:

| Sex | sex |
| --- | --- |
| Female | 0 |
| Male | 1 |

age\_gender\_data <- cohort\_B\_data |>
 dplyr::select(c("cohort\_unique\_id",
 "Age",
 "Sex")) |>
 pointblank::col\_vals\_expr(
 expr = ~ harmonisation::is\_integer\_vector(
 cohort\_A\_data[["age"]],
 allow\_na = TRUE)
 ) |>
 dplyr::mutate(
 # Convert age to type integer
 age\_years = as.integer(.data[["Age"]]),
 # Convert categorical columns to factors
 sex = dplyr::case\_when(
 .data[["Sex"]] == "Female" ~ "0",
 .data[["Sex"]] == "Male" ~ "1",
 .default = NA\_character\_
 ),
 `Sex` = forcats::fct\_relevel(
 .data[["Sex"]],
 c("Female", "Male")
 ),
 sex = forcats::fct\_relevel(
 .data[["sex"]],
 c("0", "1")),
 ) |>
 dplyr::relocate(
 "sex",
 .after = "Sex"
 ) |>
 dplyr::relocate(
 "age\_years",
 .after = "Age"
 ) |>
 pointblank::col\_vals\_in\_set(
 columns = "sex",
 set = c("0", "1")
 ) |>
 pointblank::col\_vals\_between(
 columns = "age\_years",
 left = 0,
 right = 100,
 inclusive = c(FALSE, TRUE),
 na\_pass = TRUE
 )

if (params$show\_table) {
 age\_gender\_data |>
 dplyr::distinct(.data[["Sex"]],
 .keep\_all = TRUE) |>
 knitr::kable()
}

| cohort\_unique\_id | Age | age\_years | Sex | sex |
| --- | --- | --- | --- | --- |
| B001 | 32 | 32 | Female | 0 |
| B003 | 80 | 80 | Male | 1 |

Remove unnecessary columns so that we can merge with the other fields.

age\_gender\_data <- age\_gender\_data |>
 dplyr::select(-c("Age", "Sex"))

### 3.2.2 Height, Weight, BMI and BSA

height\_cm will be mapped from the column Height. weight\_kg will be mapped from the column Weight.

bsa\_m2 in m^2 will be calculated as sqrt([Height(cm) x Weight(kg)]/3600) bmi will be calculated as Weight(kg)/((Height(m))^2)

All values are then converted to two decimal places.

body\_measurement\_data <- cohort\_B\_data |>
 dplyr::select(c("cohort\_unique\_id",
 "Height", "Weight")) |>
 dplyr::mutate(
 height\_cm = .data[["Height"]],
 weight\_kg = .data[["Weight"]],
 bsa\_m2 = sqrt((.data[["height\_cm"]] \* .data[["weight\_kg"]]) / 3600),
 bsa\_m2 = harmonisation::round\_to\_nearest\_digit(.data[["bsa\_m2"]], digits = 2),
 bmi = .data[["weight\_kg"]] / ((.data[["height\_cm"]] / 100)^2),
 bmi = harmonisation::round\_to\_nearest\_digit(.data[["bmi"]], digits = 2),
 height\_cm = harmonisation::round\_to\_nearest\_digit(.data[["height\_cm"]], digits = 2),
 weight\_kg = harmonisation::round\_to\_nearest\_digit(.data[["weight\_kg"]], digits = 2)
 ) |>
 pointblank::col\_vals\_between(
 columns = "bmi",
 left = 10,
 right = 50,
 inclusive = c(TRUE, TRUE),
 na\_pass = TRUE
 )

if (params$show\_table) {
 body\_measurement\_data |>
 head(n = 5) |>
 knitr::kable()
}

| cohort\_unique\_id | Height | Weight | height\_cm | weight\_kg | bsa\_m2 | bmi |
| --- | --- | --- | --- | --- | --- | --- |
| B001 | 170 | 63 | 170 | 63 | 1.72 | 21.80 |
| B002 | 167 | 71 | 167 | 71 | 1.81 | 25.46 |
| B003 | 184 | 77 | 184 | 77 | 1.98 | 22.74 |
| B004 | 160 | 83 | 160 | 83 | 1.92 | 32.42 |
| B005 | 155 | 61 | 155 | 61 | 1.62 | 25.39 |

Remove unnecessary columns so that we can merge with the other fields.

body\_measurement\_data <- body\_measurement\_data |>
 dplyr::select(-c("Height", "Weight"))

### 3.2.3 Smoking History

smoke\_current is grouped as follows:

| Smoke History | smoke\_current |
| --- | --- |
| non-smoker | 0 |
| past smoker | 0 |
| current smoker | 1 |
| NA | -1 |

smoke\_past is grouped as follows:

| Smoke History | smoke\_past |
| --- | --- |
| non-smoker | 0 |
| past smoker | 1 |
| current smoker | 0 |
| NA | -1 |

We do a check to ensure that we can only have these scenarios

* smoke\_current as 1 and smoke\_past as 0 for current smokers
* smoke\_current as 0 and smoke\_past as 1 for past smokers
* smoke\_current as 0 and smoke\_past as 0 for non-smokers
* smoke\_current as -1 and smoke\_past as -1 for unknown

smoking\_data <- cohort\_B\_data |>
 dplyr::select(c("cohort\_unique\_id",
 "Smoke History")) |>
 dplyr::mutate(
 smoke\_current = dplyr::case\_when(
 is.na(.data[["Smoke History"]]) ~ "-1",
 .data[["Smoke History"]] == "non-smoker" ~ "0",
 .data[["Smoke History"]] == "past smoker" ~ "0",
 .data[["Smoke History"]] == "current smoker" ~ "1",
 .default = NA\_character\_
 ),
 smoke\_current = forcats::fct\_relevel(
 .data[["smoke\_current"]],
 c("0", "1")),
 smoke\_past = dplyr::case\_when(
 is.na(.data[["Smoke History"]]) ~ "-1",
 .data[["Smoke History"]] == "non-smoker" ~ "0",
 .data[["Smoke History"]] == "past smoker" ~ "1",
 .data[["Smoke History"]] == "current smoker" ~ "0",
 .default = NA\_character\_
 ),
 smoke\_past = forcats::fct\_relevel(
 .data[["smoke\_past"]],
 c("0", "1")),
 `Smoke History` = forcats::fct(
 .data[["Smoke History"]]
 )
 ) |>
 pointblank::col\_vals\_in\_set(
 columns = c("smoke\_current", "smoke\_past"),
 set = c("0", "1", "-1")
 ) |>
 pointblank::col\_vals\_expr(
 expr = pointblank::expr(
 (.data[["smoke\_current"]] == "1" & .data[["smoke\_past"]] == "0") |
 (.data[["smoke\_current"]] == "-1" & .data[["smoke\_past"]] == -"1") |
 (.data[["smoke\_current"]] == "0" & .data[["smoke\_past"]] %in% c("0", "1"))
 )
 )

if (params$show\_table) {
 smoking\_data |>
 dplyr::distinct(.data[["Smoke History"]],
 .keep\_all = TRUE) |>
 knitr::kable()
}

| cohort\_unique\_id | Smoke History | smoke\_current | smoke\_past |
| --- | --- | --- | --- |
| B001 | non-smoker | 0 | 0 |
| B002 | current smoker | 1 | 0 |
| B004 | past smoker | 0 | 1 |
| B017 | NA | -1 | -1 |

Remove unnecessary columns so that we can merge with the other fields.

smoking\_data <- smoking\_data |>
 dplyr::select(-c("Smoke History"))

### 3.2.4 Chest Pain

#### 3.2.4.1 Shortness of Breath

have\_sob is grouped as follows:

| Dyspnea | have\_sob |
| --- | --- |
| no | 0 |
| yes | 1 |

shortness\_of\_breath\_data <- cohort\_B\_data |>
 dplyr::select(c("cohort\_unique\_id", "Dyspnea")) |>
 dplyr::mutate(
 have\_sob = dplyr::case\_when(
 .data[["Dyspnea"]] == "no" ~ "0",
 .data[["Dyspnea"]] == "yes" ~ "1",
 .default = NA\_character\_
 ),
 have\_sob = forcats::fct\_relevel(
 as.character(.data[["have\_sob"]]),
 c("0", "1")),
 Dyspnea = forcats::fct\_relevel(
 as.character(.data[["Dyspnea"]]),
 c("no", "yes")),
 ) |>
 pointblank::col\_vals\_in\_set(
 columns = c("have\_sob"),
 set = c("0", "1", "-1")
 )

if (params$show\_table) {
 shortness\_of\_breath\_data |>
 dplyr::distinct(.data[["Dyspnea"]],
 .keep\_all = TRUE) |>
 knitr::kable()
}

| cohort\_unique\_id | Dyspnea | have\_sob |
| --- | --- | --- |
| B001 | no | 0 |
| B002 | yes | 1 |

Remove unnecessary columns so that we can merge with the other fields.

shortness\_of\_breath\_data <- shortness\_of\_breath\_data |>
 dplyr::select(-c("Dyspnea"))

#### 3.2.4.2 Have chest pain or not

have\_chest\_pain is grouped as follows:

| Chest Pain Character | have\_chest\_pain |
| --- | --- |
| no chest pain | 0 |
| typical, atypical or nonanginal | 1 |

have\_chest\_pain\_data <- cohort\_B\_data |>
 dplyr::select(c("cohort\_unique\_id", "Chest Pain Character")) |>
 dplyr::mutate(
 have\_chest\_pain = dplyr::case\_when(
 .data[["Chest Pain Character"]] %in% c("no chest pain") ~ "0",
 .data[["Chest Pain Character"]] %in% c("typical", "atypical", "nonanginal") ~ "1",
 .default = NA\_character\_
 ),
 have\_chest\_pain = forcats::fct\_relevel(
 .data[["have\_chest\_pain"]],
 c("0", "1")
 ),
 `Chest Pain Character` = forcats::fct\_relevel(
 as.character(.data[["Chest Pain Character"]]),
 c("no chest pain", "typical", "atypical", "nonanginal")
 )
 ) |>
 pointblank::col\_vals\_in\_set(
 columns = c("have\_chest\_pain"),
 set = c("0", "1")
 )

if (params$show\_table) {
 have\_chest\_pain\_data |>
 dplyr::distinct(.data[["Chest Pain Character"]],
 .keep\_all = TRUE) |>
 knitr::kable()
}

| cohort\_unique\_id | Chest Pain Character | have\_chest\_pain |
| --- | --- | --- |
| B001 | atypical | 1 |
| B002 | no chest pain | 0 |
| B005 | typical | 1 |
| B006 | nonanginal | 1 |

Remove unnecessary columns so that we can merge with the other fields.

have\_chest\_pain\_data <- have\_chest\_pain\_data |>
 dplyr::select(-c("Chest Pain Character"))

#### 3.2.4.3 Symptomatic or Asymptomatic

symptoms is grouped as follows:

| have\_sob | have\_chest\_pain | symptoms |
| --- | --- | --- |
| -1 | -1 | -1 |
| 0 | 0 | 0 |
| 0 or 1 | 1 | 1 |
| 1 | 0 | 2 |

symptoms\_data <- cohort\_B\_data |>
 dplyr::select(c("cohort\_unique\_id")) |>
 dplyr::inner\_join(shortness\_of\_breath\_data,
 by = dplyr::join\_by("cohort\_unique\_id"),
 unmatched = "error",
 relationship = "one-to-one") |>
 dplyr::inner\_join(have\_chest\_pain\_data,
 by = dplyr::join\_by("cohort\_unique\_id"),
 unmatched = "error",
 relationship = "one-to-one") |>
 dplyr::mutate(
 symptoms = dplyr::case\_when(
 (.data[["have\_chest\_pain"]] == "-1" &
 .data[["have\_sob"]] == "-1"
 ) ~ "-1",
 (.data[["have\_chest\_pain"]] == "0" &
 .data[["have\_sob"]] == "0"
 ) ~ "0",
 (.data[["have\_chest\_pain"]] == "1" &
 .data[["have\_sob"]] %in% c("0", "1")
 ) ~ "1",
 (.data[["have\_chest\_pain"]] == "0" &
 .data[["have\_sob"]] == "1"
 ) ~ "2",
 .default = NA\_character\_
 ),
 symptoms = forcats::fct\_relevel(
 .data[["symptoms"]],
 c("0", "1", "2"))
 ) |>
 pointblank::col\_vals\_in\_set(
 columns = c("symptoms"),
 set = c("0", "1", "2")
 )

if (params$show\_table) {
 symptoms\_data |>
 dplyr::distinct(.data[["have\_chest\_pain"]], .data[["have\_sob"]],
 .keep\_all = TRUE) |>
 knitr::kable()
}

| cohort\_unique\_id | have\_sob | have\_chest\_pain | symptoms |
| --- | --- | --- | --- |
| B001 | 0 | 1 | 1 |
| B002 | 1 | 0 | 2 |
| B003 | 1 | 1 | 1 |
| B007 | 0 | 0 | 0 |

Remove unnecessary columns so that we can merge with the other fields.

symptoms\_data <- symptoms\_data |>
 dplyr::select(-c("have\_chest\_pain", "have\_sob"))

#### 3.2.4.4 Chest Pain Type

chest\_pain\_type is grouped as follows:

| Dyspnea | Chest Pain Character | chest\_pain\_type |
| --- | --- | --- |
| no | no chest pain | 0 |
| no or yes | typical | 1 |
| no or yes | atypical | 2 |
| no or yes | nonanginal | 3 |
| yes | no chest pain | 4 |

chest\_pain\_type\_data <- cohort\_B\_data |>
 dplyr::select(c("cohort\_unique\_id", "Chest Pain Character", "Dyspnea")) |>
 dplyr::mutate(
 chest\_pain\_type = dplyr::case\_when(
 (.data[["Chest Pain Character"]] == "no chest pain" &
 .data[["Dyspnea"]] == "no"
 ) ~ "0",
 (.data[["Chest Pain Character"]] == "typical" &
 .data[["Dyspnea"]] %in% c("no", "yes")
 ) ~ "1",
 (.data[["Chest Pain Character"]] == "atypical" &
 .data[["Dyspnea"]] %in% c("no", "yes")
 ) ~ "2",
 (.data[["Chest Pain Character"]] == "nonanginal" &
 .data[["Dyspnea"]] %in% c("no", "yes")
 ) ~ "3",
 (.data[["Chest Pain Character"]] == "no chest pain" &
 .data[["Dyspnea"]] == "yes"
 ) ~ "4",
 .default = NA\_character\_
 ),
 `Chest Pain Character` = forcats::fct\_relevel(
 as.character(.data[["Chest Pain Character"]]),
 c("no chest pain", "typical", "atypical", "nonanginal")
 ),
 `Dyspnea` = forcats::fct\_relevel(
 as.character(.data[["Dyspnea"]]),
 c("no", "yes")
 ),
 chest\_pain\_type = forcats::fct\_relevel(
 .data[["chest\_pain\_type"]],
 c("0", "1", "2", "3"))
 ) |>
 dplyr::relocate(
 "Chest Pain Character",
 .after = "cohort\_unique\_id"
 ) |>
 pointblank::col\_vals\_in\_set(
 columns = c("chest\_pain\_type"),
 set = c("0", "1", "2", "3", "4")
 )

if (params$show\_table) {
 chest\_pain\_type\_data |>
 dplyr::distinct(.data[["Dyspnea"]], .data[["Chest Pain Character"]],
 .keep\_all = TRUE) |>
 knitr::kable()
}

| cohort\_unique\_id | Chest Pain Character | Dyspnea | chest\_pain\_type |
| --- | --- | --- | --- |
| B001 | atypical | no | 2 |
| B002 | no chest pain | yes | 4 |
| B003 | atypical | yes | 2 |
| B005 | typical | no | 1 |
| B006 | nonanginal | yes | 3 |
| B007 | no chest pain | no | 0 |
| B013 | nonanginal | no | 3 |
| B015 | typical | yes | 1 |

Remove unnecessary columns so that we can merge with the other fields.

chest\_pain\_type\_data <- chest\_pain\_type\_data |>
 dplyr::select(-c("Dyspnea", "Chest Pain Character"))

#### 3.2.4.5 Combined chest pain related tables

We combine all chest related tables together

join\_specification <- dplyr::join\_by("cohort\_unique\_id")

chest\_pain\_data <- cohort\_B\_data |>
 dplyr::select(c("cohort\_unique\_id")) |>
 dplyr::inner\_join(have\_chest\_pain\_data,
 by = join\_specification,
 unmatched = "error",
 relationship = "one-to-one") |>
 dplyr::inner\_join(chest\_pain\_type\_data,
 by = join\_specification,
 unmatched = "error",
 relationship = "one-to-one") |>
 dplyr::inner\_join(shortness\_of\_breath\_data,
 by = join\_specification,
 unmatched = "error",
 relationship = "one-to-one") |>
 dplyr::inner\_join(symptoms\_data,
 by = join\_specification,
 unmatched = "error",
 relationship = "one-to-one")

testthat::expect\_true(
 pointblank::has\_columns(
 chest\_pain\_data,
 columns = c("have\_sob", "have\_chest\_pain", "symptoms", "chest\_pain\_type")
 )
)

testthat::expect\_equal(
 ncol(chest\_pain\_data), 5
)

### 3.2.5 Combine Demographics

We combine all the data to give the demo\_behave\_data.

join\_specification <- dplyr::join\_by("cohort\_unique\_id")

demo\_behave\_data <- cohort\_B\_data |>
 dplyr::select(c("cohort\_unique\_id")) |>
 dplyr::inner\_join(age\_gender\_data,
 by = join\_specification,
 unmatched = "error",
 relationship = "one-to-one") |>
 dplyr::inner\_join(body\_measurement\_data,
 by = join\_specification,
 unmatched = "error",
 relationship = "one-to-one") |>
 dplyr::inner\_join(smoking\_data,
 by = join\_specification,
 unmatched = "error",
 relationship = "one-to-one") |>
 dplyr::inner\_join(chest\_pain\_data,
 by = join\_specification,
 unmatched = "error",
 relationship = "one-to-one") |>
 dplyr::relocate(c("bsa\_m2", "bmi"),
 .after = "sex")

testthat::expect\_true(
 pointblank::has\_columns(
 demo\_behave\_data,
 columns = c(
 "age\_years", "sex",
 "height\_cm", "weight\_kg", "bsa\_m2", "bmi",
 "smoke\_current", "smoke\_past",
 "have\_sob", "have\_chest\_pain",
 "symptoms", "chest\_pain\_type"
 )
 )
)

testthat::expect\_equal(
 ncol(demo\_behave\_data), 13
)

## 3.3 Write Preprocessed File

We output data to be used for the next session.

demo\_behave\_data |>
 fst::write\_fst(
 path = here::here(
 params$analysis\_folder,
 params$harmonisation\_folder,
 params$preprocessing\_folder,
 "02\_demographic\_data.fst"),
)

# 4. Export To Excel

out\_type <- knitr::opts\_chunk$get("rmarkdown.pandoc.to")

## 4.1 Read all tabular data

We read all tabular data from the previous section.

## 4.2 Export Data as Excel

We export the standardised data to an excel file called harmonised\_Cohort\_B.xlsx

# Create a new workbook
my\_workbook <- openxlsx::createWorkbook()

sheet\_name = c("demographics")

output\_data = list(demo\_behave\_data) |>
 purrr::map(
 .f = harmonisation::add\_cohort\_name,
 cohort\_name = params$cohort\_name,
 cohort\_name\_column = "cohort\_name"
 )

purrr::walk2(
 .x = sheet\_name,
 .y = output\_data,
 .f = harmonisation::write\_to\_sheet,
 workbook = my\_workbook
)

# Save workbook
openxlsx::saveWorkbook(
 wb = my\_workbook,
 file = here::here(params$analysis\_folder,
 params$output\_folder,
 params$cleaned\_folder,
 params$output\_excel\_file),
 overwrite = TRUE
)