Harmonisation Template for Cohort A

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) |
| 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) |
| readxl | 1.4.4 | 2025-02-27 | 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) |

## 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 A Data

## 2.1 Read Data

We read the data and have the following warnings

cohort\_A\_data <- readxl::read\_excel(
 path = here::here("data-raw",
 "Cohort\_A",
 "data\_to\_harmonise\_age\_issue.xlsx"),
 sheet = "Sheet1",
 col\_types = c(
 "text", "numeric"
 )
 )

This warning occurs because we expect the second column Age to be numeric but there exists some text columns.

Suppose we ask the collaborator to fix the age column and the collaborator returns a new file. To ensure that there are no messages, we can use [testthat::expect\_no\_condition](https://testthat.r-lib.org/reference/expect_error.html).

Here is an example when it gives an error with the old file

testthat::expect\_no\_condition(
 readxl::read\_excel(
 path = here::here("data-raw",
 "Cohort\_A",
 "data\_to\_harmonise\_age\_issue.xlsx"),
 sheet = "Sheet1",
 col\_types = c(
 "text", "numeric"
 )
 )
)

Error: Expected `readxl::read\_excel(...)` to run without any conditions.
ℹ Actually got a <simpleWarning> with text:
 Expecting numeric in B7 / R7C2: got 'missing'

We can read the new file in the following way. However, this method means that you will need to read the file twice.

testthat::expect\_no\_condition(
 readxl::read\_excel(
 path = here::here("data-raw",
 "Cohort\_A",
 "data\_to\_harmonise\_age\_issue\_fixed.xlsx"),
 sheet = "Sheet1",
 col\_types = c(
 "text", "numeric"
 )
 )
)

cohort\_A\_data <- readxl::read\_excel(
 path = here::here("data-raw",
 "Cohort\_A",
 "data\_to\_harmonise\_age\_issue\_fixed.xlsx"),
 sheet = "Sheet1",
 col\_types = c(
 "text", "numeric"
 )
 )

To read the file only once, we can use the tee pipe operator [%T>%](https://magrittr.tidyverse.org/reference/tee.html).

cohort\_A\_data <- readxl::read\_excel(
 path = here::here("data-raw",
 "Cohort\_A",
 "data\_to\_harmonise\_age\_issue\_fixed.xlsx"),
 sheet = "Sheet1",
 col\_types = c(
 "text", "numeric"
 )
 ) %T>%
 testthat::expect\_no\_condition()

## 2.2 Check for unique patient id

We can use [pointblank::rows\_distinct](https://rstudio.github.io/pointblank/reference/rows_distinct.html) to check if the column Serial Number has unique values.

cohort\_A\_data <- readxl::read\_excel(
 path = here::here("data-raw",
 "Cohort\_A",
 "data\_to\_harmonise\_age\_issue\_fixed.xlsx"),
 sheet = "Sheet1",
 col\_types = c(
 "text", "numeric"
 )
 ) %T>%
 testthat::expect\_no\_condition() |>
 dplyr::rename(cohort\_unique\_id = "Serial Number") |>
 # Remove rows when the ID value is NA
 dplyr::filter(!is.na(.data[["cohort\_unique\_id"]])) |>
 dplyr::mutate(
 cohort\_unique\_id = as.character(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",
 )

## 2.3 Clean Weight columns

Sometimes the collaborator will not give you a new file and will only respond with an email acknowledging that it is an error.

You will need to edit the values yourself. It is best not to edit the file as you may forget to make the manual change if the collaborator gives you a new version a few months later with the same error.

It is also advised to record such changes before data harmonisation.

We read the data with the some issues with the weight.

cohort\_A\_data <- readxl::read\_excel(
 path = here::here("data-raw",
 "Cohort\_A",
 "data\_to\_harmonise.xlsx"),
 sheet = "Sheet1",
 col\_types = c(
 "text", # unique id
 "numeric", "text", # age and sex
 "numeric", "numeric", # height and weight
 "numeric", "numeric", "numeric", "numeric", # smoking history
 "numeric", "numeric" # symptoms
 )
 ) %T>%
 testthat::expect\_no\_condition() |>
 dplyr::rename(cohort\_unique\_id = "Serial Number") |>
 # Remove rows when the ID value is NA
 dplyr::filter(!is.na(.data[["cohort\_unique\_id"]])) |>
 dplyr::mutate(
 cohort\_unique\_id = as.character(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",
 )

## 2.4 Update Weight

Here are the following patient’s height that needs to be updated.

* A018 has a weight of 215.4kg. Value is changed to 90 kg.

weight\_data <- cohort\_A\_data |>
 dplyr::select(c("cohort\_unique\_id", "weight")) |>
 # Check if these patient IDs are present
 pointblank::col\_vals\_make\_subset(
 columns = c("cohort\_unique\_id"),
 set = c("A018")
 ) |>
 dplyr::mutate(
 updated\_weight = dplyr::case\_when(
 .data[["cohort\_unique\_id"]] == "A018" & .data[["weight"]] == 215.4 ~ 90.1,
 .default = .data[["weight"]]
 ),
 )

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

weight\_data <- weight\_data |>
 dplyr::select(-c("weight"))

## 2.5 Merge updated weight

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

cohort\_A\_data <- cohort\_A\_data |>
 dplyr::inner\_join(weight\_data,
 by = join\_specification,
 unmatched = "error",
 relationship = "one-to-one") |>
 dplyr::mutate(
 `weight` = .data[["updated\_weight"]]
 ) |>
 dplyr::select(-c("updated\_weight"))

## 2.6 Check corrections

We check if the corrections are made based on the collaborator request. Changes are made manually on the excel file as the collaborator is no longer providing newer version of the data.

* weight changed from 215.4kg to 90.1kg for patient A018

cohort\_A\_data |>
 # Check if these patient IDs are present
 pointblank::expect\_col\_vals\_make\_subset(
 columns = c("cohort\_unique\_id"),
 set = c("A018")
 ) |>
 pointblank::expect\_col\_vals\_expr(
 expr = pointblank::expr(
 dplyr::case\_when(
 .data[["cohort\_unique\_id"]] %in% "A018" ~
 isTRUE(all.equal(
 target = 90.1,
 current = cohort\_A\_data[["weight"]][which(cohort\_A\_data[["cohort\_unique\_id"]] == "A018")],
 tolerance = 0.0001)
 ))
 )
 )

## 2.7 Write Preprocessed File

We output data to be used for the next session.

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

# 3. Extract Demographic

## 3.1 Read Preprocessed File

We read output data from the previous section.

## 3.2 Check for integer vector

We have a function that checks if the numeric vector has integers.

non\_integer\_data <- data.frame(
 non\_integer\_col = c(-1, 0, NA, 2.0000,
 3.010, pi, exp(1)
 )
)

non\_integer\_data |>
 pointblank::col\_vals\_expr(
 expr = ~ harmonisation::is\_integer\_vector(
 input\_vector = non\_integer\_data[["non\_integer\_col"]],
 allow\_na = TRUE)
 )

Error: The `col\_vals\_expr()` validation failed beyond the absolute threshold level (1).
\* failure level (3) >= failure threshold (1)

## 3.3 Demographics and Behavioral parameters

### 3.3.1 Age and Sex

age\_years will be mapped from the column age. age value of 0 is set as missing.

sex is grouped as follows:

| sex before | sex |
| --- | --- |
| F | 0 |
| M | 1 |

age\_gender\_data <- cohort\_A\_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 age of 0 to NA
 age\_years = dplyr::case\_when(
 .data[["age\_years"]] == 0 ~ NA\_integer\_,
 .default = .data[["age\_years"]]
 ),
 sex\_before = .data[["sex"]],
 # Convert categorical columns to factors
 sex = dplyr::case\_when(
 .data[["sex\_before"]] == "F" ~ "0",
 .data[["sex\_before"]] == "M" ~ "1",
 .default = as.character(.data[["sex\_before"]])
 ),
 `sex\_before` = forcats::fct\_relevel(
 .data[["sex\_before"]],
 c("F", "M")
 ),
 sex = forcats::fct\_relevel(
 .data[["sex"]],
 c("0", "1")),
 ) |>
 dplyr::relocate(
 "sex",
 .after = "sex\_before"
 ) |>
 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\_before"]],
 .keep\_all = TRUE) |>
 knitr::kable()
}

| cohort\_unique\_id | age | age\_years | sex\_before | sex |
| --- | --- | --- | --- | --- |
| A001 | 26 | 26 | M | 1 |
| A002 | 55 | 55 | F | 0 |

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

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

### 3.3.2 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.

To date, only patient A010 has a bmi greater than 50.

body\_measurement\_data <- cohort\_A\_data |>
 dplyr::select(c("cohort\_unique\_id",
 "weight", "height")) |>
 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\_gt(
 columns = "bmi",
 preconditions = ~ . %>%
 dplyr::filter(
 .data[["cohort\_unique\_id"]] %in% c("A010")
 ),
 value = 50,
 na\_pass = TRUE
 ) |>
 pointblank::col\_vals\_between(
 columns = "bmi",
 preconditions = ~ . %>%
 dplyr::filter(
 !.data[["cohort\_unique\_id"]] %in% c("A010")
 ),
 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 | weight | height | height\_cm | weight\_kg | bsa\_m2 | bmi |
| --- | --- | --- | --- | --- | --- | --- |
| A001 | 75.8 | 177.8 | 177.8 | 75.8 | 1.93 | 23.98 |
| A002 | 57.2 | 166.2 | 166.2 | 57.2 | 1.63 | 20.71 |
| A003 | 57.6 | 169.1 | 169.1 | 57.6 | 1.64 | 20.14 |
| A004 | 129.3 | 186.4 | 186.4 | 129.3 | 2.59 | 37.21 |
| A005 | 86.1 | 186.0 | 186.0 | 86.1 | 2.11 | 24.89 |

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

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

### 3.3.3 Smoking History

smoke\_current will be mapped from the column smoke\_current\_good. smoke\_past will be mapped from the column smoke\_past\_good.

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\_A\_data |>
 dplyr::select(c("cohort\_unique\_id",
 "smoke\_current\_good", "smoke\_past\_good")) |>
 dplyr::mutate(
 smoke\_current = as.character(.data[["smoke\_current\_good"]]),
 smoke\_current\_good = forcats::fct\_relevel(
 as.character(.data[["smoke\_current\_good"]]),
 c("0", "1")),
 smoke\_current = forcats::fct\_relevel(
 .data[["smoke\_current"]],
 c("0", "1")),
 smoke\_past = as.character(.data[["smoke\_past\_good"]]),
 smoke\_past\_good = forcats::fct\_relevel(
 as.character(.data[["smoke\_past\_good"]]),
 c("0", "1")),
 smoke\_past = forcats::fct\_relevel(
 .data[["smoke\_past"]],
 c("0", "1")),
 ) |>
 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\_current\_good"]],
 .data[["smoke\_past\_good"]],
 .keep\_all = TRUE) |>
 knitr::kable()
}

| cohort\_unique\_id | smoke\_current\_good | smoke\_past\_good | smoke\_current | smoke\_past |
| --- | --- | --- | --- | --- |
| A001 | 0 | 0 | 0 | 0 |
| A002 | 1 | 0 | 1 | 0 |
| A004 | 0 | 1 | 0 | 1 |
| A010 | 1 | 1 | 1 | 1 |
| A016 | -1 | -1 | -1 | -1 |

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

smoking\_data <- smoking\_data |>
 dplyr::select(-c("smoke\_current\_good", "smoke\_past\_good"))

Here is a case when the validation has failed.

smoking\_data\_bad <- cohort\_A\_data |>
 dplyr::select(c("cohort\_unique\_id",
 "smoke\_current\_bad", "smoke\_past\_bad")) |>
 dplyr::filter(
 .data[["cohort\_unique\_id"]] %in% c("A010", "A016")
 ) |>
 dplyr::mutate(
 smoke\_current = as.character(.data[["smoke\_current\_bad"]]),
 smoke\_past = as.character(.data[["smoke\_past\_bad"]]),
 )

smoking\_data\_bad |>
 pointblank::col\_vals\_in\_set(
 columns = c("smoke\_current", "smoke\_past"),
 set = c("0", "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"))
 )
 )

Error: The `col\_vals\_expr()` validation failed beyond the absolute threshold level (1).
\* failure level (2) >= failure threshold (1)

### 3.3.4 Chest Pain

#### 3.3.4.1 Shortness of Breath

have\_sob values remained unchanged.

shortness\_of\_breath\_data <- cohort\_A\_data |>
 dplyr::select(c("cohort\_unique\_id", "have\_sob")) |>
 dplyr::mutate(
 have\_sob = forcats::fct\_relevel(
 as.character(.data[["have\_sob"]]),
 c("0", "1"))
 ) |>
 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[["have\_sob"]],
 .keep\_all = TRUE) |>
 knitr::kable()
}

| cohort\_unique\_id | have\_sob |
| --- | --- |
| A001 | 0 |
| A002 | 1 |

#### 3.3.4.2 Have chest pain or not

have\_chest\_pain is grouped as follows:

| chest\_pain\_type | have\_chest\_pain |
| --- | --- |
| 0 | 0 |
| 1, 2 or 3 | 1 |

have\_chest\_pain\_data <- cohort\_A\_data |>
 dplyr::select(c("cohort\_unique\_id", "chest\_pain\_type")) |>
 dplyr::mutate(
 have\_chest\_pain = dplyr::case\_when(
 .data[["chest\_pain\_type"]] %in% c(0) ~ "0",
 .data[["chest\_pain\_type"]] %in% c(1, 2, 3) ~ "1",
 .default = NA\_character\_
 ),
 have\_chest\_pain = forcats::fct\_relevel(
 .data[["have\_chest\_pain"]],
 c("0", "1")
 ),
 chest\_pain\_type = forcats::fct\_relevel(
 as.character(.data[["chest\_pain\_type"]]),
 c("0", "1", "2", "3", "4", "-1")
 )
 ) |>
 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\_type"]],
 .keep\_all = TRUE) |>
 knitr::kable()
}

| cohort\_unique\_id | chest\_pain\_type | have\_chest\_pain |
| --- | --- | --- |
| A001 | 1 | 1 |
| A002 | 0 | 0 |
| A007 | 2 | 1 |
| A012 | 3 | 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\_type"))

#### 3.3.4.3 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\_A\_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 |
| --- | --- | --- | --- |
| A001 | 0 | 1 | 1 |
| A002 | 1 | 0 | 2 |
| A003 | 0 | 0 | 0 |
| A006 | 1 | 1 | 1 |

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

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

#### 3.3.4.4 Chest Pain Type

chest\_pain\_type is grouped as follows:

| have\_sob | chest\_pain\_type before | chest\_pain\_type |
| --- | --- | --- |
| -1 | -1 | -1 |
| 0 | 0 | 0 |
| 0 or 1 | 1 | 1 |
| 0 or 1 | 2 | 2 |
| 0 or 1 | 3 | 2 |
| 1 | 0 | 4 |

chest\_pain\_type\_data <- cohort\_A\_data |>
 dplyr::select(c("cohort\_unique\_id", "chest\_pain\_type")) |>
 dplyr::inner\_join(shortness\_of\_breath\_data,
 by = dplyr::join\_by("cohort\_unique\_id"),
 unmatched = "error",
 relationship = "one-to-one") |>
 dplyr::mutate(
 chest\_pain\_type\_before = .data[["chest\_pain\_type"]],
 chest\_pain\_type = dplyr::case\_when(
 (.data[["chest\_pain\_type\_before"]] == "-1" &
 .data[["have\_sob"]] == "-1"
 ) ~ "-1",
 (.data[["chest\_pain\_type\_before"]] == "0" &
 .data[["have\_sob"]] == "0"
 ) ~ "0",
 (.data[["chest\_pain\_type\_before"]] == "1" &
 .data[["have\_sob"]] %in% c("0", "1")
 ) ~ "1",
 (.data[["chest\_pain\_type\_before"]] == "2" &
 .data[["have\_sob"]] %in% c("0", "1")
 ) ~ "2",
 (.data[["chest\_pain\_type\_before"]] == "3" &
 .data[["have\_sob"]] %in% c("0", "1")
 ) ~ "3",
 (.data[["chest\_pain\_type\_before"]] == "0" &
 .data[["have\_sob"]] == "1"
 ) ~ "4",
 .default = NA\_character\_
 ),
 chest\_pain\_type\_before = forcats::fct\_relevel(
 as.character(.data[["chest\_pain\_type\_before"]]),
 c("0", "1", "2", "3")),
 chest\_pain\_type = forcats::fct\_relevel(
 .data[["chest\_pain\_type"]],
 c("0", "1", "2", "3"))
 ) |>
 dplyr::relocate(
 "chest\_pain\_type\_before",
 .before = "chest\_pain\_type"
 ) |>
 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[["have\_sob"]],
 .data[["chest\_pain\_type\_before"]],
 .keep\_all = TRUE) |>
 knitr::kable()
}

| cohort\_unique\_id | chest\_pain\_type\_before | chest\_pain\_type | have\_sob |
| --- | --- | --- | --- |
| A001 | 1 | 1 | 0 |
| A002 | 0 | 4 | 1 |
| A003 | 0 | 0 | 0 |
| A006 | 1 | 1 | 1 |
| A007 | 2 | 2 | 0 |
| A012 | 3 | 3 | 0 |

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

chest\_pain\_type\_data <- chest\_pain\_type\_data |>
 dplyr::select(-c("have\_sob", "chest\_pain\_type\_before"))

#### 3.3.4.5 Combined chest pain related tables

We combine all chest related tables together

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

chest\_pain\_data <- cohort\_A\_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
)

### 3.3.5 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\_A\_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
)

## 3.4 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\_A.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
)