Harmonisation Template for Cohort A

My Name

2025-07-03

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>.

## Acknowledgement

Layout of this page is inspired from R package [rcompendium](https://frbcesab.github.io/rcompendium).

## File Structure

Here is the file structure of this project.

harmonisation\_template/ # Root of the compendium  
│  
├── harmonisation\_template.Rproj # RStudio project file  
|  
├── .quarto/ # Intermediate files/folders generated   
| # Quarto renders to the documents.  
|  
├── archive/ # Folder to keep previous books and harmonised data  
│ ├── reports/ # Folder containing previous documentation   
| | # of data harmonisation   
| └── harmonised/ # Folder containing previous GPS-CAD harmonised data  
|   
├── reports/ # Documentation of data harmonisation   
│   
├── data-raw/ # Cohort raw data (.csv, .gpkg, etc.)  
│ ├── {cohort name}/ # Folder containing cohort raw data,  
| | # and data dictionary   
| └── data-dictionary/ # Data dictionary for harmonised data  
| └── data-input/ # Data input file from collaborators  
|   
├── docs/ # R functions documentation generating using  
| # pkgdown:::build\_site\_external()  
|   
├── inst/ # Arbitrary additional files to include in the  
| | # package.  
| |   
| └── WORDLIST # File generating by spelling::update\_wordlist()  
|   
├── man/ # R functions helps (automatically updated)  
│ ├── {fun-demo}.Rd # Documentation of the demo R function  
│ └── harmonisation-template.Rd # High-level documentation  
|   
├── quarto-yaml-template/ # Folder containing template files for quarto book generation  
│ ├── \_quarto\_{cohort name}.yml # Quarto book generation for each cohort  
│ └── \_quarto\_all.yml # Quarto book generation for all cohorts  
|   
├── R/ # R functions location  
│ ├── {fun-demo}.R # Example of an R function  
│ └── harmonisation-template-package.R # Dummy R file for high-level documentation  
│   
├── renv/ # Folder that contains all packages   
| # installed in the renv environment.  
|   
├── codes/ # R/Quarto scripts to run data harmonisation  
| ├── quarto\_script.R # R script to render each {cohort name}\_Cleaning/ folder.   
| | # folder into html, pdf and word document.  
│ ├── {cohort name}\_Cleaning/ # Quarto scripts to run data harmonisation  
| | # and output them for each cohort.  
| └── Combine/ # Quarto scripts to filter harmonised data  
| # based on inclusion/exclusion criteria,   
| | # combined the filtered data for preliminary analysis.  
|   
├── tests/ # Test units file created by R package testhat  
│   
├── .lintr # Configuration for linting  
| # R projects and packages using linter  
|   
├── .Rbuildignore # List of files/folders to be ignored while   
│ # checking/installing the package  
|   
├── .renvignore # List of files/folders to be ignored when   
│ # renv is doing its snapshot  
|   
├── \_pkgdown.yml # Configuration for R package documentation  
| # using pkgdown:::build\_site\_external()  
|   
├── \_quarto.yml # Configuration for Quarto book generation  
| # Also the project configuration file  
|   
├── custom-reference.docx # Microsoft word template for data harmonisation   
| # documentation to Word  
|   
├── DESCRIPTION # Project metadata[\*]  
|   
├── index.qmd # Home page of Quarto book content  
|   
├── 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 # Automatically generated  
│   
├── README.md # GitHub README (automatically generated)  
├── README.Rmd # GitHub README (ignore for now)  
│   
|   
├── references.bib # Bibtex file for Quarto book   
|   
├── references.qmd # Reference document for Quarto book  
|   
├── renv.lock # Metadata of R packages installed generated  
| # using renv::snapshot  
|   
├── csl\_file.csl # Citation Style Language (CSL) file to ensure  
| # citations follows the Lancet journal  
  
  
[\*] These files are automatically created but user needs to manually add some information.

## Installation

### Installing R

Go to <https://cran.rstudio.com/>. Choose a version of R that matches the computer’s operating system.

### Installing RStudio

Go to <https://posit.co/download/rstudio-desktop/>. Scroll down and choose a version of RStudio that matches the computer’s operating system.

### Installing Rtools

Go to <https://cran.r-project.org/bin/windows/Rtools/>. Choose a version of Rtools that matches the R version that was installed.

### Quarto

Quarto converts R scripts into a technical report or notebook in html, pdf, Microsoft Word, [etc.](https://quarto.org/docs/output-formats/all-formats.html) It is installed together with RStudio. User can also go to <https://quarto.org/docs/get-started/> to install it separately. For Quarto to be able to create pdf files, a [pdf engine](https://quarto.org/docs/output-formats/pdf-engine.html) must be installed as well. For ease, it is suggested to install [TinyTex](https://yihui.org/tinytex/) using the terminal command quarto install tinytex.

## R Package Installation

Use Posit Public Package Manager [PPM](https://packagemanager.posit.co/client) to set up your repository environment to install R packages from [CRAN](https://cloud.r-project.org/). This is because PPM allows installation of frozen R package versions based on a snapshot date.

One way to do that is to set in the .Rprofile file with the code options(repos = c(P3M = "{link to repository url form Posit Public Package Manager}"))

R packages can be installed using the package [pak](https://pak.r-lib.org/) as an alternative to [install.packages()](https://rdrr.io/r/utils/install.packages.html) and [remotes::install\_github()](https://remotes.r-lib.org/reference/install\_github.html). Benefits of using [pak`](https://pak.r-lib.org/) can be found [here](https://pak.r-lib.org/reference/features.html)

You can also view your respository environment using the command [pak::repo\_get()](https://pak.r-lib.org/reference/repo_get.html)

R package can be loaded using the command library({package\_name}). You can use the R package [annotater](https://annotater.liomys.mx/) to add additional information on what the loaded package does.

## Using renv

You can increase reproducibility by using the package [renv](https://rstudio.github.io/renv/). Install renv from CRAN with pak::pak("renv"). If this is your first time using renv, start with the [Introduction to renv vignette](https://rstudio.github.io/renv/articles/renv.html). Use renv::init(bare = TRUE) to start with an empty renv environment.

renv will freeze the exact package versions you depend on (in renv.lock). This ensures that each collaborator (or you in the future) will use the exact same versions of these packages. Moreover renv provides to each project its own private package library making each project isolated from others.

Install required dependencies locally with install.packages() or [renv::install()](https://rstudio.github.io/renv/reference/install.html) from CRAN, Bioconductor, Github, explicit file path, etc.

Sometimes the right [downloader](https://community.rstudio.com/t/can-not-install-packages-after-initializing-renv/106064) (libcurl or others) needs to set for installation of R packages inside the renv environment to be successful. Setting the R environmental variable RENV\_DOWNLOAD\_FILE\_METHOD = “libcurl” may help.

Save the local environment with [renv::snapshot()](https://rstudio.github.io/renv/reference/snapshot.html) to create the renv.lock file.

## R Functions Management

R functions heavily used in this project can be found in the R folder. Documentation (man folder), test units (test folder) corresponding to these functions are structured the same as creating an R package. Relevant R packages required for R package development (and available on Posit Public Package Manager [PPM](https://packagemanager.posit.co/client)) are

library("usethis")  
library("devtools")  
library("roxygen2")  
library("testthat")  
library("covr")  
library("spelling")  
library("lintr")  
library("sinew")  
library("pkgdown")

Here is an example of the command to use pak::pak("{package name}") to install packages from [PPM](https://packagemanager.posit.co/client).

There is no need to source the functions in the R folder. Use [devtools::load\_all()](https://devtools.r-lib.org/reference/load_all.html) instead. [devtools::load\_all()](https://devtools.r-lib.org/reference/load_all.html) will load required dependencies listed in DESCRIPTION and R functions stored in R/. Prior installation of these dependencies is required for the load to be successful.

After loading, R functions can be documented (using [devtools::document()](https://devtools.r-lib.org/reference/document.html)), tested (using [devtools::test()](https://devtools.r-lib.org/reference/test.html) and then [devtools::check()](https://devtools.r-lib.org/reference/check.html)) and even installed as an R package (using [devtools::install](https://devtools.r-lib.org/reference/install.html)).

More information of this workflow can be found in [Chapter 1: The Whole Game](https://r-pkgs.org/whole-game.html) of the R Packages (2e) book.

## R Packages

R packages installed from Posit Public Package Manager [PPM](https://packagemanager.posit.co/client) using command pak::pak("{package name}") are

library("renv")  
library("sessioninfo")  
library("knitr")  
library("rmarkdown")  
library("quarto")  
library("rlang")  
library("cli")  
  
library("fs")  
library("here")  
library("fst")  
library("readxl")  
library("vroom")  
  
library("dplyr")  
library("tidyr")  
library("magrittr")  
library("stringr")  
library("forcats")  
library("purrr")  
library("lubridate")  
library("tibble")  
library("glue")  
  
library("collateral")  
library("pointblank")  
library("testthat")  
  
library("htmltools")  
library("htmlwidgets")  
library("fontawesome")  
library("reactable")  
library("flextable")  
  
library("openxlsx")  
  
library("harmonisation")

Here are all the R packages used in this analysis.

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

| package | version | date | source |
| --- | --- | --- | --- |
| cli | 3.6.4 | 2025-02-13 | RSPM |
| collateral | 0.5.2 | 2021-10-25 | RSPM |
| covr | 3.6.4 | 2023-11-09 | RSPM |
| devtools | 2.4.5 | 2022-10-11 | RSPM |
| dplyr | 1.1.4 | 2023-11-17 | RSPM |
| flextable | 0.9.7 | 2024-10-27 | RSPM |
| fontawesome | 0.5.3 | 2024-11-16 | RSPM |
| forcats | 1.0.0 | 2023-01-29 | RSPM |
| fs | 1.6.5 | 2024-10-30 | RSPM |
| fst | 0.9.8 | 2022-02-08 | RSPM |
| glue | 1.8.0 | 2024-09-30 | RSPM |
| harmonisation | 0.0.0.9999 | 2025-03-09 | local |
| here | 1.0.1 | 2020-12-13 | RSPM |
| htmltools | 0.5.8.1 | 2024-04-04 | RSPM |
| htmlwidgets | 1.6.4 | 2023-12-06 | RSPM |
| knitr | 1.49 | 2024-11-08 | RSPM |
| lintr | 3.2.0 | 2025-02-12 | RSPM |
| lubridate | 1.9.4 | 2024-12-08 | RSPM |
| magrittr | 2.0.3 | 2022-03-30 | RSPM |
| openxlsx | 4.2.8 | 2025-01-25 | RSPM |
| pkgdown | 2.1.1 | 2024-09-17 | RSPM |
| pointblank | 0.12.2 | 2024-10-23 | RSPM |
| purrr | 1.0.4 | 2025-02-05 | RSPM |
| quarto | 1.4.4 | 2024-07-20 | RSPM |
| reactable | 0.4.4 | 2023-03-12 | RSPM |
| readxl | 1.4.4 | 2025-02-27 | RSPM |
| renv | 1.1.0 | 2025-01-29 | RSPM (R 4.4.0) |
| rlang | 1.1.5 | 2025-01-17 | RSPM |
| rmarkdown | 2.29 | 2024-11-04 | RSPM |
| roxygen2 | 7.3.2 | 2024-06-28 | RSPM |
| sessioninfo | 1.2.2 | 2021-12-06 | CRAN (R 4.4.2) |
| sinew | 0.4.0 | 2022-03-31 | RSPM |
| spelling | 2.3.1 | 2024-10-04 | RSPM |
| stringr | 1.5.1 | 2023-11-14 | RSPM |
| testthat | 3.2.3 | 2025-01-13 | RSPM |
| tibble | 3.2.1 | 2023-03-20 | RSPM |
| tidyr | 1.3.1 | 2024-01-24 | RSPM |
| usethis | 3.1.0 | 2024-11-26 | RSPM |
| vroom | 1.6.5 | 2023-12-05 | RSPM |

## 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 4.4.2 (2024-10-31 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 | 2025-03-10 |
| pandoc | 3.2 @ C:/Program Files/RStudio/resources/app/bin/quarto/bin/tools/ (via rmarkdown) |
| quarto | 1.6.37 @ C:/Program Files/Quarto/bin/quarto.exe/ (via quarto) |
| knitr | 1.49 from RSPM |

## DESCRIPTION

The DESCRIPTION file contains important compendium metadata. Though DESCRIPTION file is specific to R package, it can be used to work with research compendia (see below). For further information on how to edit this file, please read <https://r-pkgs.org/description.html>.

## Data Harmonisation

To start the harmonisation of data, run the R script quarto\_script.R in reports folder.

For each cohort, the script will clean the raw data and create a Quarto book for each cohort in html, word and pdf.

This involves copying a specific yml file (\_quarto\_{cohort name}.yml) from the quarto-yaml-template folder to the project folder harmonisation\_template and rename it as \_quarto.yml, overwriting any existing \_quarto.yml file. Using the \_quarto.yml file. Quarto will then start running the Quarto scripts in the reports/{cohort\_name}\_Cleaning folder. This involves reading the raw data in the data-raw/{cohort\_name} folder, placing preprocessing data in the reports/{cohort\_name}\_Cleaning/preprocessed\_data folder, outputting the harmonised data as excel file called cleaned\_{cohort\_name}.xlsx in the reports folder. Also, the data harmonisation process documentation will be created in the books/{cohort\_name} folder as a Quarto book in html, word and pdf.

After data harmonisation, data combining for all cohorts, data filtering and preliminary analysis will be done by copying \_quarto\_Prelim.yml file from the quarto-yaml-template folder to the project folder harmonisation\_template and rename it as \_quarto.yml, overwriting any existing \_quarto.yml file. Using the \_quarto.yml file, Quarto runs the Quarto scripts in the reports/Combine folder. Results will be outputted as excel files called harmonised.xlsx, harmonised\_batch1.xlsx, harmonised\_batch2.xlsx in the reports folder. In addition, the preliminary results will be created in the books/Prelim folder as a Quarto book in html, word and pdf.

After doing this for each cohort, the script will then create a combined data harmonisation process documentation (for all the cohorts) as a Quarto book in html. The specific yml file (\_quarto\_all.yml) in the quarto-yaml-template folder will be used and the documentation will be created in the books/all folder. Data combining for all cohorts, data filtering and preliminary analysis will also be done by running Quarto scripts in the reports/Combine folder.

## General Recommendations

* Ensure the workspace is always in a blank state. Use [usethis::use\_blank\_slate(scope = c("user", "project"))](https://usethis.r-lib.org/reference/use_blank_slate.html) to create this setting.
* Keep the root of the project as clean as possible
* Store your raw data in data-raw
* Document raw data modifications. See Flowchart.xlsx.
* Export modified raw data in reports/{cohort\_name}\_Cleaning/preprocessed\_data
* Store only **R functions** in R/
* Store only **R scripts** and/or **qmd** in reports/{cohort\_name}\_Cleaning
* Built relative paths using [here::here()](https://here.r-lib.org/reference/here.html)
* Call external functions as {package\_name}::{function()}
* Use [devtools::document()](https://devtools.r-lib.org/reference/document.html) to update the NAMESPACE
* Use [rcompendium::add\_dependencies](https://frbcesab.github.io/rcompendium/reference/add_dependencies.html) to update the list of required dependencies in DESCRIPTION
* Do not source your functions but use instead [devtools::load\_all()](https://devtools.r-lib.org/reference/load_all.html). [devtools::load\_all()](https://devtools.r-lib.org/reference/load_all.html) will load required dependencies listed in DESCRIPTION and R functions stored in R/

# 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 |
| fontawesome | 0.5.3 | 2024-11-16 | RSPM |
| forcats | 1.0.0 | 2023-01-29 | RSPM |
| glue | 1.8.0 | 2024-09-30 | RSPM |
| harmonisation | 0.0.0.9999 | 2025-03-09 | local |
| here | 1.0.1 | 2020-12-13 | RSPM |
| htmltools | 0.5.8.1 | 2024-04-04 | RSPM |
| lubridate | 1.9.4 | 2024-12-08 | RSPM |
| magrittr | 2.0.3 | 2022-03-30 | RSPM |
| openxlsx | 4.2.8 | 2025-01-25 | RSPM |
| pointblank | 0.12.2 | 2024-10-23 | RSPM |
| purrr | 1.0.4 | 2025-02-05 | RSPM |
| quarto | 1.4.4 | 2024-07-20 | RSPM |
| reactable | 0.4.4 | 2023-03-12 | RSPM |
| readxl | 1.4.4 | 2025-02-27 | RSPM |
| sessioninfo | 1.2.2 | 2021-12-06 | CRAN (R 4.4.2) |
| stringr | 1.5.1 | 2023-11-14 | RSPM |
| testthat | 3.2.3 | 2025-01-13 | RSPM |
| tibble | 3.2.1 | 2023-03-20 | RSPM |
| tidyr | 1.3.1 | 2024-01-24 | RSPM |

## 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 4.4.2 (2024-10-31 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 | 2025-03-10 |
| pandoc | 3.2 @ C:/Program Files/RStudio/resources/app/bin/quarto/bin/tools/ (via rmarkdown) |
| quarto | 1.6.37 @ C:/Program Files/Quarto/bin/quarto.exe/ (via quarto) |
| knitr | 1.49 from RSPM |

## 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 Age 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 age.

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::left\_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",  
 .before = "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  
 )

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 = 15,  
 right = 50,  
 inclusive = c(TRUE, TRUE),  
 na\_pass = TRUE  
 )

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"))  
 )  
 )

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")  
 )

#### 3.3.4.2 Have chest pain or not

have\_chest\_pain is grouped as follows:

| chest\_pain\_type | have\_chest\_pain |
| --- | --- |
| 0 or 4 | 0 |
| 1, 2 or 3 | 1 |
| -1 | -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, 4) ~ "0",  
 .data[["chest\_pain\_type"]] %in% c(1, 2, 3) ~ "1",  
 .data[["chest\_pain\_type"]] %in% c(-1) ~ "-1",  
 .default = NA\_character\_  
 ),  
 have\_chest\_pain = forcats::fct\_relevel(  
 .data[["have\_chest\_pain"]],  
 c("0", "1", "-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")  
 )

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::left\_join(shortness\_of\_breath\_data,   
 by = dplyr::join\_by("cohort\_unique\_id"),  
 unmatched = "error",  
 relationship = "one-to-one") |>   
 dplyr::left\_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")  
 )

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::left\_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 = "cohort\_unique\_id"  
 ) |>   
 pointblank::col\_vals\_in\_set(  
 columns = c("chest\_pain\_type"),  
 set = c("0", "1", "2", "3", "4")  
 )

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::left\_join(have\_chest\_pain\_data,   
 by = join\_specification,  
 unmatched = "error",  
 relationship = "one-to-one") |>   
 dplyr::left\_join(chest\_pain\_type\_data,   
 by = join\_specification,  
 unmatched = "error",  
 relationship = "one-to-one") |>   
 dplyr::left\_join(shortness\_of\_breath\_data,   
 by = join\_specification,  
 unmatched = "error",  
 relationship = "one-to-one") |>   
 dplyr::left\_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::left\_join(age\_gender\_data,  
 by = join\_specification,  
 unmatched = "error",  
 relationship = "one-to-one") |>  
 dplyr::left\_join(body\_measurement\_data,  
 by = join\_specification,  
 unmatched = "error",  
 relationship = "one-to-one") |>  
 dplyr::left\_join(smoking\_data,  
 by = join\_specification,  
 unmatched = "error",  
 relationship = "one-to-one") |>  
 dplyr::left\_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"),  
)