

# Harmonisation Template for Cohort B

My Name

2025-03-10



# Table of contents

Acknowledgement . . . . .	5
File Structure . . . . .	5
Installation . . . . .	8
Installing R . . . . .	8
Installing RStudio . . . . .	8
Installing Rtools . . . . .	9
Quarto . . . . .	9
R Package Installation . . . . .	9
Using <code>renv</code> . . . . .	9
R Functions Management . . . . .	10
R Packages . . . . .	10
R Platform Information . . . . .	13
DESCRIPTION . . . . .	13
Data Harmonisation . . . . .	13
General Recommendations . . . . .	14
<b>I Cohort B Cleaning</b>	<b>15</b>
<b>1 R Package And Environment</b>	<b>16</b>
1.1 R Packages Used . . . . .	16
1.2 R Platform Information . . . . .	16
1.3 Data dictionary . . . . .	17
<b>2 Read Cohort B Data</b>	<b>18</b>
2.1 Read Data . . . . .	18
2.2 Write Preprocessed File . . . . .	20
<b>3 Extract Demographic</b>	<b>21</b>
3.1 Read Preprocessed File . . . . .	21
3.2 Demographics and Behavioral parameters . . . . .	21
3.2.1 Age and Sex . . . . .	21
3.2.2 Height, Weight, BMI and BSA . . . . .	22
3.2.3 Smoking History . . . . .	23
3.2.4 Chest Pain . . . . .	25
3.2.4.1 Shortness of Breath . . . . .	25
3.2.4.2 Have chest pain or not . . . . .	26
3.2.4.3 Symptomatic or Asymptomatic . . . . .	27

3.2.4.4	Chest Pain Type . . . . .	28
3.2.4.5	Combined chest pain related tables . . . . .	29
3.2.5	Combine Demographics . . . . .	30
3.3	Write Preprocessed File . . . . .	31
<b>4</b>	<b>Export To Excel</b>	<b>32</b>
4.1	Read all tabular data . . . . .	32
4.2	Export Data as Excel . . . . .	32

# Preface

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

## Acknowledgement

Layout of this page is inspired from R package [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
|   |   |
|   |   |   documents.                                # Quarto renders to the
|   |   |   |
|   |   |   |   archive/                             # Folder to keep previous books
|   |   |   |   and harmonised data
|   |   |   |   |   reports/                         # Folder containing previous
|   |   |   |   |   |   documentation
|   |   |   |   |   |   |
|   |   |   |   |   |   |   # of data harmonisation
|   |   |   |   |   |   |   # 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}.yaml	# Quarto book generation for each
cohort		
	_quarto_all.yaml	# 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.		

<pre>       {cohort name}_Cleaning/ harmonisation      cohort.      Combine/ harmonised data   criteria,      preliminary analysis.       tests/ package testthat      .lintr   linter       .Rbuildignore ignored while        .renvignore ignored when        _pkgdown.yml documentation   pkgdown:::build_site_external()       _quarto.yml generation   file       custom-reference.docx harmonisation         DESCRIPTION       index.qmd       LICENSE generated via </pre>	<pre> # Quarto scripts to run data      # and output them for each  # Quarto scripts to filter      # based on inclusion/exclusion      # combined the filtered data for  # Test units file created by R  # Configuration for linting     # R projects and packages using  # List of files/folders to be      # checking/installing the package  # List of files/folders to be      # renv is doing its snapshot  # Configuration for R package      # using  # Configuration for Quarto book      # Also the project configuration  # Microsoft word template for data      # documentation to Word  # Project metadata[*]  # Home page of Quarto book content  # Content of the MIT license </pre>
---	--

	# 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.* 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](#) must be installed as well. For ease, it is suggested to install [TinyTex](#) using the terminal command `quarto install tinytex`.

## R Package Installation

Use Posit Public Package Manager [PPM](#) to set up your repository environment to install R packages from [CRAN](#). 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](#) as an alternative to `install.packages()` and `remotes::install_github()`. Benefits of using [pak](#) can be found [here](#)

You can also view your repository environment using the command `pak::repo_get()`

R package can be loaded using the command `library({package_name})`. You can use the R package [annotater](#) to add additional information on what the loaded package does.

## Using renv

You can increase reproducibility by using the package [renv](#). Install [renv](#) from CRAN with `pak::pak("renv")`. If this is your first time using [renv](#), start with the [Introduction to renv vignette](#). 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()` from CRAN, Bioconductor, Github, explicit file path, etc.

Sometimes the right [downloader](#) (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()` 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](#)) 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](#).

There is no need to source the functions in the `R` folder. Use `devtools::load_all()` instead. `devtools::load_all()` 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()`), tested (using `devtools::test()` and then `devtools::check()`) and even installed as an R package (using `devtools::install`).

More information of this workflow can be found in [Chapter 1: The Whole Game](#) of the R Packages (2e) book.

## R Packages

R packages installed from Posit Public Package Manager [PPM](#) 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	2025-03-12
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 yaml file (`_quarto_{cohort name}.yaml`) from the `quarto-yaml-template` folder to the project folder `harmonisation_template` and rename it as `_quarto.yaml`, overwriting any existing `_quarto.yaml` file. Using the `_quarto.yaml` 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"))` 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()`
- Call external functions as `{package_name}::{function()}`
- Use `devtools::document()` to update the `NAMESPACE`
- Use `rcompendium::add_dependencies` to update the list of required dependencies in `DESCRIPTION`
- Do not source your functions but use instead `devtools::load_all()`. `devtools::load_all()` will load required dependencies listed in `DESCRIPTION` and R functions stored in `R/`

**Part I**

**Cohort B Cleaning**

# 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
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
vroom	1.6.5	2023-12-05	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	2025-03-12
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 B Data

### 2.1 Read Data

We read the file `data_to_harmonise_age_issue.csv` using `vroom::vroom`

```
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`

If there are issues with the data, the output of `vroom::problems` will be a `tibble`.

```
cohort_B_data |>  
  vroom::problems()
```

```
# A tibble: 3 x 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`

```

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",
  .before = "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
)

```

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  $\sqrt{[Height(cm) \times Weight(kg)]/3600}$  `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 = 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"))
```

### 3.2.3 Smoking History

smoke\_current is grouped as follows:

Smoke History	smoke_current
non-smoker	0
past smoker	0
current smoker	1

smoke\_past is grouped as follows:

Smoke History	smoke_past
non-smoker	0
past smoker	0
current smoker	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"))
    )
  )
)

```

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")
  )

```

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")
  )
```

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::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"))
```

#### 3.2.4.4 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")
)

```

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::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
)

```

### 3.2.5 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::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
)

```

### 3.3 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  
)
```