# Harmonisation Template for Cohort B

My Name

2025-03-10

# Table of contents

# Preface

Here is the documentation of the data harmonisation step generated using Quarto. 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.

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

## 1.2 R Platform Information

Here are the R platform environment used in this analysis.

```
harmonisation::get_r_platform_info() |>
  knitr::kable()
```

| setting | value |
|---|---|
| version | R version 4.5.0 (2025-04-11 ucrt) |
| os | Windows 11 x64 (build 26100) |
| system | x86_64, mingw32 |
| ui | RTerm |
| language | (EN) |
| collate | English_Singapore.utf8 |
| ctype | English_Singapore.utf8 |
| tz | Asia/Singapore |
| date | 2025-06-11 |
| pandoc | 3.4 @ C:/Program Files/RStudio/resources/app/bin/quarto/bin/tools/ (via rmarkdown) |
| quarto | 1.7.30 @ C:/Program Files/Quarto/bin/quarto.exe/ (via quarto) |
| knitr | 1.49 from RSPM (R 4.5.0) |

## 1.3 Data dictionary

Check to see if the data dictionary 20250310_data_dictionary.xlsx exists.

```
dict_relative_path <- fs::path(
  "data-raw",
  "data_dictionary",
  params$data_dictionary
)


dict_path <- here::here(dict_relative_path)

if (!file.exists(dict_path)) {
  stop(glue::glue("Input data dictionary {dict_path} cannot be found"))
}
```

# 2 Read Cohort B Data

## 2.1 Read Data

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

```
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",
    .after = "Sex"
  ) |>
  dplyr::relocate(
    "age_years",
    .after = "Age"
  ) |>
  pointblank::col_vals_in_set(
    columns = "sex",
    set = c("0", "1")
  ) |>
  pointblank::col_vals_between(
    columns = "age_years",
    left = 0,
    right = 100,
    inclusive = c(FALSE, TRUE),
    na_pass = TRUE
  )
```

```
if (params$show_table) {
  age_gender_data |>
    dplyr::distinct(.data[["Sex"]],
                    .keep_all = TRUE) |>
    knitr::kable()
}
```

| cohort_unique_id | Age | age_years | Sex | sex |
|---|---|---|---|---|
| B001 | 32 | 32 | Female | 0 |
| B003 | 80 | 80 | Male | 1 |

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

```
age_gender_data <- age_gender_data |>
  dplyr::select(-c("Age", "Sex"))
```

### 3.2.2 Height, Weight, BMI and BSA

`height_cm` will be mapped from the column `Height`. `weight_kg` will be mapped from the column `Weight`.

`bsa_m2` in m^2 will be calculated as sqrt([Height(cm) x Weight(kg)]/3600) `bmi` will be calculated as Weight(kg)/((Height(m))^2)

All values are then converted to two decimal places.

```
body_measurement_data <- cohort_B_data |>
  dplyr::select(c("cohort_unique_id",
                  "Height", "Weight")) |>
  dplyr::mutate(
    height_cm = .data[["Height"]],
    weight_kg = .data[["Weight"]],
    bsa_m2 = sqrt((.data[["height_cm"]] * .data[["weight_kg"]]) / 3600),
    bsa_m2 = harmonisation::round_to_nearest_digit(.data[["bsa_m2"]],
     ↪  digits = 2),
    bmi = .data[["weight_kg"]] / ((.data[["height_cm"]] / 100)^2),
    bmi = harmonisation::round_to_nearest_digit(.data[["bmi"]], digits =
     ↪  2),
    height_cm = harmonisation::round_to_nearest_digit(.data[["height_cm"]],
     ↪  digits = 2),
    weight_kg = harmonisation::round_to_nearest_digit(.data[["weight_kg"]],
     ↪  digits = 2)
  ) |>
  pointblank::col_vals_between(
    columns = "bmi",
    left = 10,
    right = 50,
    inclusive = c(TRUE, TRUE),
    na_pass = TRUE
  )
```

```
if (params$show_table) {
  body_measurement_data |>
    head(n = 5) |>
    knitr::kable()
}
```

| cohort_unique_id | Height | Weight | height_cm | weight_kg | bsa_m2 | bmi |
|---|---|---|---|---|---|---|
| B001 | 170 | 63 | 170 | 63 | 1.72 | 21.80 |
| B002 | 167 | 71 | 167 | 71 | 1.81 | 25.46 |
| B003 | 184 | 77 | 184 | 77 | 1.98 | 22.74 |

| cohort_unique_id | Height | Weight | height_cm | weight_kg | bsa_m2 | bmi |
|---|---|---|---|---|---|---|
| B004 | 160 | 83 | 160 | 83 | 1.92 | 32.42 |
| B005 | 155 | 61 | 155 | 61 | 1.62 | 25.39 |

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

```
body_measurement_data <- body_measurement_data |>
  dplyr::select(-c("Height", "Weight"))
```

### 3.2.3 Smoking History

`smoke_current` is grouped as follows:

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

`smoke_past` is grouped as follows:

| Smoke History | smoke_past |
|---|---|
| non-smoker | 0 |
| past smoker | 1 |
| current smoker | 0 |
| NA | -1 |

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

- `smoke_current` as 1 and `smoke_past` as 0 for current smokers
- `smoke_current` as 0 and `smoke_past` as 1 for past smokers
- `smoke_current` as 0 and `smoke_past` as 0 for non-smokers
- `smoke_current` as -1 and `smoke_past` as -1 for unknown

```
smoking_data <- cohort_B_data |>
  dplyr::select(c("cohort_unique_id",
                  "Smoke History")) |>
  dplyr::mutate(
    smoke_current = dplyr::case_when(
      is.na(.data[["Smoke History"]]) ~ "-1",
```

```
      .data[["Smoke History"]] == "non-smoker" ~ "0",
      .data[["Smoke History"]] == "past smoker" ~ "0",
      .data[["Smoke History"]] == "current smoker" ~ "1",
      .default = NA_character_
    ),
    smoke_current = forcats::fct_relevel(
      .data[["smoke_current"]],
      c("0", "1")),
    smoke_past = dplyr::case_when(
      is.na(.data[["Smoke History"]]) ~ "-1",
      .data[["Smoke History"]] == "non-smoker" ~ "0",
      .data[["Smoke History"]] == "past smoker" ~ "1",
      .data[["Smoke History"]] == "current smoker" ~ "0",
      .default = NA_character_
    ),
    smoke_past = forcats::fct_relevel(
      .data[["smoke_past"]],
      c("0", "1")),
    `Smoke History` = forcats::fct(
      .data[["Smoke History"]]
    )
  ) |>
  pointblank::col_vals_in_set(
    columns = c("smoke_current", "smoke_past"),
    set = c("0", "1", "-1")
  ) |>
  pointblank::col_vals_expr(
    expr = pointblank::expr(
      (.data[["smoke_current"]] == "1" & .data[["smoke_past"]] == "0") |
      (.data[["smoke_current"]] == "-1" & .data[["smoke_past"]] == -"1") |
      (.data[["smoke_current"]] == "0" & .data[["smoke_past"]] %in% c("0",
↪   "1"))
    )
  )
```

```
if (params$show_table) {
  smoking_data |>
    dplyr::distinct(.data[["Smoke History"]],
                    .keep_all = TRUE) |>
    knitr::kable()
}
```

| cohort_unique_id | Smoke History | smoke_current | smoke_past |
|---|---|---|---|
| B001 | non-smoker | 0 | 0 |
| B002 | current smoker | 1 | 0 |
| B004 | past smoker | 0 | 1 |
| B017 | NA | -1 | -1 |

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

```
smoking_data <- smoking_data |>
  dplyr::select(-c("Smoke History"))
```

### 3.2.4 Chest Pain

#### 3.2.4.1 Shortness of Breath

`have_sob` is grouped as follows:

| Dyspnea | have_sob |
|---|---|
| no | 0 |
| yes | 1 |

```
shortness_of_breath_data <- cohort_B_data |>
  dplyr::select(c("cohort_unique_id", "Dyspnea")) |>
  dplyr::mutate(
    have_sob = dplyr::case_when(
      .data[["Dyspnea"]] == "no" ~ "0",
      .data[["Dyspnea"]] == "yes" ~ "1",
      .default = NA_character_
    ),
    have_sob = forcats::fct_relevel(
      as.character(.data[["have_sob"]]),
      c("0", "1")),
    Dyspnea = forcats::fct_relevel(
      as.character(.data[["Dyspnea"]]),
      c("no", "yes")),
  ) |>
  pointblank::col_vals_in_set(
    columns = c("have_sob"),
    set = c("0", "1", "-1")
  )
```

```
if (params$show_table) {
  shortness_of_breath_data |>
    dplyr::distinct(.data[["Dyspnea"]],
                    .keep_all = TRUE) |>
    knitr::kable()
}
```

| cohort_unique_id | Dyspnea | have_sob |
|---|---|---|
| B001 | no | 0 |
| B002 | yes | 1 |

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

```
shortness_of_breath_data <- shortness_of_breath_data |>
  dplyr::select(-c("Dyspnea"))
```

#### 3.2.4.2 Have chest pain or not

`have_chest_pain` is grouped as follows:

| Chest Pain Character | have_chest_pain |
|---|---|
| no chest pain | 0 |
| typical, atypical or nonanginal | 1 |

```
have_chest_pain_data <- cohort_B_data |>
  dplyr::select(c("cohort_unique_id", "Chest Pain Character")) |>
  dplyr::mutate(
    have_chest_pain = dplyr::case_when(
      .data[["Chest Pain Character"]] %in% c("no chest pain") ~ "0",
      .data[["Chest Pain Character"]] %in% c("typical", "atypical",
↪   "nonanginal") ~ "1",
      .default = NA_character_
    ),
    have_chest_pain = forcats::fct_relevel(
      .data[["have_chest_pain"]],
      c("0", "1")
    ),
    `Chest Pain Character` = forcats::fct_relevel(
      as.character(.data[["Chest Pain Character"]]),
      c("no chest pain", "typical", "atypical", "nonanginal")
```

```
    )
  ) |>
  pointblank::col_vals_in_set(
    columns = c("have_chest_pain"),
    set = c("0", "1")
  )
```

```
if (params$show_table) {
  have_chest_pain_data |>
    dplyr::distinct(.data[["Chest Pain Character"]],
                    .keep_all = TRUE) |>
    knitr::kable()
}
```

| cohort_unique_id | Chest Pain Character | have_chest_pain |
|---|---|---|
| B001 | atypical | 1 |
| B002 | no chest pain | 0 |
| B005 | typical | 1 |
| B006 | nonanginal | 1 |

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

```
have_chest_pain_data <- have_chest_pain_data |>
  dplyr::select(-c("Chest Pain Character"))
```

#### 3.2.4.3 Symptomatic or Asymptomatic

`symptoms` is grouped as follows:

| have_sob | have_chest_pain | symptoms |
|---|---|---|
| -1 | -1 | -1 |
| 0 | 0 | 0 |
| 0 or 1 | 1 | 1 |
| 1 | 0 | 2 |

```
symptoms_data <- cohort_B_data |>
  dplyr::select(c("cohort_unique_id")) |>
  dplyr::inner_join(shortness_of_breath_data,
                    by = dplyr::join_by("cohort_unique_id"),
                    unmatched = "error",
```

```
                    relationship = "one-to-one") |>
  dplyr::inner_join(have_chest_pain_data,
                    by = dplyr::join_by("cohort_unique_id"),
                    unmatched = "error",
                    relationship = "one-to-one") |>
  dplyr::mutate(
    symptoms = dplyr::case_when(
      (.data[["have_chest_pain"]] == "-1" &
       .data[["have_sob"]] == "-1"
      ) ~ "-1",
      (.data[["have_chest_pain"]] == "0" &
       .data[["have_sob"]] == "0"
      ) ~ "0",
      (.data[["have_chest_pain"]] == "1" &
       .data[["have_sob"]] %in% c("0", "1")
      ) ~ "1",
      (.data[["have_chest_pain"]] == "0" &
       .data[["have_sob"]] == "1"
      ) ~ "2",
      .default = NA_character_
    ),
    symptoms = forcats::fct_relevel(
      .data[["symptoms"]],
      c("0", "1", "2"))
  ) |>
  pointblank::col_vals_in_set(
    columns = c("symptoms"),
    set = c("0", "1", "2")
  )
```

```
if (params$show_table) {
  symptoms_data |>
    dplyr::distinct(.data[["have_chest_pain"]], .data[["have_sob"]],
                    .keep_all = TRUE) |>
    knitr::kable()
}
```

| cohort_unique_id | have_sob | have_chest_pain | symptoms |
|---|---|---|---|
| B001 | 0 | 1 | 1 |
| B002 | 1 | 0 | 2 |
| B003 | 1 | 1 | 1 |
| B007 | 0 | 0 | 0 |

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

```
symptoms_data <- symptoms_data |>
  dplyr::select(-c("have_chest_pain", "have_sob"))
```

#### 3.2.4.4 Chest Pain Type

`chest_pain_type` is grouped as follows:

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

```
chest_pain_type_data <- cohort_B_data |>
  dplyr::select(c("cohort_unique_id", "Chest Pain Character", "Dyspnea"))
↪  |>
  dplyr::mutate(
    chest_pain_type = dplyr::case_when(
      (.data[["Chest Pain Character"]] == "no chest pain" &
       .data[["Dyspnea"]] == "no"
      ) ~ "0",
      (.data[["Chest Pain Character"]] == "typical" &
       .data[["Dyspnea"]] %in% c("no", "yes")
      ) ~ "1",
      (.data[["Chest Pain Character"]] == "atypical" &
       .data[["Dyspnea"]] %in% c("no", "yes")
      ) ~ "2",
      (.data[["Chest Pain Character"]] == "nonanginal" &
       .data[["Dyspnea"]] %in% c("no", "yes")
      ) ~ "3",
      (.data[["Chest Pain Character"]] == "no chest pain" &
       .data[["Dyspnea"]] == "yes"
      ) ~ "4",
      .default = NA_character_
    ),
    `Chest Pain Character` = forcats::fct_relevel(
      as.character(.data[["Chest Pain Character"]]),
      c("no chest pain", "typical", "atypical", "nonanginal")
    ),
    `Dyspnea` = forcats::fct_relevel(
      as.character(.data[["Dyspnea"]]),
      c("no", "yes")
```

```
    ),
    chest_pain_type = forcats::fct_relevel(
      .data[["chest_pain_type"]],
      c("0", "1", "2", "3"))
  ) |>
  dplyr::relocate(
    "Chest Pain Character",
    .after = "cohort_unique_id"
  ) |>
  pointblank::col_vals_in_set(
    columns = c("chest_pain_type"),
    set = c("0", "1", "2", "3", "4")
  )
```

```
if (params$show_table) {
  chest_pain_type_data |>
    dplyr::distinct(.data[["Dyspnea"]], .data[["Chest Pain Character"]],
                    .keep_all = TRUE) |>
    knitr::kable()
}
```

| cohort_unique_id | Chest Pain Character | Dyspnea | chest_pain_type |
|---|---|---|---|
| B001 | atypical | no | 2 |
| B002 | no chest pain | yes | 4 |
| B003 | atypical | yes | 2 |
| B005 | typical | no | 1 |
| B006 | nonanginal | yes | 3 |
| B007 | no chest pain | no | 0 |
| B013 | nonanginal | no | 3 |
| B015 | typical | yes | 1 |

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

```
chest_pain_type_data <- chest_pain_type_data |>
  dplyr::select(-c("Dyspnea", "Chest Pain Character"))
```

#### 3.2.4.5 Combined chest pain related tables

We combine all chest related tables together

```
join_specification <- dplyr::join_by("cohort_unique_id")

chest_pain_data <- cohort_B_data |>
  dplyr::select(c("cohort_unique_id")) |>
  dplyr::inner_join(have_chest_pain_data,
                    by = join_specification,
                    unmatched = "error",
                    relationship = "one-to-one") |>
  dplyr::inner_join(chest_pain_type_data,
                    by = join_specification,
                    unmatched = "error",
                    relationship = "one-to-one") |>
  dplyr::inner_join(shortness_of_breath_data,
                    by = join_specification,
                    unmatched = "error",
                    relationship = "one-to-one") |>
  dplyr::inner_join(symptoms_data,
                    by = join_specification,
                    unmatched = "error",
                    relationship = "one-to-one")

testthat::expect_true(
  pointblank::has_columns(
    chest_pain_data,
    columns = c("have_sob", "have_chest_pain", "symptoms",
     ↪  "chest_pain_type")
  )
)

testthat::expect_equal(
  ncol(chest_pain_data), 5
)
```

### 3.2.5 Combine Demographics

We combine all the data to give the `demo_behave_data`.

```
join_specification <- dplyr::join_by("cohort_unique_id")

demo_behave_data <- cohort_B_data |>
  dplyr::select(c("cohort_unique_id")) |>
  dplyr::inner_join(age_gender_data,
                    by = join_specification,
                    unmatched = "error",
```

```
                    relationship = "one-to-one") |>
  dplyr::inner_join(body_measurement_data,
                    by = join_specification,
                    unmatched = "error",
                    relationship = "one-to-one") |>
  dplyr::inner_join(smoking_data,
                    by = join_specification,
                    unmatched = "error",
                    relationship = "one-to-one") |>
  dplyr::inner_join(chest_pain_data,
                    by = join_specification,
                    unmatched = "error",
                    relationship = "one-to-one") |>
  dplyr::relocate(c("bsa_m2", "bmi"),
                  .after = "sex")


testthat::expect_true(
  pointblank::has_columns(
    demo_behave_data,
    columns = c(
      "age_years", "sex",
      "height_cm", "weight_kg", "bsa_m2", "bmi",
      "smoke_current", "smoke_past",
      "have_sob", "have_chest_pain",
      "symptoms", "chest_pain_type"
    )
  )
)

testthat::expect_equal(
  ncol(demo_behave_data), 13
)
```

## 3.3 Write Preprocessed File

We output data to be used for the next session.

```
demo_behave_data |>
  fst::write_fst(
    path = here::here(
      params$analysis_folder,
      params$harmonisation_folder,
      params$preprocessing_folder,
```

```
      "02_demographic_data.fst"),
)
```

# 4 Export To Excel

```
out_type <- knitr::opts_chunk$get("rmarkdown.pandoc.to")
```

## 4.1 Read all tabular data

We read all tabular data from the previous section.

## 4.2 Export Data as Excel

We export the standardised data to an excel file called harmonised_Cohort_B.xlsx

```
# Create a new workbook
my_workbook <- openxlsx::createWorkbook()

sheet_name = c("demographics")

output_data = list(demo_behave_data)  |>
  purrr::map(
    .f = harmonisation::add_cohort_name,
    cohort_name = params$cohort_name,
    cohort_name_column = "cohort_name"
  )

purrr::walk2(
    .x = sheet_name,
    .y = output_data,
    .f = harmonisation::write_to_sheet,
    workbook = my_workbook
)

# Save workbook
openxlsx::saveWorkbook(
  wb = my_workbook,
  file = here::here(params$analysis_folder,
                    params$output_folder,
```

```
                    params$cleaned_folder,
                    params$output_excel_file),
  overwrite = TRUE
)
```