Harmonisation Template for Cohort A

My Name

2025-03-10

Table of contents

Preface

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

1 R Package And Environment

1.1 R Packages Used

Here are the R packages used in this analysis.

```r
harmonisation::get_r_package_info() |>
  knitr::kable()
```

package	version	date	source
dplyr	1.1.4	2023-11-17	RSPM (R 4.5.0)
fontawesome	0.5.3	2024-11-16	RSPM (R 4.5.0)
forcats	1.0.0	2023-01-29	RSPM (R 4.5.0)
glue	1.8.0	2024-09-30	RSPM (R 4.5.0)
harmonisation	1.0.0.0	2025-05-20	local
here	1.0.1	2020-12-13	RSPM (R 4.5.0)
htmltools	0.5.8.1	2024-04-04	RSPM (R 4.5.0)
magrittr	2.0.3	2022-03-30	RSPM (R 4.5.0)
openxlsx	4.2.8	2025-01-25	RSPM (R 4.5.0)
pointblank	0.12.2	2024-10-23	RSPM (R 4.5.0)
purrr	1.0.4	2025-02-05	RSPM (R 4.5.0)
quarto	1.4.4	2024-07-20	RSPM (R 4.5.0)
reactable	0.4.4	2023-03-12	RSPM (R 4.5.0)
readxl	1.4.4	2025-02-27	RSPM (R 4.5.0)
sessioninfo	1.2.2	2021-12-06	CRAN (R 4.5.0)
stringr	1.5.1	2023-11-14	RSPM (R 4.5.0)
testthat	3.2.3	2025-01-13	RSPM (R 4.5.0)
tibble	3.2.1	2023-03-20	RSPM (R 4.5.0)
tidyr	1.3.1	2024-01-24	RSPM (R 4.5.0)

1.2 R Platform Information

Here are the R platform environment used in this analysis.

```r
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.

```r
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`.

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>%`.

```
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` to check if the column `Serial Number` has unique values.

```
cohort_A_data <- readxl::read_excel(
  path = here::here("data-raw",
                    "Cohort_A",
                    "data_to_harmonise_age_issue_fixed.xlsx"),
  sheet = "Sheet1",
  col_types = c(
    "text", "numeric"
    )
  ) %T>%
  testthat::expect_no_condition() |>
  dplyr::rename(cohort_unique_id = "Serial Number") |>
  # Remove rows when the ID value is NA
  dplyr::filter(!is.na(.data[["cohort_unique_id"]])) |>
  dplyr::mutate(
    cohort_unique_id = as.character(cohort_unique_id)
  ) |>
  # Remove white spaces in column names
  dplyr::rename_all(stringr::str_trim) |>
  # Check if cohort id is unique
  pointblank::rows_distinct(
    columns = "cohort_unique_id",
  )
```

2.3 Clean Weight columns

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

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

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

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

```
cohort_A_data <- readxl::read_excel(
  path = here::here("data-raw",
                    "Cohort_A",
                    "data_to_harmonise.xlsx"),
  sheet = "Sheet1",
  col_types = c(
    "text", # unique id
    "numeric", "text", # age and sex
    "numeric", "numeric", # height and weight
```

```
    "numeric", "numeric", "numeric", "numeric", # smoking history
    "numeric", "numeric" # symptoms
    )
  ) %T>%
  testthat::expect_no_condition() |>
  dplyr::rename(cohort_unique_id = "Serial Number") |>
  # Remove rows when the ID value is NA
  dplyr::filter(!is.na(.data[["cohort_unique_id"]])) |>
  dplyr::mutate(
    cohort_unique_id = as.character(cohort_unique_id)
  ) |>
  # Remove white spaces in column names
  dplyr::rename_all(stringr::str_trim) |>
  # Check if cohort id is unique
  pointblank::rows_distinct(
    columns = "cohort_unique_id",
  )
```

2.4 Update Weight

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

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

```
weight_data <- cohort_A_data |>
  dplyr::select(c("cohort_unique_id", "weight")) |>
  # Check if these patient IDs are present
  pointblank::col_vals_make_subset(
    columns = c("cohort_unique_id"),
    set = c("A018")
  ) |>
  dplyr::mutate(
    updated_weight = dplyr::case_when(
      .data[["cohort_unique_id"]] == "A018" & .data[["weight"]] == 215.4 ~
 ↪  90.1,
      .default = .data[["weight"]]
    ),
  )
```

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

```
weight_data <- weight_data |>
  dplyr::select(-c("weight"))
```

2.5 Merge updated weight

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

cohort_A_data <- cohort_A_data |>
  dplyr::inner_join(weight_data,
                   by = join_specification,
                   unmatched = "error",
                   relationship = "one-to-one") |>
  dplyr::mutate(
    `weight` = .data[["updated_weight"]]
  ) |>
  dplyr::select(-c("updated_weight"))
```

2.6 Check corrections

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

- `weight` changed from 215.4kg to 90.1kg for patient `A018`

```
cohort_A_data |>
  # Check if these patient IDs are present
  pointblank::expect_col_vals_make_subset(
    columns = c("cohort_unique_id"),
    set = c("A018")
  ) |>
  pointblank::expect_col_vals_expr(
    expr = pointblank::expr(
      dplyr::case_when(
        .data[["cohort_unique_id"]] %in% "A018" ~
          isTRUE(all.equal(
            target = 90.1,
            current =
             ↪  cohort_A_data[["weight"]][which(cohort_A_data[["cohort_unique_id"]]
             ↪  == "A018")],
            tolerance = 0.0001)
          ))
    )
  )
```

2.7 Write Preprocessed File

We output data to be used for the next session.

```
cohort_A_data |>
  fst::write_fst(
    path = here::here(params$analysis_folder,
                      params$harmonisation_folder,
                      params$preprocessing_folder,
                      "01_Cohort_A_cleaned.fst")
)
```

3 Extract Demographic

3.1 Read Preprocessed File

We read output data from the previous section.

3.2 Check for integer vector

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

```r
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

```r
age_gender_data <- cohort_A_data |>
  dplyr::select(c("cohort_unique_id",
                  "age",
                  "sex")) |>
  pointblank::col_vals_expr(
    expr = ~ harmonisation::is_integer_vector(
      cohort_A_data[["age"]],
      allow_na = TRUE)
  ) |>
  dplyr::mutate(
    # Convert age to type integer
    age_years = as.integer(.data[["age"]]),
    # Convert age of 0 to NA
    age_years = dplyr::case_when(
      .data[["age_years"]] == 0 ~ NA_integer_,
      .default = .data[["age_years"]]
    ),
    sex_before = .data[["sex"]],
    # Convert categorical columns to factors
    sex = dplyr::case_when(
      .data[["sex_before"]] == "F" ~ "0",
      .data[["sex_before"]] == "M" ~ "1",
      .default = as.character(.data[["sex_before"]])
    ),
    `sex_before` = forcats::fct_relevel(
      .data[["sex_before"]],
      c("F", "M")
    ),
    sex = forcats::fct_relevel(
      .data[["sex"]],
      c("0", "1")),
  ) |>
  dplyr::relocate(
    "sex",
    .after = "sex_before"
  ) |>
  pointblank::col_vals_in_set(
    columns = "sex",
    set = c("0", "1")
  ) |>
  pointblank::col_vals_between(
```

```
    columns = "age_years",
    left = 0,
    right = 100,
    inclusive = c(FALSE, TRUE),
    na_pass = TRUE
  )
```

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

cohort_unique_id	age	age_years	sex_before	sex
A001	26	26	M	1
A002	55	55	F	0

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

```
age_gender_data <- age_gender_data |>
  dplyr::select(-c("age", "sex_before"))
```

3.3.2 Height, Weight, BMI and BSA

`height_cm` will be mapped from the column `height`. `weight_kg` will be mapped from the column `weight`.

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

All values are then converted to two decimal places.

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

```
body_measurement_data <- cohort_A_data |>
  dplyr::select(c("cohort_unique_id",
                  "weight", "height")) |>
  dplyr::mutate(
    height_cm = .data[["height"]],
    weight_kg = .data[["weight"]],
    bsa_m2 = sqrt((.data[["height_cm"]] * .data[["weight_kg"]]) / 3600),
```

```
    bsa_m2 = harmonisation::round_to_nearest_digit(.data[["bsa_m2"]],
     ↪  digits = 2),
    bmi = .data[["weight_kg"]] / ((.data[["height_cm"]] / 100)^2),
    bmi = harmonisation::round_to_nearest_digit(.data[["bmi"]], digits =
     ↪  2),
    height_cm = harmonisation::round_to_nearest_digit(.data[["height_cm"]],
     ↪  digits = 2),
    weight_kg = harmonisation::round_to_nearest_digit(.data[["weight_kg"]],
     ↪  digits = 2)
  ) |>
  pointblank::col_vals_gt(
    columns = "bmi",
    preconditions = ~ . %>%
      dplyr::filter(
        .data[["cohort_unique_id"]] %in% c("A010")
      ),
    value = 50,
    na_pass = TRUE
  ) |>
  pointblank::col_vals_between(
    columns = "bmi",
    preconditions = ~ . %>%
      dplyr::filter(
        !.data[["cohort_unique_id"]] %in% c("A010")
      ),
    left = 10,
    right = 50,
    inclusive = c(TRUE, TRUE),
    na_pass = TRUE
  )
```

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

cohort_unique_id	weight	height	height_cm	weight_kg	bsa_m2	bmi
A001	75.8	177.8	177.8	75.8	1.93	23.98
A002	57.2	166.2	166.2	57.2	1.63	20.71
A003	57.6	169.1	169.1	57.6	1.64	20.14
A004	129.3	186.4	186.4	129.3	2.59	37.21
A005	86.1	186.0	186.0	86.1	2.11	24.89

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

```
body_measurement_data <- body_measurement_data |>
  dplyr::select(-c("height", "weight"))
```

3.3.3 Smoking History

`smoke_current` will be mapped from the column `smoke_current_good`. `smoke_past` will be mapped from the column `smoke_past_good`.

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

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

```
smoking_data <- cohort_A_data |>
  dplyr::select(c("cohort_unique_id",
                  "smoke_current_good", "smoke_past_good")) |>
  dplyr::mutate(
    smoke_current = as.character(.data[["smoke_current_good"]]),
    smoke_current_good = forcats::fct_relevel(
      as.character(.data[["smoke_current_good"]]),
      c("0", "1")),
    smoke_current = forcats::fct_relevel(
      .data[["smoke_current"]],
      c("0", "1")),
    smoke_past = as.character(.data[["smoke_past_good"]]),
    smoke_past_good = forcats::fct_relevel(
      as.character(.data[["smoke_past_good"]]),
      c("0", "1")),
    smoke_past = forcats::fct_relevel(
      .data[["smoke_past"]],
      c("0", "1")),
  ) |>
  pointblank::col_vals_in_set(
    columns = c("smoke_current", "smoke_past"),
    set = c("0", "1", "-1")
  ) |>
  pointblank::col_vals_expr(
    expr = pointblank::expr(
      (.data[["smoke_current"]] == "1" & .data[["smoke_past"]] == "0") |
      (.data[["smoke_current"]] == "-1" & .data[["smoke_past"]] == -"1") |
      (.data[["smoke_current"]] == "0" & .data[["smoke_past"]] %in% c("0",
↪   "1"))
```

```
    )
  )
```

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

cohort_unique_id	smoke_current_good	smoke_past_good	smoke_current	smoke_past
A001	0	0	0	0
A002	1	0	1	0
A004	0	1	0	1
A010	1	1	1	1
A016	-1	-1	-1	-1

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

```
smoking_data <- smoking_data |>
  dplyr::select(-c("smoke_current_good", "smoke_past_good"))
```

Here is a case when the validation has failed.

```
smoking_data_bad <- cohort_A_data |>
  dplyr::select(c("cohort_unique_id",
                  "smoke_current_bad", "smoke_past_bad")) |>
  dplyr::filter(
      .data[["cohort_unique_id"]] %in% c("A010", "A016")
  ) |>
  dplyr::mutate(
    smoke_current = as.character(.data[["smoke_current_bad"]]),
    smoke_past = as.character(.data[["smoke_past_bad"]]),
  )

smoking_data_bad |>
  pointblank::col_vals_in_set(
    columns = c("smoke_current", "smoke_past"),
    set = c("0", "1")
  ) |>
  pointblank::col_vals_expr(
```

```
    expr = pointblank::expr(
      (.data[["smoke_current"]] == "1" & .data[["smoke_past"]] == "0") |
      (.data[["smoke_current"]] == "-1" & .data[["smoke_past"]] == "-1") |
      (.data[["smoke_current"]] == "0" & .data[["smoke_past"]] %in% c("0",
 ↪  "1"))
    )
  )
```

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

3.3.4 Chest Pain

3.3.4.1 Shortness of Breath

`have_sob` values remained unchanged.

```
shortness_of_breath_data <- cohort_A_data |>
  dplyr::select(c("cohort_unique_id", "have_sob")) |>
  dplyr::mutate(
    have_sob = forcats::fct_relevel(
      as.character(.data[["have_sob"]]),
      c("0", "1"))
  ) |>
  pointblank::col_vals_in_set(
    columns = c("have_sob"),
    set = c("0", "1", "-1")
  )
```

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

cohort_unique_id	have_sob
A001	0
A002	1

3.3.4.2 Have chest pain or not

`have_chest_pain` is grouped as follows:

chest_pain_type	have_chest_pain
0	0
1, 2 or 3	1

```r
have_chest_pain_data <- cohort_A_data |>
  dplyr::select(c("cohort_unique_id", "chest_pain_type")) |>
  dplyr::mutate(
    have_chest_pain = dplyr::case_when(
      .data[["chest_pain_type"]] %in% c(0) ~ "0",
      .data[["chest_pain_type"]] %in% c(1, 2, 3) ~ "1",
      .default = NA_character_
    ),
    have_chest_pain = forcats::fct_relevel(
      .data[["have_chest_pain"]],
      c("0", "1")
    ),
    chest_pain_type = forcats::fct_relevel(
      as.character(.data[["chest_pain_type"]]),
      c("0", "1", "2", "3", "4", "-1")
    )
  ) |>
  pointblank::col_vals_in_set(
    columns = c("have_chest_pain"),
    set = c("0", "1")
  )
```

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

cohort_unique_id	chest_pain_type	have_chest_pain
A001	1	1
A002	0	0
A007	2	1
A012	3	1

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

```r
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

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

```
      c("0", "1", "2"))
  ) |>
  pointblank::col_vals_in_set(
    columns = c("symptoms"),
    set = c("0", "1", "2")
  )
```

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

cohort_unique_id	have_sob	have_chest_pain	symptoms
A001	0	1	1
A002	1	0	2
A003	0	0	0
A006	1	1	1

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

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

3.3.4.4 Chest Pain Type

`chest_pain_type` is grouped as follows:

have_sob	chest_pain_type before	chest_pain_type
-1	-1	-1
0	0	0
0 or 1	1	1
0 or 1	2	2
0 or 1	3	2
1	0	4

```
chest_pain_type_data <- cohort_A_data |>
  dplyr::select(c("cohort_unique_id", "chest_pain_type")) |>
  dplyr::inner_join(shortness_of_breath_data,
```

```
                   by = dplyr::join_by("cohort_unique_id"),
                   unmatched = "error",
                   relationship = "one-to-one") |>
  dplyr::mutate(
    chest_pain_type_before = .data[["chest_pain_type"]],
    chest_pain_type = dplyr::case_when(
      (.data[["chest_pain_type_before"]] == "-1" &
       .data[["have_sob"]] == "-1"
      ) ~ "-1",
      (.data[["chest_pain_type_before"]] == "0" &
       .data[["have_sob"]] == "0"
      ) ~ "0",
      (.data[["chest_pain_type_before"]] == "1" &
       .data[["have_sob"]] %in% c("0", "1")
      ) ~ "1",
      (.data[["chest_pain_type_before"]] == "2" &
       .data[["have_sob"]] %in% c("0", "1")
      ) ~ "2",
      (.data[["chest_pain_type_before"]] == "3" &
       .data[["have_sob"]] %in% c("0", "1")
      ) ~ "3",
      (.data[["chest_pain_type_before"]] == "0" &
       .data[["have_sob"]] == "1"
      ) ~ "4",
      .default = NA_character_
    ),
    chest_pain_type_before = forcats::fct_relevel(
      as.character(.data[["chest_pain_type_before"]]),
      c("0", "1", "2", "3")),
    chest_pain_type = forcats::fct_relevel(
      .data[["chest_pain_type"]],
      c("0", "1", "2", "3"))
  ) |>
  dplyr::relocate(
    "chest_pain_type_before",
    .before = "chest_pain_type"
  ) |>
  pointblank::col_vals_in_set(
    columns = c("chest_pain_type"),
    set = c("0", "1", "2", "3", "4")
  )
```

```
if (params$show_table) {
  chest_pain_type_data |>
    dplyr::distinct(.data[["have_sob"]],
```

```
                    .data[["chest_pain_type_before"]],
                    .keep_all = TRUE) |>
    knitr::kable()
}
```

cohort_unique_id	chest_pain_type_before	chest_pain_type	have_sob
A001	1	1	0
A002	0	4	1
A003	0	0	0
A006	1	1	1
A007	2	2	0
A012	3	3	0

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

```
chest_pain_type_data <- chest_pain_type_data |>
  dplyr::select(-c("have_sob", "chest_pain_type_before"))
```

3.3.4.5 Combined chest pain related tables

We combine all chest related tables together

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

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

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

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

3.3.5 Combine Demographics

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

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

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


testthat::expect_true(
  pointblank::has_columns(
    demo_behave_data,
```

```
    columns = c(
      "age_years", "sex",
      "height_cm", "weight_kg", "bsa_m2", "bmi",
      "smoke_current", "smoke_past",
      "have_sob", "have_chest_pain",
      "symptoms", "chest_pain_type"
    )
  )
)

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

3.4 Write Preprocessed File

We output data to be used for the next session.

```
demo_behave_data |>
  fst::write_fst(
    path = here::here(
      params$analysis_folder,
      params$harmonisation_folder,
      params$preprocessing_folder,
      "02_demographic_data.fst"),
)
```

4 Export To Excel

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

4.1 Read all tabular data

We read all tabular data from the previous section.

4.2 Export Data as Excel

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

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

sheet_name = c("demographics")

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

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

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

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