

Methodology: analytical pipeline

Metadata

Title: BY-COVID WP5.2 Baseline Use Case: SARS-CoV-2 vaccine effectiveness - analytical pipeline

Authors: Marjan Meurisse, Javier González-Galindo, Francisco Estupiñán-Romero, Santiago Royo-Sierra, Nina Van Goethem, Enrique Bernal-Delgado

Output: 1_DQA.html, 2_validation.html, 3_imputation.html, 4_matching.html, 5_descriptive.html, 6_survival-analysis.html, results-survival-analysis-<country>.xlsx

Folder structure:

```
analytical-pipeline/  
- input/  
- output/  
- scripts/  
- logs/  
- documentation/  
- analytical-pipeline.Rproj
```

Overview

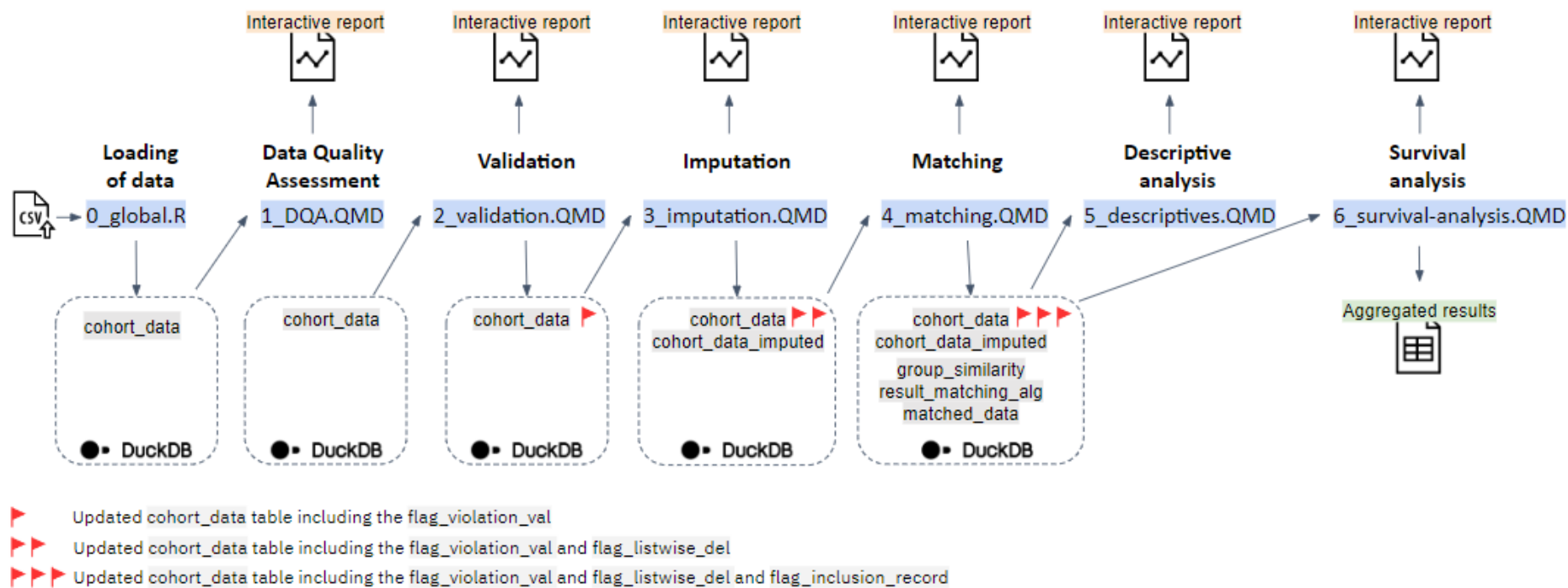


Figure 1. Overview of the different scripts used in the analytical pipeline and output generated

General settings and loading of data

Script: 0_global.R

Input: csv file (uploaded)

Intermediate output: BY-COVID-WP5-BaselineUseCase-VE.duckdb (database),
cohort_data (database table in BY-COVID-WP5-BaselineUseCase-VE.duckdb)

Description:

A DuckDB database file is created (BY-COVID-WP5-BaselineUseCase-VE.duckdb). Data are imported from a csv file using the R package Arrow and inserted into the cohort_data database table within the BY-COVID-WP5-BaselineUseCase-VE.duckdb. Data types are manually specified according to the Common Data Model specification (1) when reading the data using a schema.

Data quality assessment

Script: 1_DQA.QMD

Input: cohort_data (database table in BY-COVID-WP5-BaselineUseCase-VE.duckdb)

Output pipeline: 1_DQA.html (report)

Description:

A Data Quality Assessment (DQA) on the cohort_data is performed and an interactive html report (1_DQA.html) is created. This report provides an overview of the data and includes dataset statistics, variable types, missing data profiles and potential alerts.

Validation

Script: 2_validation.QMD

Input: cohort_data (database table in BY-COVID-WP5-BaselineUseCase-VE.duckdb)

Output pipeline: 2_validation.html (report)

Intermediate output: cohort_data including the variable flag_violation_val (database table in BY-COVID-WP5-BaselineUseCase-VE.duckdb)

Description:

In order to ensure interoperability (i.e., allowing to reproduce the same analyses in every node) the use case requires that the data from each of the nodes complies with the Common Data Model specification (1). As such, the imported data must comply with a number of pre-specified validation rules (see Table 1). The cohort_data are tested against this set of validation rules and the results of this validation process are summarised in an interactive html report (2_validation.html). These validation rules are considered 'essential' not to be violated in order for the record to be used for the subsequent analysis. A logical variable flag_violation_val is created in the cohort_data table in the

BY-COVID-WP5-BaselineUseCase-VE.duckdb DuckDB database and set to TRUE for a record when at least one of the validation rules in the pre-specified set (Table 1) is violated (otherwise this variable is set to FALSE).

Table 1. Set of pre-specified validation rules, testing compliance with the Common Data Model specification.

is.na(age_nm) age_nm >= 5 & age_nm <=115
is.na(sex_cd) sex_cd %in% c("0","1","2","9")
is.na(dose_1_brand_cd) dose_1_brand_cd %in% c("BP","MD","JJ","AZ","NV")
is.na(dose_2_brand_cd) dose_2_brand_cd %in% c("BP","MD","JJ","AZ","NV")
is.na(number_doses) number_doses >= 0 & number_doses <= 10
fully_vaccinated_bl==FALSE fully_vaccinated_bl==TRUE & !is.na(vaccination_schedule_cd)
is.na(test_type_cd) test_type_cd %in% c("PCR","AG","other")
is.na(variant_cd) variant_cd %in% c("alpha","beta","gamma","delta","omicron","epsilon","zeta","eta","theta","iota","kappa","lambda","mu"),
is.na(pregnancy_bl) pregnancy_bl==FALSE (pregnancy_bl==TRUE & sex_cd==2 & age_nm>=12 & age_nm<=55)
is.na(essential_worker_bl) essential_worker_bl==FALSE (essential_worker_bl==TRUE & age_nm>=16 & age_nm<=70)
(is.na(dose_1_dt) & is.na(dose_2_dt)) is.na(dose_2_dt) !is.na(dose_1_dt) & !is.na(dose_2_dt) & (dose_1_dt < dose_2_dt)
(is.na(dose_2_dt) & is.na(dose_3_dt)) is.na(dose_3_dt) !is.na(dose_2_dt) & !is.na(dose_3_dt) & (dose_2_dt < dose_3_dt)
is.na(previous_infection_dt) is.na(confirmed_case_dt) !is.na(previous_infection_dt) & !is.na(confirmed_case_dt) & (previous_infection_dt < confirmed_case_dt)
is.na(confirmed_case_dt) is.na(exitus_dt) !is.na(confirmed_case_dt) & !is.na(exitus_dt) & (confirmed_case_dt <= exitus_dt)
is.na(previous_infection_dt) is.na(exitus_dt) !is.na(previous_infection_dt) & !is.na(exitus_dt) & (previous_infection_dt <= exitus_dt)
is.na(fully_vaccinated_dt) is.na(exitus_dt) !is.na(fully_vaccinated_dt) & !is.na(exitus_dt) & fully_vaccinated_dt <= exitus_dt
(!is.na(dose_1_dt) & !is.na(dose_2_dt) & !is.na(dose_3_dt) & number_doses>=3) (!is.na(dose_1_dt) & !is.na(dose_2_dt) & is.na(dose_3_dt) & number_doses==2) (!is.na(dose_1_dt) & is.na(dose_2_dt) & is.na(dose_3_dt) & number_doses==1) (is.na(dose_1_dt) & is.na(dose_2_dt) & is.na(dose_3_dt) & number_doses==0)

<code>is.na(dose_1_dt) (is.na(dose_1_dt) & !is.na(dose_1_brand_cd))</code>
<code>is.na(dose_2_dt) (is.na(dose_2_dt) & !is.na(dose_2_brand_cd) & is.na(dose_1_dt) & is.na(dose_1_brand_cd))</code>
<code>is.na(dose_3_dt) (is.na(dose_3_dt) & !is.na(dose_3_brand_cd) & is.na(dose_2_dt) & is.na(dose_2_brand_cd) & !is.na(dose_1_dt) & is.na(dose_1_brand_cd))</code>
<code>(dose_1_brand_cd=="JJ" & is.na(dose_1_dt) & !is.na(fully_vaccinated_dt) & fully_vaccinated_bl==TRUE) (dose_1_brand_cd!="JJ" & is.na(dose_2_dt) & !is.na(fully_vaccinated_dt) & fully_vaccinated_bl==TRUE) (is.na(dose_1_brand_cd) & is.na(fully_vaccinated_dt) & fully_vaccinated_bl==FALSE) (dose_1_brand_cd!="JJ" & is.na(dose_2_dt) & is.na(fully_vaccinated_dt) & fully_vaccinated_bl==FALSE)</code>

Imputation of missing data

Script: `3_imputation.QMD`

Input: cohort_data including the flag_violation_val (database table in BY-COVID-WP5-BaselineUseCase-VE.duckdb)

Output pipeline: `3_imputation.html` (report)

Intermediate output: cohort_data including the flag_violation_val and flag_listwise_del, cohort_data_imputed and imputation_methods (database tables in BY-COVID-WP5-BaselineUseCase-VE.duckdb)

Description - concept:

A decision tree was constructed to guide the decisions on how to handle missing data (see Figure 2).

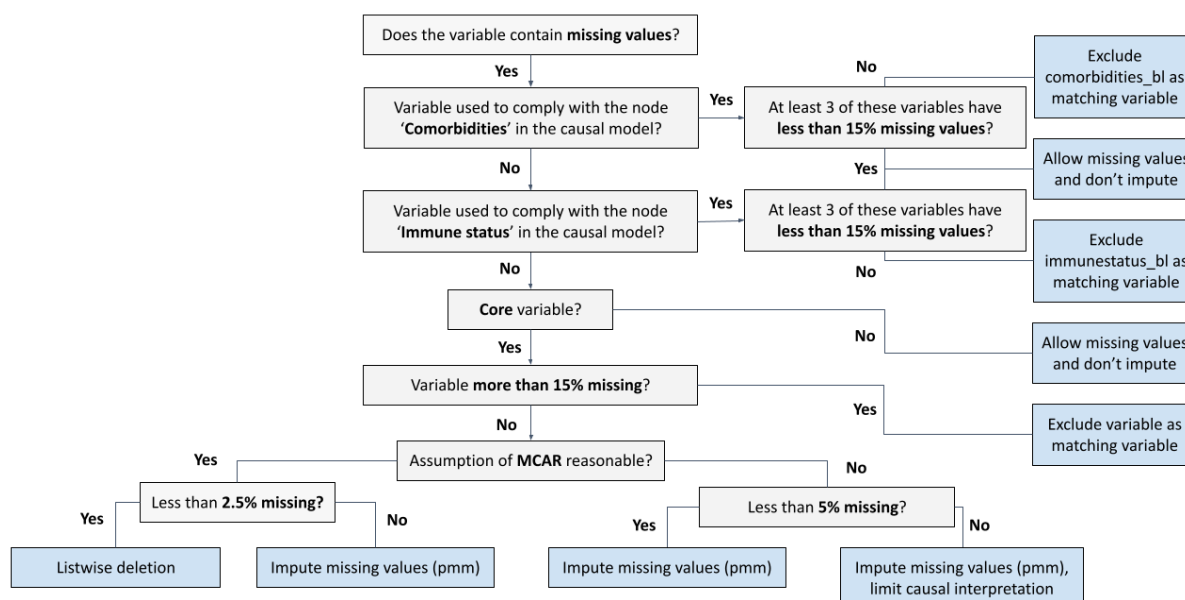


Figure 2. Decision tree for the imputation of missing data according to the variable. MCAR: missing completely at random.

For each variable, different checks are conducted, based on which a decision is made on how to handle missing data:

- *Does the variable contain any missing values?*
→ If not, no imputation of missing values nor listwise deletion is performed.
- *Is the variable used to comply with the node 'Comorbidities' in the causal model (i.e., used to compute comorbidities_bl)?*
→ If yes and at least three of the variables used to comply with the node 'Comorbidities' in the causal model have less than 15% missing values, missing values are allowed and no imputation nor listwise deletion is performed.
→ If yes and less than three of the variables used to comply with the node 'Comorbidities' in the causal model have less than 15% missing values, comorbidities_bl is excluded as matching variable.
- *Is the variable used to comply with the node 'Immune status' in the causal model (i.e., used to compute immunestatus_bl)?*
→ If yes and at least three of the variables used to comply with the node 'Immune status' in the causal model have less than 15% missing values, missing values are allowed and no imputation nor listwise deletion is performed.
→ If yes and less than three of the variables used to comply with the node 'Immune status' in the causal model have less than 15% missing values, immunestatus_bl is excluded as matching variable.
- *Is the variable considered as a 'core' variable?*
→ If not, missing values are allowed and no imputation of missing values nor listwise deletion is performed. A set of 'core' variables have been identified: age_nm, sex_cd, residence_area_cd, pregnancy_bl, essential_worker_bl, institutionalized_bl, foreign_bl and socecon_lvl_cd. Missing values in these variables will raise issues in the subsequent analyses, such as not being able to apply the exclusion criteria or obstructing the matching process.
- *Does the variable contain more than 15% missing values?*
→ If yes, the variable is excluded as a matching variable.
- *Is it reasonable to assume 'missingness completely at random' (MCAR) in the subset of variables used to predict missing values (core variables plus variables used to comply with the node 'Comorbidities' and 'Immune status' in the causal model, with less than 15% missing values)?* We aim to assess the mechanisms behind the missing values in these variables of the dataset (i.e., in the subset of the dataset used to predict missing values). MCAR assumes the independence of missingness of the data of both observed and unobserved data, and the assumption of MCAR can be tested based on the observed data only (2). Reasonability of MCAR is tested using Little's test (2).
→ If yes (i.e. when MCAR is reasonable):
Does the variable contain less than 2.5% missing values?

- If yes, listwise deletion.
- If no, the missing values are imputed using the predictive mean matching (pmm) approach, to prevent losing a large amount of records.
- If not (i.e. when MCAR is not reasonable):
 - Does the variable contain less than 5% missing values?*
 - If yes, the missing values are imputed using the predictive mean matching (pmm) approach.
 - If not (between 5-15% missing), the missing values are imputed using the predictive mean matching (pmm) approach (see simulations of Collins et al. (3), where multiple imputation estimates were remarkably robust against MNAR in many instances), however, obtained estimates should not be interpreted causally.

Description - implementation:

Only a subset of the `cohort_data` table, compliant with the common data model specification (`flag_violation_val==FALSE`), is considered for imputation through a temporary view of a query `cohort_view`.

A set of core variables (`core_v`) is specified, as well as the variables used for the computation of the variables `comorbidites_bl` and `immunestatus_bl` (`comorb_v` and `imm_v`).

A table is created (`df_var_imputation_method`) containing for each variable in the subset (created by `cohort_view`) information on the different checks in the decision tree (see Figure 2), i.e., does the variable contain missing values (`Missing_values`, TRUE/FALSE), does the variable contain less than 2.5% missing values (`Perc_missing_lt`, TRUE/FALSE), does the variable contain less than 5% missing values (`Perc_missing_lt5`, TRUE/FALSE), does the variable contain less than 15% missing values (`Perc_missing_lt15`, TRUE/FALSE), is the variable a core variable (`Core`, TRUE/FALSE), is the variable used for the computation of `comorbidites_bl` (`Comorbidity`, TRUE/FALSE), is the variable used for the computation of `immunestatus_bl` (`Immunestatus`, TRUE/FALSE), do more than three of the variables used to comply with the node 'Comorbidities' in the causal model have less than 15% missing values (`comorb_incl`, TRUE/FALSE), and do more than three of the variables used to comply with the node 'Immune status' in the causal model have less than 15% missing values (`imm_incl`, TRUE/FALSE). Further, for the subset of `cohort_view` containing variables used to predict missing values (core variables plus variables used to comply with the node 'Comorbidities' and 'Immune status' in the causal model, with less than 15% missing values), the MCAR assumption was tested using the `mcar_test` function (Little's test statistic) of the R package `nanian` (4).

Based on this information, for each variable it was decided whether imputation will be performed (`imputation_method` with possibilities: 'No missing values', 'Allow missing values and don't impute missing values (comorbidity)', 'Allow missing values and don't

impute missing values (immune status)', 'Allow missing values and don't impute missing values (no core variable)', 'Listwise deletion where core variable has missing values (MCAR reasonable)', 'Imputation of missing values (MCAR reasonable)', 'Imputation of missing values (not MCAR)', 'Imputation of missing values (not MCAR, limit causal interpretation)', 'Exclude comorbidities_bl as matching variable (comorbidity)', 'Exclude immunestatus_bl as matching variable (immune status)', 'Exclude core variable as matching variable (more than 15% missing values)'. A report (3_imputation.html) is generated summarising the results of the different checks and methods used for dealing with missing values. Further, the table with variable characteristics (i.e., per variable the checks in the decision tree and the methods used for dealing with missing values) is saved as the DuckDB database table `imputation_methods`.

A logical variable `flag_listwise_del` is created in the `cohort_data` table in the BY-COVID-WP5-BaselineUseCase-VE.duckdb DuckDB database. For variables for which it was decided to not use records for which the value of this variable is missing for further analysis (`imputation_method=='Listwise deletion where core variable has missing values (MCAR reasonable)'`), the `flag_listwise_del` is set to TRUE for records/patients with a missing value for this variable in the subset of the `cohort_data` table (created by `cohort_view`).

If there are variables for which it was decided to impute missing values (`imputation_method=='Imputation of missing values (MCAR reasonable)'` or `imputation_method=='Imputation of missing values (not MCAR)'` or `imputation_method=='Imputation of missing values (not MCAR, limit causal interpretation)'`), single-imputation (SI) of missing values was performed in the subset of the `cohort_data` table with the pmm approach using the R package `mice` (`m=1`) (5). The other 'core' variables plus variables used to comply with the node 'Comorbidities' and 'Immune status' in the causal model with less than 15% missing values, are used as predictors for the incomplete 'core' variables. This results in an imputed dataset. From this imputed dataset, only records with at least one value imputed are filtered and saved in a separate database table `cohort_data_imputed` in the BY-COVID-WP5-BaselineUseCase-VE.duckdb DuckDB database. The tables `cohort_data` and `cohort_data_imputed` are used in parallel in further analyses. The R function `coalesce` (returning the first non-NULL evaluated expression) is used when a core variable is used. In this way, when a value is missing for a patient in a core variable of `cohort_data` (`coalesce` evaluates NULL value for first expression), the value of this variable for this patient will be obtained from `cohort_data_imputed` (second expression).

Matching

Scripts: `4_matching.QMD` (sourcing `4_matching.R`)

Input: cohort_data including the flag_violation_val and flag_listwise_del, cohort_data_imputed and imputation_methods (database tables in BY-COVID-WP5-BaselineUseCase-VE.duckdb)

Output pipeline: 4_matching.html (report)

Intermediate output: group_similarity, result_matching_alg, and matched_data (database tables in BY-COVID-WP5-BaselineUseCase-VE.duckdb)

Description - concept:

In the sequential emulated target trial (see study protocol (6)), a sequence of nested (*daily*) trials are emulated with increasing time (t_1, t_2, \dots, t_n), iterating over the days in the enrollment period. At each eligible time during the enrollment period, the vaccination status of eligible individuals is assessed and every individual who has completed a primary vaccination schedule at that time (*treated/exposed*) is matched to an individual who has not (yet) completed the primary vaccination schedule (*control*). Newly vaccinated individuals (completing a primary vaccination schedule) are eligible for inclusion in the study, even if they had previously been selected in the “no (or partial) vaccine group”. Follow-up ends at diagnosis of SARS-CoV-2 infection, death, completed primary vaccination (for unvaccinated or partially vaccinated controls), completed primary vaccination of the matched control (for primary vaccinated persons), booster dose (for primary vaccinated persons), booster dose of the matched vaccinated person (for unvaccinated or partially vaccinated controls), or the end of the study period (i.e., the most recent date at which data is available at time of analysis).

Description - implementation:

Based on the table imputation_methods in the BY-COVID-WP5-BaselineUseCase-VE.duckdb DuckDB database, a vector of variables considered for matching (i.e., variables selected from sex_cd, age_cd, residence_area_cd, pregnancy_bl, essential_worker_bl, institutionalized_bl, foreign_bl, soecon_lvl_cd, comorbidities_bl, immunestatus_bl based on missingness) is constructed (v_matching_incl).

New variables are created in the cohort_data table in the BY-COVID-WP5-BaselineUseCase-VE.duckdb DuckDB database and are calculated based on the variables that are already present in the data: comorbidities_bl, immunestatus_bl, age_cd, boost_bl, and flag_inclusion_record (see Table 2). The variable age_cd, which is considered as a ‘core variable’ and missing values might therefore have been imputed in age_nm (see decision tree, Figure 2), is also generated in cohort_data_imputed.

Table 2. Calculation of variables

Variable	Calculation
comorbidities_bl	CASE WHEN diabetes_bl OR obesity_bl OR heart_failure_bl OR copd_bl OR

	solid_tumor_without_metastasis_bl OR chronic_liver_disease_bl OR chronic_kidney_disease_bl OR sickle_cell_disease_bl OR hypertension_bl THEN TRUE ELSE FALSE
immune_status_bl	CASE WHEN blood_cancer_bl OR transplanted_bl OR hiv_infection_bl OR primary_immunodeficiency_bl OR immunosuppression_bl THEN TRUE ELSE FALSE
age_cd	CASE WHEN age_nm >= 0 and age_nm <=4 THEN 1 WHEN age_nm >= 5 and age_nm <=9 THEN 2 WHEN age_nm >= 10 and age_nm <=14 THEN 3 WHEN age_nm >= 15 and age_nm <=19 THEN 4 WHEN age_nm >= 20 and age_nm <=24 THEN 5 WHEN age_nm >= 25 and age_nm <=29 THEN 6 WHEN age_nm >= 30 and age_nm <=34 THEN 7 WHEN age_nm >= 35 and age_nm <=39 THEN 8 WHEN age_nm >= 40 and age_nm <=44 THEN 9 WHEN age_nm >= 45 and age_nm <=49 THEN 10 WHEN age_nm >= 50 and age_nm <=54 THEN 11 WHEN age_nm >= 55 and age_nm <=59 THEN 12 WHEN age_nm >= 60 and age_nm <=64 THEN 13 WHEN age_nm >= 65 and age_nm <=69 THEN 14 WHEN age_nm >= 70 and age_nm <=74 THEN 15 WHEN age_nm >= 75 and age_nm <=79 THEN 16 WHEN age_nm >= 80 and age_nm <=84 THEN 17 WHEN age_nm >= 85 THEN 18 ELSE NULL
boost_bl	CASE WHEN vaccination_schedule_cd == 'JJ' AND dose_2_dt IS NOT NULL THEN TRUE WHEN vaccination_schedule_cd != 'JJ' AND vaccination_schedule_cd IS NOT NULL AND dose_3_dt IS NOT NULL THEN TRUE ELSE FALSE
flag_inclusion_rec ord	CASE WHEN previous_infection_bl==TRUE OR flag_violating_val==TRUE OR flag_listwise_del==TRUE THEN FALSE ELSE TRUE

For each combination of the variables considered for matching (i.e., variables listed in `v_matching_incl`), a `group_id` is created and a `group_id` is assigned to each patient based on these variables.

Records with the `flag_inclusion_records` equal to `FALSE` (i.e., records from individuals with a previous infection, records violating one of the ‘essential’ validation rules and/or records set to be listwise deleted) are not considered for the further analyses. A view `cohort_view` is created only selecting those records with `flag_inclusion_records==TRUE`.

A list of unique dates (`dates_v`) is extracted at which newly vaccinated (i.e., completing their primary vaccination schedule) individuals are identified during the enrollment period (i.e.

between 1 January 2021 and 1 September 2021) (using the self-produced `getDates` function). Subsequently, the matching algorithm will iterate over these selected dates (i.e., sequential nested trials).

A data frame (`df_original`) is generated (using the self-produced `calculate_similarity` function) with unique combinations of the variables considered for matching (i.e., variables listed in `v_matching_incl`) and the corresponding `group_id`. For each group, the 10 groups with the 'nearest' distance based on these variables are matched (matching method: 'nearest', nearest neighbour matching on the propensity score; distance: 'glm', logistic regression propensity score) using the R package 'MatchIt'. A new table, `group_similarity`, is created in the `BY-COVID-WP5-BaselineUseCase-VE.duckdb` DuckDB database containing for each `group_id` the 10 nearest matched groups and corresponding distances.

Next, we start with the iterative matching process (using the self-produced `doMatch` function), iterating over dates in `dates_v` (see Figure 3).

On each date:

1. The `group_id`'s of patients completing their primary vaccination schedule on that date are selected from the `cohort_data` table. For each of these `group_id`'s, the number of patients completing their primary vaccination schedule on that date (*intervention group*, `full_vaccine_group`) are counted (`full_vaccine_n_group`) and the number of possible controls (*control group*, `control_group_id`, WHERE `fully_vaccinated_bl`==FALSE OR later considered fully vaccinated AND previous infection date null or later than that date AND confirmed case date null or later than that date AND exitus date null or later than that date) are counted (`control_n_group`), resulting in the table `groups_by_date`.
→ function `getGroupsByDate`
2. Iteration over each of these `group_id`'s (row in `groups_by_date` table) (function `loop_group`) is performed. Depending on the number of patients in the intervention and control group for the `group_id`, a different methodology for matching is used:
 - a. *If `control_n_group` is null (no exact match found):*
For the `group_id`, the `group_id`'s of the 10 most similar groups (maximum, less when less than 10 similar groups found) based on the variables considered for matching (i.e., variables listed in `v_matching_incl`) are obtained from the `group_similarity` table in `BY-COVID-WP5-BaselineUseCase-VE.duckdb`. From patients eligible as control (WHERE `fully_vaccinated_bl` == FALSE OR later considered fully vaccinated AND previous infection date null or later than that date AND confirmed case date null or later than that date AND exitus date null or later than that date) and with `group_id` equal to one of these similar groups, up to 150 controls (maximum) are randomly selected from each similar group from the `cohort_data` table (coalesce with `cohort_data_imputed`). As such, up to 1500 controls can be selected (can be less when less than 10 similar groups found)

or when in each of the similar groups less than 150 eligible controls found at that date).

→ function `getSampleForMatch`

i. *If similar group found:*

The selected 'similar' potential controls are one-to-one matched (matching method: 'nearest', nearest neighbour matching on the propensity score; distance: 'glm', logistic regression propensity score) to the patients in the intervention group based on the variables considered for matching (i.e., variables listed in `v_matching_incl`), using the R package 'MatchIt'. The `person_id`'s of the cases (`person_id`) and the `person_id`'s of the controls (`matched_id`) are collected as different rows and a subclass for each match is generated.

ii. *If no similar group found:*

For each patient in the intervention group, 150 eligible controls are randomly selected from the `cohort_data` table (coalesce with `cohort_data_imputed`). The selected eligible controls are one-to-one matched (see matching method before) to the persons in the intervention group, selecting for each person in the intervention group the nearest control from the 150 eligible controls. The `person_id`'s of the cases (`person_id`) and the `person_id`'s of the controls (`matched_id`) are collected as different rows and a subclass for each match is generated.

b. *If control_n_group is not null (exact match(es) found):*

From the `cohort_data` (coalesce with `cohort_data_imputed`), for all cases with the `group_id`, the variables used for matching (i.e., variables listed in `v_matching_incl`), the `person_id` and `fully_vaccination_bl` are extracted. Hence, a number of rows for cases equal to `full_vaccine_n_group` are obtained. Also a number of controls equal to `control_n_group` are selected with the same information.

→ function `getSampleNBigger`

i. *If less (or equal) cases than controls:*

Since there are more controls than cases, for each case a different exact match in the control group was found and selected. The `person_id`'s of the cases (`person_id`) and the `person_id`'s of the controls (`matched_id`) are collected as different rows and a subclass for each match is generated.

ii. *If more cases than controls:*

When there are more cases than controls that can be matched exactly, each control is matched to a case. The controls (which have been matched already) are re-used to match with the remaining

cases without a match. As such, a person in the control group can be matched more than once. The `person_id`'s of the cases (`person_id`) and the `person_id`'s of the controls (`matched_id`) are collected as different rows and a subclass for each match is generated.

3. Row bind the data frame obtained for each `group_id`
4. Append the results obtained for each date to a database table `result_matching_alg` in the `BY-COVID-WP5-BaselineUseCase-VE.duckdb` DuckDB database. In this table, one record corresponds to one matched pair.

A new table, `matched_data`, is subsequently created in the `BY-COVID-WP5-BaselineUseCase-VE.duckdb` DuckDB database, with two records per match (i.e., one for the case and one for the control). A person can appear more than once in this table (e.g., once - or more - as a control and once as a case). For each record, the `fully_vaccinated_dt`, `confirmed_case_dt`, `exitus_dt` and `boost_dt` are added. Further, variables for the date of onset (`dt_onset`, for controls, NA for cases), whether or not the patient experienced the outcome during the follow-up time (`status`) and follow-up time (`futime`) are calculated.

→ function `getStatusMatch`

After matching, the covariate balance is assessed for the variables considered for matching (i.e., variables listed in `v_matching_incl`). Covariate balance is assessed by looking at the Standardised Mean Distances (SMD) and propensity score distribution before and after matching. The results of this assessment are documented in a report (`4_matching.html`).

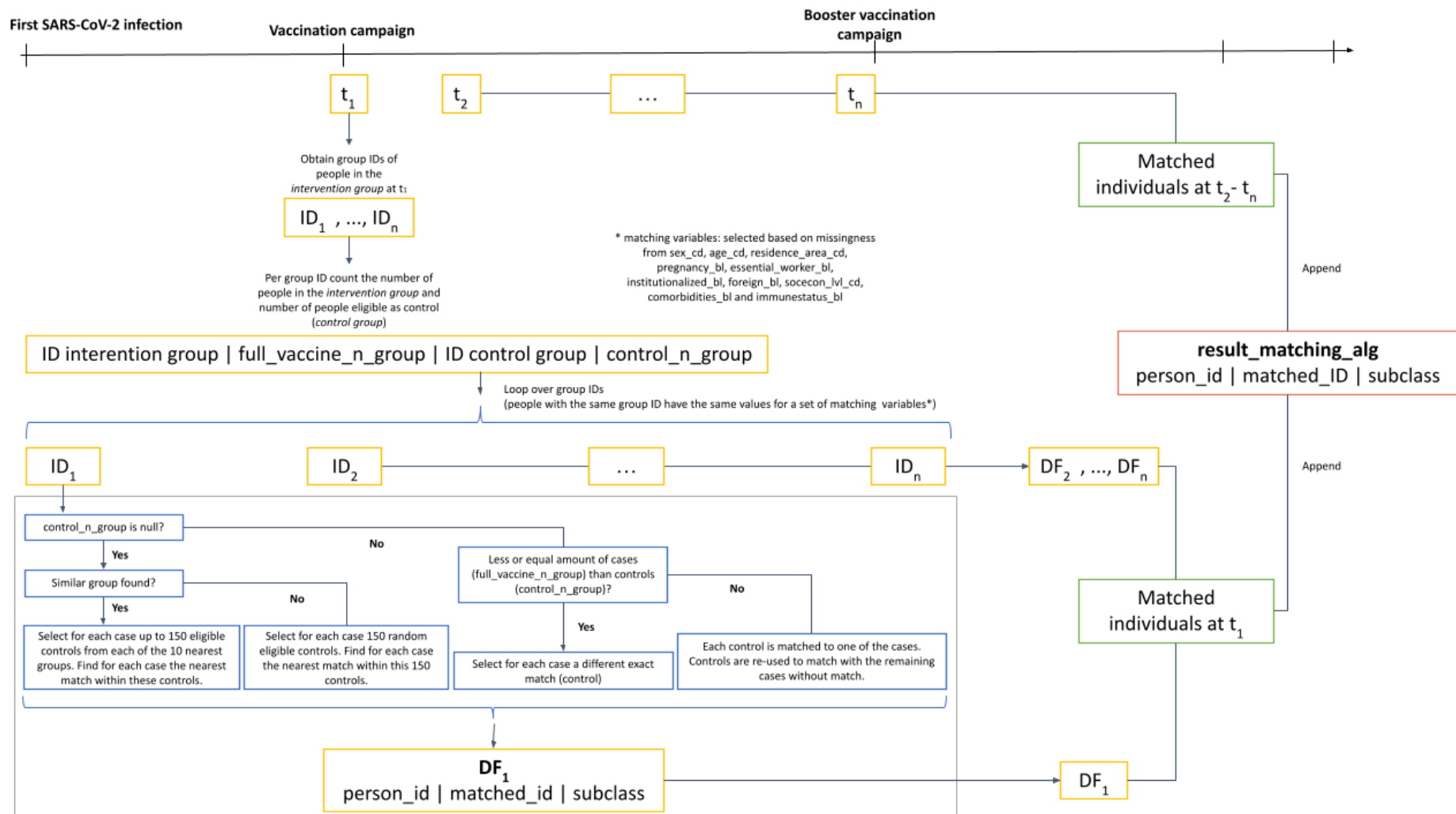


Figure 3. Visual representation of the matching algorithm

Descriptive analysis

Script: 5_descriptives.QMD

Input: cohort_data including the flag_violation_val, flag_listwise_del and flag_inclusion_record, cohort_data_imputed, imputation_methods and matched_data (database tables in BY-COVID-WP5-BaselineUseCase-VE.duckdb)

Output pipeline: 5_descriptive.html (report)

Description:

The descriptive analysis contains four elements: the considered time periods (data extraction period, enrollment period and study period), a flowchart describing the study population selection (CONSORT diagram), a descriptive table with the baseline characteristics of the matched study population by intervention group, and results of a survival analysis in the unmatched population (adjusted and unadjusted).

Survival analysis

Script: 6_survival-analysis.QMD

Input: matched_data (database table in BY-COVID-WP5-BaselineUseCase-VE.duckdb)

Output pipeline: 6_survival-analysis.html (report), results-survival-analysis-<country>.xlsx (aggregated results)

Description:

A survival analysis is conducted in the matched population (matched_data). The survival function is estimated using the Kaplan-Meier estimator and represented visually using a Kaplan-Meier curve. The survival function is estimated for the control and intervention group. Further, the probability of not obtaining a SARS-CoV-2 infection beyond a certain time after onset of follow-up (survival function, estimated using the Kaplan-Meier estimator) is reported for different periods. The median survival time is also calculated and reported (if the probability of not obtaining a SARS-CoV-2 infection dropped below 50%). A Cox regression model was built to quantify the effectiveness of completing a primary vaccination schedule in preventing SARS-CoV-2 infection. A hazard ratio (HR) is computed and reported, which can be interpreted as the instantaneous rate of SARS-CoV-2 infections in individuals that are at risk for obtaining an infection. Proportional hazards during the study period might be unlikely. As such, the Restricted Mean Survival Time (RMST) and Restricted Mean Time Lost (RMTL) ratios are additionally calculated, providing an alternative estimate for the Average Treatment Effect (ATE), without requiring the proportional hazards assumption to be met. In addition to the main analysis, survival over time, HR and ATE were estimated within subgroups determined by the vaccination schedule received by the case in the matched population and within subgroups determined by the NUTS3 residence area of individuals.

Aggregated results (survival over time, HR, RMST/RMST, and an estimated ATE) are saved in different tabs of the excel file `results-survival-analysis-<country>.xlsx`, for both the main analysis and additional analyses.

References

1. Estupiñán-Romero F, Van Goethem N, Meurisse M, González-Galindo J, Bernal-Delgado E. BY-COVID - WP5 - Baseline Use Case: SARS-CoV-2 vaccine effectiveness assessment - Common Data Model Specification. 2023 Jan 26 [cited 2023 Feb 22]; Available from: <https://zenodo.org/record/7572373>
2. Li C. Little's Test of Missing Completely at Random. *The Stata Journal*. 2013 Dec 1;13(4):795–809.
3. Collins LM, Schafer JL, Kam CM. A comparison of inclusive and restrictive strategies in modern missing data procedures. *Psychol Methods*. 2001 Dec;6(4):330–51.
4. Tierney N. `naniar` [Internet]. 2023 [cited 2023 Apr 13]. Available from: <https://github.com/njtierney/naniar>
5. Buuren S van, Groothuis-Oudshoorn K, Vink G, Schouten R, Robitzsch A, Rockenschaub P, et al. `mice`: Multivariate Imputation by Chained Equations [Internet]. 2022 [cited 2023 Apr 13]. Available from: <https://CRAN.R-project.org/package=mice>
6. Meurisse M, Van Goethem N, Estupiñán-Romero F, González-Galindo J, Royo-Sierra S, Martínez-Lizaga N, et al. BY-COVID - WP5 - Baseline Use Case: COVID-19 vaccine effectiveness assessment - Study protocol. 2023 Jan 23 [cited 2023 Jan 31]; Available from: <https://zenodo.org/record/7560731>