RECAP

"Research on European Children and Adults born Preterm"

Grant Agreement number: 733280

Deliverable 5.3

R package for harmonisation of VPT data

Workpackage:	WP 5
Task:	T 5.2
Due Date:	30st June 2019 (M30)
Actual Submission Date:	10th July 2019 (M30)
Last Amendment deliverable:	9rd July 2019
Project Dates:	Project Start Date: January 01, 2017 Project Duration: 51 months
Responsible partner:	TNO
Responsible author:	Prof. dr. Stef van Buuren (TNO)
Email:	Stef.vanBuuren@tno.nl
Contributors:	Stef van Buuren (TNO), Manon Grevinga (TNO), Leonhard Bakker (TNO), Sylvia van der Pal (TNO), Paula Raissa Silva (INESCTEC), Rui Camacho (INESCTEC)

Project funded by the European Commission within H2020-SC1-2016-2017/H2020-SC1-2016-RTD		
Dissemination Level		
PU	Public	
PP	Restricted to other programme participants (including the Commission Services)	
RE	Restricted to a group specified by the consortium (including the Commission Services)	
CO	Confidential, only for members of the consortium (including the Commission Services)	CO

Document History:

Version	Date	Changes	From	Review
V0.1		Deliverable template	Maaike Beltman	
V1.1	May 24, 2019	Deliverable 5.3 first outline	Sylvia vd Pal	Stef van Buuren
V1.2	May 29, 2019	Deliverable 5.3 CH2 background	Manon Grevinga	Stef van Buuren
V1.3	June 3, 2019	Additional topics for chapter 2.1	Sylvia vd Pal	Stef van Buuren
V1.4	June 25, 2019	Revision of main text	Stef van Buuren	Dieter Wolke
V1.5	July 3, 2019	Incorporate suggestions by Wolke	Stef van Buuren	Dieter Wolke
V1.6	July 9, 2019	Add section 4 on mitigation	Stef van Buuren	FINAL

SUMMARY

This Deliverable 5.3 (D5.3) of the RECAP project describes the WP5 "R package for harmonisation of VPT data".

This deliverable reports on the work that has been done to implement the `mice` R-package for multiple imputation for federated analysis in the `DataSHIELD` environment. `DataSHIELD` was adopted by RECAP as the platform for distributed data analysis that "moves the analyses to the data". The R package in `mice` implements multiple imputation, a technique that replaces missing data by synthetic values. The technique provides a sound scientific approach for solving complex harmonisation problems that are common to cohort data of children and adults as collected in RECAP.

The report highlights how multiple imputation can be used to solve harmonisation issues. We propose a generic approach to implement mice in a federated environment, and discuss the experiences and lessons learned thus far.

Our conclusions are:

- There are no intrinsic barriers to implement mice in a federated environment.
- It is straightforward to store imputations exclusively at the client side (so it does not require storage on the individual nodes), a possibility that is fine for imputations that have a narrow scope for specific statistical analyses.
- In the philosophy of both `mice` and `DataSHIELD`, imputations would target multiple users and multiple analyses. In that case it is more sensible to distribute the imputations over the nodes, and store them at the location of individual nodes, next to the data. This entails a use case with an interesting trade-off: *While the data never leave the node, the information in the data does, but - in return - the node gets supplemental data that is informed by the data present in other nodes*.
- The iterative algorithm to make this work is currently hard to realize in `DataSHIELD`. Some of planned extensions of `DataSHIELD`, like server-side storage will ease implementation.
- An algorithm that needs to iterate over nodes is slow, due to the substantial overhead needed in communication, data access and privacy checking. This is no fundamental bottleneck, and speed problem seems addressable by improved architecture of client-server technology.

Suggestions for further works are:

- Work out a few use cases in RECAP that demonstrate the added value of multiple imputation for solving harmonisation problems;
- Improve automation of both client-side and server-side imputation approaches.

Table of contents

1 INTRODUCTION

1.1 Purpose and Scope

This Deliverable 5.3 (D5.3) of the RECAP project describes the WP5 "R package for harmonisation of VPT data".

This deliverable reports on the work that has been done to implement the `mice` R-package for multiple imputation for federated analysis in the `DataSHIELD` environment (Wolfson, 2010). `DataSHIELD` was adopted by RECAP as the platform for distributed data analysis that "moves the analyses to the data". The R package in `mice` implements multiple imputation, a technique that replaces missing data by synthetic values. The technique provides a sound scientific approach for solving complex harmonisation problems that are common cohort data of children and adults as collected in RECAP.

1.2 References to other RECAP Documents

This document is a sequel to the conceptual framework for harmonisation as described in deliverable 5.1 (D5.1), and the Workshop for project members on conceptual framework as described in deliverable 5. 2 (D5.2). Statistical concepts of these deliverables have been implemented in the RECAP statistical analysis platform (WP4) as specified in deliverable 4.1 (D4.1), in close collaboration with the other work packages. This document also builds upon the harmonisation procedures as developed in WP3, and specified in deliverable 3.1 (D3.1).

Definitions, Abbreviations and Acronyms

Table 1 List of Abbreviations and Acronyms

Abbreviation/ Acronym	DEFINITION
VPT	Very preterm
VLBW	Very Low birth weight
IPD	Individual Patient Data
MAR	Missing at random
IPW	inverse probability weighting
MCAR	missing completely at random
MICE	Multivariate Imputation by Chained Equations

2 BACKGROUND

2.1 Harmonisation within RECAP

This deliverable, deliverable 5.3 *R package for harmonisation of VPT data*, aims to have a general `R` package, that can be used by all cohorts, to harmonise their data on the RECAP platform. On the platform, every cohort saves the cohort data on its own RECAP node, which is under control of the node administrator, typically the data owner or the database specialist associated with the cohort. The node administrator can grant access to researchers who want to use the cohort data. "Granting access" does not mean that the investigator will get a physical copy of the data, but rather allows the investigator to conduct statistical analyses on the cohort data through a remote analysis session that runs on the RECAP node (and thus not at the investigator's machine). In this way, the cohort data never needs to leave the RECAP node, while still enabling statistical analysis on the data to those who are granted access.

The scenario can be easily extended to multiple nodes. Suppose the investigator wants to combine the data from two cohorts and analyse the combined data jointly. While it is not possible to physically create a combined data set, the investigator can run a separate analysis on each node, and then combine the results from each of the analyses. There are several `R` packages that assist in combining the results of two or more analysis results. Hence, the investigator can do an analysis on the combined data without having to actually combine the data. This is known as *federated analysis*.

An important requirement for federated analysis to work is that the data are harmonised. For example, both nodes should code sex in the same way, and use the same measurement units. If data are not harmonised across nodes, the combined analysis will not make sense. Harmonisation is, however, a multi-headed monster: *there is only one way in which data are properly harmonised, and there are many ways in which data can deviate from this*. Some harmonisation problems are easy to solve (like converting from grams to kilograms, or days to years), others can be complex (like how to compare the measurement by different instruments for measuring quality of life or child development), and some may seem outright hopeless (if the cohorts measure different things in different ways).

For example, all cohorts will have data about gestational age, however it could be that this variable is measured in *days*, *completed weeks + days*, *completed weeks only*, *days rounded to nearest week*, and

so on. To harmonise such variables is generally straightforward once we have chosen a specific target representation.

Many harmonisation problems can be circumvented and solved by judiciously mapping different variables into a common framework and derive one-to-one transformations that will convert the original cohort data into common variables. WP3 created a monumental inventory of the information collected in all cohorts, mapped the collected information into a comprehensive schema, and proposed a procedure to derive transformations into common, harmonised variables. In practice, such transformations are implemented on the RECAP platforms by means of either category conversion tables, or by means of `Magma JS` code, that will recode collected variables in each cohort into a harmonised variable for each cohort. Once such harmonised data are available at the RECAP nodes, we can embark on joint analyses that use data from multiple cohorts.

Deliverable D5.3 (this report) attempts to address harmonisation problems for which the transformation approach may not work or is considered of dubious value. For example, suppose that cohort A asks the question: *How far can you walk without stopping/experiencing severe discomfort, on your own, with aid if normally used?* with response categories

1 can't walk
2 a few steps only
3 more than a few steps but less than 200 yards
4 200 yards or more.

In cohort B we may have asked the question: *Can you walk 400 meters without resting (with walking stick if necessary)?* with response categories

1 yes, no difficulty
2 yes, with minor difficulty
3 yes, with major difficulty
4 no

Of course, both questions intend to measure the ability to walk, but it is far from clear how an answer on the "How far" question can be converted into a response on the "Can you" question. In order to address those types of incomparability we need a little more advanced statistical methods, like *response conversion* to a common scale (Van Buuren et al, 2005) or *multiple imputation* (Van Buuren 2018, section 9.4).

Deliverable 5.1 introduced a conceptual framework for harmonisation problems and suggested multiple imputation as a as a generic solution. *It is not yet known whether multiple imputation can be applied in a federated environment.* The goal of the present report is to outline what is needed to make multiple imputation work applicable federated data analyses.

2.2 RECAP platform and the use of DataSHIELD

The RECAP platform allows cohort data owners to save their individual level data on their own node and grant access to researchers who want to use their data for analysis. In this way, researchers can use the data without physically getting the individual level data. This means that the data of the cohorts is saved at different locations, namely on their own cohort nodes. Moreover, this implies that the individual level data cannot be merged into one datafile, and, therefore, requires specialised analysis. Federated analysis has the ability to analyse each data set locally at the node and pool the analyses from different nodes into one result. Thus, there is no need for physically getting the dataset.

`DataSHIELD` is software can detect and filter out statistical analyses that would inadvertently disclose the data. For example, `DataSHIELD` does not allow the creation of scatterplots or of dumps of the data table. Instead, `DataSHIELD` insists on a "minimal number of records" per cell, so that individual data records cannot be disclosed, either in part or in full. However, `DataSHIELD` does allow the estimation of parameters, like regression weights. The list of procedures supported by `DataSHIELD` is relative short (essentially restricted to several types of regressions) but is expected to grow over time.

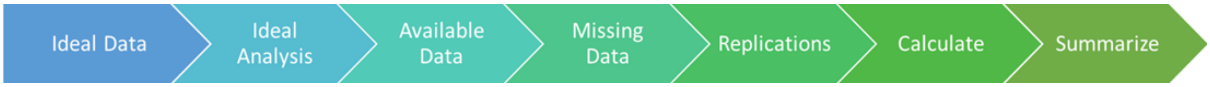
A direct consequence of distributed data storage is that the harmonisation needs to be done at the cohort nodes. In general, in order to harmonise data properly, access is needed to the individual level data, so as to make sure that the data is harmonisation agreeing to the harmonisation procedure developed by WP3. The cohort administrator is the only role with full access and needs to delegate access to the data harmoniser. This implies that the investigator who wants to perform joint analyses cannot also be the data harmoniser (unless – of course – provided with full access, but this would defeat the purpose of the platform).

The mapping-transformation approach is well supported in RECAP. Detailed instructions for harmonisation are available https://gitlab.inesctec.pt/wp4-recap/wp4_workshop/wikis/harmonisation.

2.3 MICE package for harmonisation

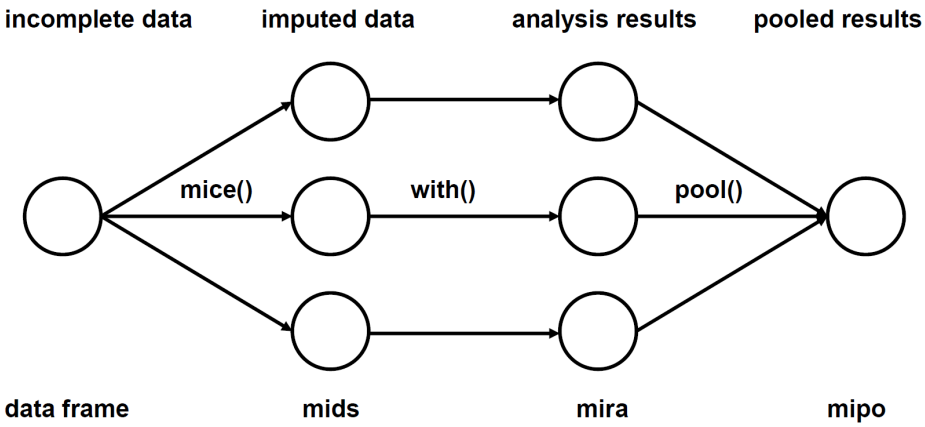
The mapping-transformation approach to harmonisation should be used where possible. There will be cases, however, where the assumption needed to make transformations work are too strong or would

lead to unrealistic and/or incomparable measures. In those case, we may leave the original cohort variables in the study, and code any unobserved measures in the other cohorts as missing values. Such harmonisation problems are candidates to be treated by methods aimed to solve incomplete data problems. Deliverable D5.2 proposes a seven-step approach to missing data problems.



The figure summarizes these steps. For illustration, let us apply these steps to the problem with 2 walking disability questions from Section 2.1

1. Ideal Data: envision what data we would like to have had to solve our problem given unlimited resources: *Ideally, we would have like to have measured both cohorts A and B using the same item, e.g., if both A and B had used question "How far"*;
2. Ideal Analysis: define what analysis we would perform if we had the ideal data: *Ideally, we would then count the percentage of people in categories 1, 2 and 3 as a estimate on walking problem in cohorts A and B*;
3. Available Data: evaluate which parts of the ideal data are available to us: *We don't have the data of cohort B on question "How far"*;
4. Missing data: determine why some parts of the ideal data are missing: *The missingness is caused by design, and unlikely to be related to the response on question "How far"*;
5. Replications: construct replications of the unseen ideal data: *Now we need information on the relation between questions "How far" and "Can you". For simplicity suppose we have a third study that measured both. From their contingency table we can draw realistic values for "How far" given we know the response of "Can you". We need to repeat the process m times to account for the uncertainty of the imputed data*;
6. Calculate: our answer from each replication by the method of point 2: *For both cohorts, calculate the percentage in categories 1, 2 and 3 in both cohorts for each of the m replications, and calculate the standard errors;*
7. Summarize: the answer over the replications: *Aggregate the results over the m estimates, and calculate the proper standard error. We now have point estimates and standard errors for both cohorts A and B on question "How far".*



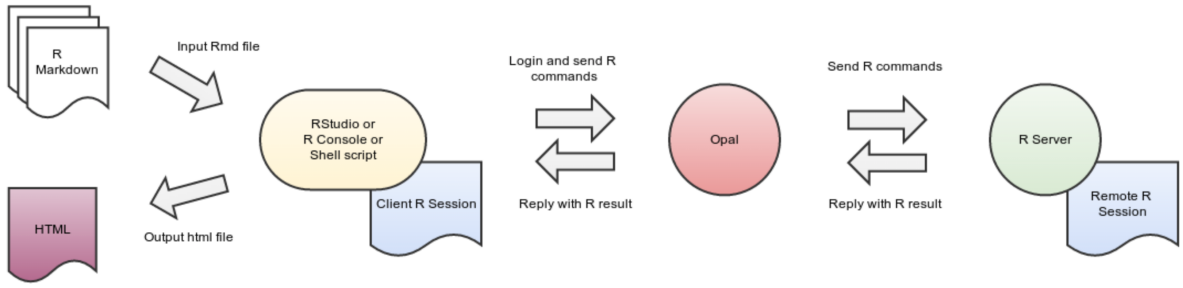
The `mice` package in `R` (Van Buuren & Oudshoorn 2011; Van Buuren 2018) implements steps 5, 6 and 7. The figure shows the main steps implemented in the `mice` package. In particular,

- step 5 in the seven-step approach corresponds to transition from incomplete data to imputed data, i.e., finding `m` (here `m` = 3) replacements for each missing response in cohort B (creating a `mids` object);
- step 6 corresponds calculating m percentages and standard errors per cohort (creating a `mira` object);
- step 7 corresponds to aggregating the cohort estimates and their standard errors (creating a `mipo` object).

It would be useful if the function `mice()`, `with()` and `pool()` in the mice package could be made available for federated data, e.g., as a component of the RECAP platform. This report describes our experiences and advances made with the integration of `mice` and `DataSHIELD`.

3 WORK DONE TO IMPLEMENT MICE IN DATASHIELD

3.1 Generic approach



The figure depicts the main flow of the analysis done in R on one `DataSHIELD` node.

There are three main function in MICE: `mice()`, `with()` and `pool()`. All functions are needed. The functionality of the `mice()` is most challenging to implement in `DataSHIELD`. We therefore concentrate on the `mice()` function.

3.2 MICE function – one node

The `mice()` function takes an incomplete data set, and returns a multiply imputed dataset as an object of class `mids`. Within the client session on `DataSHIELD` it is not possible to see the incomplete data, which resides on the server. Therefor any routines to generate imputations should run on the server.

Let us first consider the scenario where there is only one server and one analyst (client). In that case, the analyst can run the unaltered `mice()` function on the server, and produce the `mids` object for further analysis. Within the same session, we may also run the `with()` function (which applies the analysis of interest to each of the *m* imputed data sets) and the `pool()` function. The only thing that needs to be implemented are disclosure scans for the built-in `print()` , `summary`() function of the `mids`, `mira` and `mipo` objects.

Persistence over sessions of the imputations requires permanent storage of the `mids` object. The `mids` object cannot be stored on the client because it contains the full original data. Potentially the client could

store only the imputations, which are synthetic values and are not disclosive. The downside of this approach is that the connection with the original data is lost, so that additional administration is needed.

A better approach is to store the imputations along with the original data on the server. There are two different formats in which the data could be stored. Server-side storage of the generated `mids` object eases the subsequent analysis of the data. Another possibility is to expand the multiple-imputed data in long format, which may be better interoperable with the database system used in `DataSHIELD`. There are plans to support storage of user sessions at the server, which would support storage of `mids` object. It is not currently possible to "enrich" the data with imputations that could subsequently feed into the statistical analyses of other users.

3.3 MICE function – multiple nodes

In the federated analysis there are multiple nodes, each containing part of the data to be analysed. Each of the dataset will be incomplete, and – in addition – the sets of available variables will differ across nodes, thus some variables entirely missing in a given node.

We might consider two strategies to impute the data:

- Each node is a separate data set, and we produce multiple imputations per node;
- The data conceptually form one big dataset, and we produce multiple imputations for the combined data.

The first strategy is a relatively straightforward repeated application of the one-node strategy of Section 3.2. It uses a different imputation model for each node. We cannot impute variables that were not measured in that study. The strategy is primarily useful to repair item-nonresponse problems.

The second strategy mimics the method that we would have used if we had had access to all data. This strategy allows us to define a common imputation model over all nodes and allows us to impute variables that were not collected at all nodes. This strategy is needed for harmonisation across different nodes using imputation, but also more challenging to implement.

MICE imputation algorithm for multiple nodes

The conventional `mice` algorithm for one dataset applies a univariate imputation for each incomplete variable in the model. The data set is first completed by a random draw from the observed values for each variable. The `mice` algorithm then iterates over the variables, every time replacing the old imputations by newly generated imputations. A typically `mice` run makes 10 passes through the data set. The whole procedure is repeated m times, thus generating m multiply imputed data set. The number of imputed data sets is typically quite low, e.g., $m = 5$ or $m = 20$.

The next algorithm generalises the `mice` algorithm to multiple nodes. There is one client (analyst) node and multiple `DataSHIELD` nodes:

MICE on DataSHIELD algorithm - General flow

```
CLIENT
initialization:

login --> datasources
create unique pooled variable names (unique(ds.names("D")))
select variables to impute --> varnames
select imputation methods --> ds.methods
ds.mice(data = varnames, method = ds.methods, datasources = datasources, ...) --> list of mids-object without data

SERVER

initialize Dimp object per node (miceDS(..., maxit = 0)) --> Dimp

CLIENT
update missing data statistics --> request to SERVER md.stats.list()

loop over k (iteration), i (imputation number), h (variable number)

    CLIENT
    estimate global imputations model parameters (ds.glm()) --> params
    calculate predicted values --> fitted
    for PMM:
            SERVER
            for every unit, find observed y corresponding to nodes * 10 closest units
```

```
            CLIENT
            for every unit, reduce to donors to 10 closest candidates across all nodes
    SERVER:
    calculate imputations for variable h on node n under specified model
    update Dimp per node

    CLIENT:
    calculate and store convergence statistics (ds.mean, ds.var of imputations)

end loop

Result: Updated Dimp (mids object), imputed under common model, stored per node
We might wish to rename and store Dimp over sessions for later re-use

Data transferred SERVER to CLIENT:
  - missing data summary statistics
  - params
  - fitted
  - donorlist

No transfer of:
  - data (however for pmm: 10 *candidate* imputations per unit,
          may need trimming of observed data to remove sensitive extremes)
  - imputations (calculated on server-side)
  - missing data pattern (not needed at client)
```

The procedure is quite general and allows for any univariate imputation method. Univariate imputation methods based on draws from the parameter distributions of the imputation model (like normal or logistic regression imputation) require aggregating the parameter estimates across different nodes, which is standard functionality in `DataSHIELD`. Methods that sample from the original data by nearest neighbours' methods (like predictive mean matching) require finding the outcome values of the, say, 10 nearest neighbours in each node, and then one impute is randomly drawn from the combined sets of nearest neighbours (e.g. if we have three nodes, we need to draw one value from 3 * 10 = 30 candidates).

3.3.1 Predictive mean matching for multiple nodes

Predictive mean matching is the default imputation method in `mice` for continuous data. The method selects a small number of close matches (typically 5 or 10) in the predictive metric, randomly draws one of these matches, and takes the observed value of the drawn match as the imputed value. This section discusses an implementation for multiple nodes.

Suppose we have a data set with one incomplete variable Y, and with one or more complete variables X. Predictive mean matching (PMM) is an imputation method that finds plausible values from the observed Y, and uses these values to impute (fill in) the unobserved Y.
For the case that all relevant data are available, the algorithm for PMM takes the following steps:

1. Select the rows (cases) for which Y is observed. We call this subset Y_{obs} and X_{obs}. The remaining subset is called Y_{mis} and X_{mis};
2. Fit a linear regression model to predict Y_{obs} from X_{obs};
3. Calculate predicted values $\hat{Y}_{\text{obs}}$ and $\hat{Y}_{\text{mis}}$;
4. For each value in $Y_{\text{mis},i}$ find the 5 cases for which $|\hat{Y}_{\text{mis},i} - \hat{Y}_{\text{obs},j}|$ is minimum, randomly draw one of these cases (j'), and take the *observed* value $Y_{\text{obs},j'}$ as the imputation;
5. Store the imputed data value.

A (slightly more elaborated) version of these steps are implemented in `mice.impute.pmm()`, the default imputation method for numerical data. In practice, there should also be a Bayesian step that draw the model parameters to reflect the uncertainty of the imputation models, and the matching should be of a slightly different type. See Van Buuren (2018) for details. These issues are ignored here.

The following example is taken from the standard documentation of `mice.impute.pmm()`, and represent a non-typical application of the function.

```
library(mice)
## Loading required package: lattice
```

```
##
## Attaching package: 'mice'
## The following objects are masked from 'package:base':
##
##     cbind, rbind
set.seed(53177)
xname <- c('age', 'hgt', 'wgt')
r <- stats::complete.cases(boys[, xname])
x <- boys[r, xname]
y <- boys[r, 'tv']
ry <- !is.na(y)
table(ry)
## ry
## FALSE  TRUE
##   503   224
# percentage of missing data in tv
sum(!ry) / length(ry)
## [1] 0.692
```

In this example we have only missing in *y* (testicular volume). In this case, we have 224 observed values and 503 missing values, this about 69 percent in y is missing.

```
# Impute missing tv data
yimp <- mice.impute.pmm(y, ry, x)
length(yimp)
## [1] 503
table(yimp)
## yimp
##   1   2   3   4   5   6   8   9  10  12  13  15  16  17  18  20  25
##  66 142  65  60   3   6  11   1  15  13   3  28   3   1   6  49  31
```

The `mice.impute.pmm()` function return 503 imputed values. The table shows that the majority of imputed values have volumes of 1-4 ml.

There are three challenges is to implement PMM in `DataSHIELD` when there is no direct access to the underlying data in which the imputation model is fitted.

1. Fitting the imputation model on `xobs` and `yobs` in `mice:::.norm.draw()` should be distributed over different data nodes, and pooled to obtain the parameter estimates of the imputation model;
2. The matching process should find the 5 best matches across all data nodes;

3. The imputations should be stored locally at each data node.

A decidated `ds.impute.pmm()` should be able to do all this, and be registered as a protected `DataSHIELD` method.

Repositories that contain relevant work can be found at:

https://github.com/stefvanbuuren/dsMiceClient
https://github.com/stefvanbuuren/dsMice
https://github.com/gflcampos/dsMiceClient
https://github.com/gflcampos/dsMice

3.4 Problems encountered

The `mice` algorithm is iterative and needs to iterate over the nodes. A typical run that imputes 30 variables, five imputed data sets (m = 5) in 10 iterations takes about 1 minute to complete. In a federated environment with 4 nodes, there need to be $4 + 30 \times 5 \times 10 \times 4 \times 3$ (calls per iteration) = 18004 server calls. If every call takes 3 second to complete, the algorithm will run for 15 hours. Thus while possible, the available software for federated analysis using multiple nodes is a little slow to be useful for iterative algorithms like `mice`.

Persistence of the imputed data over sessions has not yet been solved. It is currently possible to save the results of one session, but we cannot use imputations created earlier. This is inconvenient and affects reproducibility of the imputes.

4 WORK PLANNED: FEDERATED MICE IN DATASHIELD

The `mice` imputation algorithm described in Section 3.3 uses a different imputation model for each node. This strategy is useful to repair item-nonresponse problem, but it disables the computation of a global imputation model that is important to build a federated data analysis process using the data in all nodes. It is also important when there is the need to impute variables that were not collected at all nodes, so the combined model can improve the results for this type of imputation and other scenarios.

The following algorithm generalises the original federated `mice` algorithm for `DataSHIELD`:

```
// Initialization
GET FROM EACH NODE:
      list of variable names without NAs
      list of variable names with NAs
      list of indexes of records of vars without NAs

// MAIN CYCLE
FOR K=1 to NA_VARS:
//NA_VARS is the number of variables with missing values
//in the current implementation only the first variable with missing values is imputed

//Build a generalized Regression Matrix for Var_K:
      FOR EACH NODE:
            get list of vars 1..K-1 with NAs
            get list of indices in which vars 1 ... K-1 has NAs
            build matrix with records of variables without missing values
            RETURN matrix
      At the client side:
            sum the matrices received from each node
            compute the regression equation
            send regression equation to all participating nodes

// Imputing values to Var1 missing values:
      FOR EACH NODE:
            use the received regression equation to compute the estimated
            values for all VAR1 missing values

           FOR EACH CELL with NA value in current NODE:
            compute the 5 nearest values to the value predicted by
            regression
            If user chooses a local assignment:
                  impute the result of a random choice among the 5 nearest
                  local values
            otherwise:
                  collect the 5 nearest values from other nodes
                  impute the result of a random choice among the 5 nearest
                  global values

      return list of imputed values
```

The implementation of federated MICE that we are proposing might further profit from recent developments in `DataSHIELD` and the R language. Although we have only implemented federated MICE up to the imputation of the first variable with missing values, we have already a couple of proposals to mitigate the efficiency issues reported in previous sections. We propose two different approaches with the advantage that they can be combined providing extra efficiency. The first one is based on the storage of more intermediate results in the server-side and the other based on the use of multithreading.

There are several intermediate results that have to be repeated several times to produce the same results. New developments of DataSHIELD (https://cran.obiba.org/web/opal/datashield.workspace_save.html) can help to store computations where repetition is avoided even if that requires storing them across different `DataSHIELD` sessions.

Although the original R language does not support multi-threading, there are already packages, written in other languages, like *C++*, that implement multi-threading and are available as an R package (using an implementation based on `Rcpp`, that "binds" the two languages). We envisage to use these two approaches to mitigate most of the efficiency issues of the implementation of a federated MICE.

5 DISCUSSION

Multiple imputation is a very useful quantitative technique for solving missing data and harmonisation problems in federated data analysis. We believe there is a future for federated data analysis. We found **no intrinsic barriers** to implement `mice` in a federated environment. It is true though, that it is quite hard to create a functional system with the current versions of the `DataSHIELD` software. It is straightforward to store imputations exclusively at the client side (so we do not require storage on the individual nodes), a possibility that is fine for imputations that have a narrow scope for a specific statistical analyses. However, in the philosophy of both `mice` and `DataSHIELD`, imputations would target multiple users and multiple analyses. In that case it is more sensible to distribute the imputations over the nodes, and store them at the location of individual nodes, next to the data. This entails a use case with an interesting trade-off: *While the data never leave the node, the information in the data does, but - in return - the node gets supplemental data that is informed by the data present in other nodes*.

The iterative algorithm is currently hard to realize in `DataSHIELD`. Some of planned extensions of `DataSHIELD`, like server-side storage will ease implementation. An algorithm that needs to iterate over nodes is slow, due to the substantial overhead needed in communication, data access and privacy

checking. This is no fundamental bottleneck, and speed problem seems addressable by improved architecture of client-server technology.

Suggestions for further works are:

- Work out a few use cases in RECAP that demonstrate the added value of multiple imputation for solving harmonisation problems;
- Improve automation of both client-side and server-side imputation approaches;
- Request features for server-side storage.

6 LITERATURE

1. van Buuren, S., Eyres, S., Tennant, A., & Hopman-Rock, M. (2005). Improving comparability of existing data by Response Conversion. *Journal of Official Statistics*, 21(1), 53-72.
2. van Buuren, S., & Groothuis-Oudshoorn, K. (2011). MICE: Multivariate Imputation by Chained Equations in R. *Journal of Statistical Software*, 45(3), 1-67.
3. van Buuren, S. (2018). *Flexible Imputation of Missing Data. Second Edition*. Boca Raton, FL: Chapman & Hall/CRC Press. https://stefvanbuuren.name/fimd/
4. Wolfson M, Wallace SE, Masca N, Rowe G, Sheehan NA, Ferretti V, Laflamme P, Tobin MD, Macleod J, Little J, Fortier I, Knoppers BM, Burton PR. (2010). DataSHIELD: resolving a conflict in contemporary bioscience–performing a pooled analysis of individual-level data without sharing the data. *International Journal of Epidemiology*, 39(5):1372-1382.