

The Medical Informatics Platform (MIP)

Data Management Guideline

Author (MIP Support Team)

Version 03.1

(2021 12 22)

Document Control

Document Owner	HBP/EBRAINS The MIP Support Team
Document Type	MIP User Guide
Document Purpose	To provide general guidelines and descriptions around data management practices to execute in order to, leveraging the MIP data tools, achieve the processing of institute's data before being ingested to the MIP analysis engine.

Date of update	Updated by (author name)	Changes Made with Brief Description	Version
2019 05 24	Iosif Spartalis Kostis Karozos	Created 1.0 version of the document	01.00
2019 08 29	Iosif Spartalis Kostis Karozos	Reorganized the Design of the Data Factory section. Explained in detail the mapping-tasks for a mock dataset.	01.10
2019 10 21	Iosif Spartalis Kostis Karozos	Updated Data Factory deployment and usage scripts	01.20
2019 12 11	Iosif Spartalis	Updated MRI pipeline execution	01.30
2019 12 12	Christian Dhondt	Documenting reformatting using the MIP Documentation Template	02.00
2019 12 13	Christian Dhondt	Document restructuring, revision (round 1)	02.00
2019 12 14	Christian Dhondt	Document restructuring, revision (round 2)	02.00
2019 12 15	Christian Dhondt	Document improvement (end to end)	02.00
2019 12 19	Christian Dhondt	All related to learning how to MIPMAP pushed in appendix as tutorial	02.00
2019 12 29	Christian Dhondt	Full revision / improvement before review with AUEB team	02.00
2020 01 15	Christian Dhondt Iosif Spartalis	Document improvement following full review with AUEB team member.	02.00
2020 02 19	Iosif Spartalis Kostis Karozos Laith Abu-Nawwas Jerome Chaptinel		
2020 02 24	Laith Abu-Nawwas	Document revision and restructuring (page 3 and 7-15); removal of section "Types of Information and Data"	02.00
2020 03 04	Iosif Spartalis	Incorporation of suggested proposals. Minor improvements and clarifications. Added the preprocessing design tool in appendix 2.	02.10

2020 05 07	Iosif Spartalis Kostis Karozos Laith Abu-Nawwas	Added some instructions in the Appendix 2. Made some small improvements and corrections.	02.20
2020 07 07	Kostis Karozos Laith Abu-Nawwas	Added Licenses for the Data Factory components	02.30
2021 10 04	Iosif Spartalis Laith Abu-Nawwas	Upgraded the previous document from a Data processing user guide to a Data Management guideline; using the data quality tools as a comprehensive data framework	V3.00
2021 12 22	Iosif Spartalis Laith Abu-Nawwas Pauline Ducouret	Compilation of all user guides for data quality tools	V3.10

Acronyms

HBP	Human Brain Project
MIP	Medical Informatics Platform
CHUV	Centre Hospitalier Universitaire Vaudois
AUEB	Athens University of Economics and Business
EHR	Electronic Health Record
SQL	Structured Query Language
CDEs	Common Data Elements
JSON	JavaScript Object Notation; is open-standard file format or data interchange format
CSV	Comma-separated values file; is a delimited text file that uses a comma to separate values.
MRI	Magnetic Resonance Imaging
DICOM	Digital Imaging and Communications in Medicine
NIFTI	Neuroimaging Informatics Technology Initiative
DQC-Tool	Data Quality Control Tools
GUI	Graphical User Interface
DC	Data Catalogue

Table of Contents

Acronyms	4
About this Document	7
Target Audience	7
Document Overview	7
Concepts and Definitions	8
The MIP	9
Data Management and Processing	10
Extraction and compilation of clinical data as a csv file	12
Definition of Common Data Elements for a medical condition	13
Data cleaning, validation, and upload	16
HBP-MIP MRI parallel neuromorphometrics pipeline – Wiki	18
Prerequisites	18
Hardware requirements.....	18
Tools Set-up:	19
MRI Organizer step:	19
HBP-MIP Data Catalogue - Wiki	22
Overview	22
Users	22
User Guide	22
Installing / Getting started	32
HBP-MIP Data Quality Control Tool - Wiki.....	35
Installation.....	35
Linux.....	35
Windows (with WLS 2)	37
MacOS	41
Files Specifications	42
MIP "Frictionless data" table-schema.....	42
CDEs Dictionary Excel file.....	42
Data Catalogue Excel file	43
Data Validation & Cleaning functionality (CSV)	43
Validation & Cleaning functionality per datatype	43
Numerical Type column.....	44

Integer Type column.....	44
Date Type column	45
Nominal Type column.....	45
GUI Manual.....	47
Data Validation and Cleaning.....	47
Dataset Schema Inference.....	49
Data Mapping	51
DICOM Tab	61
MIPMap supported functions	62
Report Files Descriptions and Details.....	64
Data Validation Report of a CSV file (PDF)	64
DICOM MRI Metadata report files	67
CLI Manual.....	69
Profiling and Validating a CSV dataset:	69
Inference of a CSV dataset's schema	70
DICOM MRI metadata validation.....	72
Developers Notes	73
Windows.....	73
MacOS	74
Ubuntu.....	75

About this Document

Target Audience

This document is targeting data processors and members of data controllers from institutes interested in analysing their clinical and research data within the MIP. It mainly addresses technical coordinators and data managers who want to prepare their data to be ingested into the MIP analysis engine.

The document provides the information needed to understand:

- The approach of data processing to achieve data ingestion to the MIP.
- The tools that have been developed to facilitate the data processing (The MIP Data Catalogue, the MRI parallel neuromorphometrics pipeline, and the MIP Data Quality Control Tool (MIP-DQC Tool).
- The user guide for each of these tools

Document Overview

This document describes the MIP Data architecture, and for which purpose it was assembled. It explains the overall data processing required to conform to MIP standards before being uploaded to the MIP and how the MIP Data Quality tools shall be used to help achieving these standards.

It is a step-by-step guide covering data end to end processing and it provides links to additional documentation and detailed instructions described in the different GitHub repositories and shall not be seen in isolation.

Concepts and Definitions

Common Data Elements (CDEs)

A set of standard variables defined by clinical experts and data scientists, which would be used by researchers to perform analysis on specific medical conditions at the federation level. In the MIP context, we use the term CDEs to refer to the standardized federated data models only.

Data Element

In metadata, the term data element is an atomic unit of data that has precise meaning or precise semantics. [Beynon-Davies P. (2004). Database Systems 3rd Edition. Palgrave, Basingstoke, UK]

Data Models (Metadata)

Data Model (Metadata) describes the structure of database variables found in specific extract of a hospital's database, including descriptive metadata, structural metadata, administrative metadata, reference metadata and statistical metadata. [1] Zeng, Marcia (2004). "Metadata Types and Functions". NISO. Archived from the original on 7 October 2016. Retrieved 5 October 2016.

Database Variables:

A variable or scalar is a storage address (identified by an index or address) paired with an associated symbolic name, which contains some known or unknown quantity of information referred to as a value. [Knuth, Donald (1997). The Art of Computer Programming. 1 (3rd ed.). Reading, Massachusetts: Addison-Wesley. p. 3-4.]

EBRAINS

EBRAINS is a new digital research infrastructure, created by the EU-funded Human Brain Project, that gathers an extensive range of data and tools for brain-related research. EBRAINS will capitalize on the work performed by the Human Brain Project teams in digital neuroscience, brain medicine, and brain-inspired technology and will take it to the next level.

Electronic Health Records (EHR)

Health information and clinical records registered per each patient per visit in the hospital's database (Oracle, SQL or any database systems) and usually transferred in db or CSV format. EHR usually contain different levels of data; we might define them in this context as spaces, domain, and sub-domain. For example, General space might include demographic, social status or patient's medical history as different data domains. On the other hand, EHR contain other data spaces related to the specific medical condition such as Dementia or Epilepsy where each space includes specific domain and sub-domain, such as medical assessments and tests, diagnoses, treatment, and operations, etc.

Medical Conditions

Diseases are often known to be medical conditions that are associated with specific symptoms and signs. ["Disease" at Dorland's Medical Dictionary]

The MIP

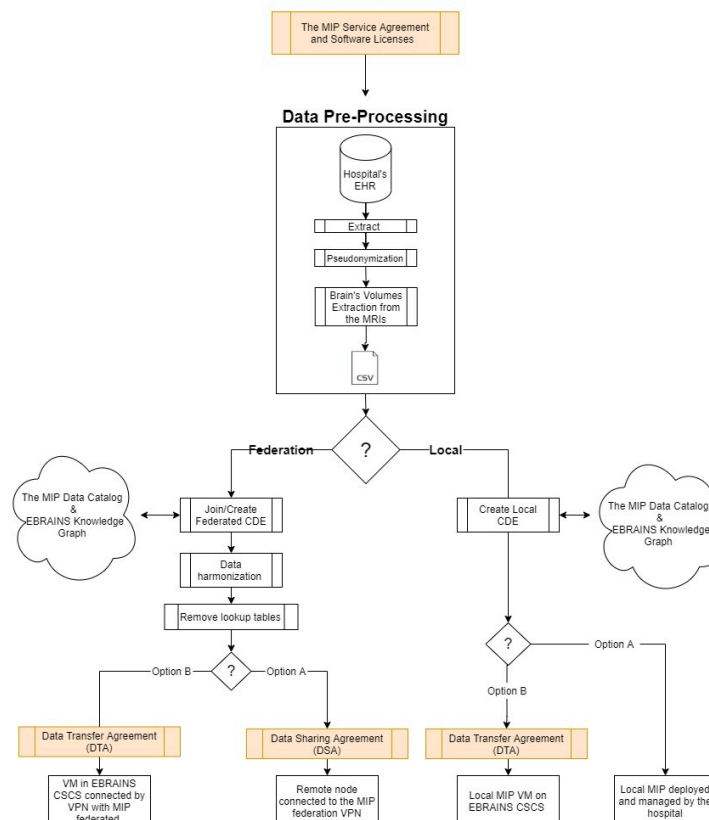
The Medical Informatics Platform (MIP) for the Human Brain Project is an innovative platform that provides an interface for various investigators (*i.e.* clinicians, neuroscientists, epidemiologists, researchers, health managers) to collect, access, explore and analyse anonymised medical data that are locked and hosted in their original hospital or research centre. Data exploration and analysis can then be performed without moving the data from the hospital where data reside, and hence without infringing on patient privacy.

The MIP has been designed adopting a 2-tier architecture, which guarantees hospital data protection and privacy by design. The MIP offers two main solutions (see diagram below):

- The MIP-Local Node contains data that can only be accessed and analysed by the Local Data Manager and its accredited staff from within the hospital.
- The MIP-Federated Node contains anonymized data and can be connected to other MIP-Federated Nodes located in other hospitals when the decision is made to join the MIP Network (medical condition-based federations of hospitals) and to provide access to hospital data to members of the network.

Upon signed agreements (between the data provider's institution and the CHUV), and admission to the MIP Network, accredited researchers can query multiple MIP federated Nodes and obtain aggregated results. Queries of the MIP-Federated Nodes does not allow to copy, download, or upload any data, nor to see individual patient's data.

The MIP Data Governance flow



Data Management and Processing

Data originating from diverse hospitals are highly heterogeneous in nature and hence cannot be uploaded into the MIP *per se*. Multiple steps need to be done that are represented in the following figure and which can be regrouped in two main groups:

- Identification of variables that will be made available in the MIP to fulfill the scientific analysis objective and definition of the Data Model (Metadata). In case of a federated MIP, all hospitals must agree to a Common Data Elements (CDEs).
- Data pre-processing to conform to the CDEs predefined and to the MIP's standard formats.

The standard format of the dataset file is .csv UTF-8. Because csv format doesn't allow to keep the Metadata structure, a second file is required, which corresponds to the CDEs for which the MIP standard format is JSON.

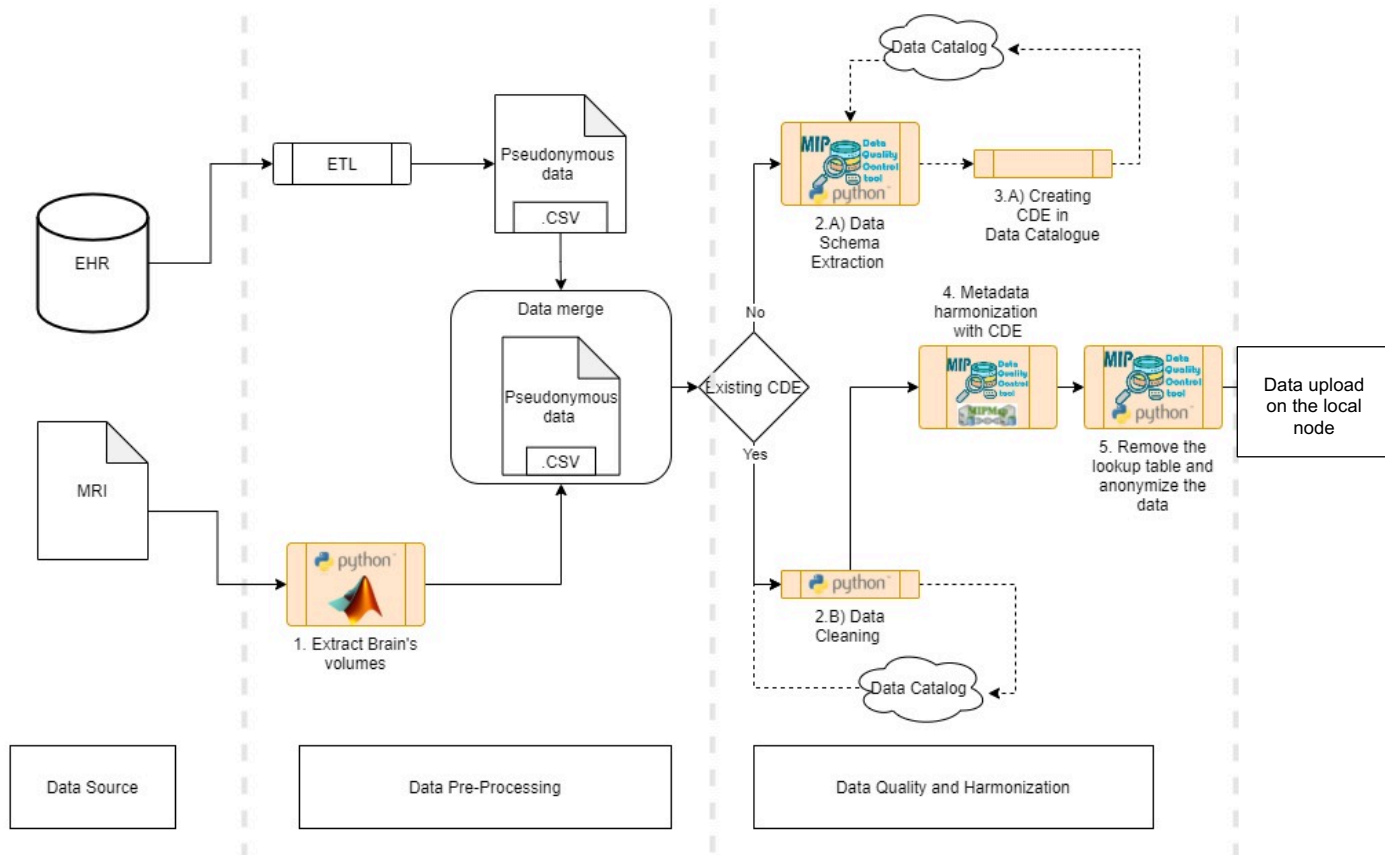
To help data managers to go through these processes, three main tools have been developed until now:

- **The MIP Data Catalogue:** Web-based application that is used to explore and manage MIP Data Models and CDEs pre-defined by the federation's community or the use case group. All the existing Medical Conditions, Federations and Hospitals and their associated Common Data Elements (CDEs) are available within the MIP Data Catalogue, which is the unique source of reference for the MIP and the MIP-DQC Tool. The MIP Data Catalogue can be accessed at the following address ([HERE](#)). *It is recommended to use any browser except Firefox.*
- **The MRI parallel neuromorphometrics pipeline:** A python wrapper to run the Statistical Parametric Mapping SPM12 pipeline in parallel with Matlab over multiple CPU cores.
- **The MIP Data Quality Control Tool (MIP-DQC Tool):** A standalone software that provides hospital personnel an easy way to explore, validate and transform their dataset based on the CDEs before uploading them into the MIP. MIP-DQC Tool has both, a Command Line Interface (CLI) and a Graphical User Interface (GUI), with only the latter having the full set of functionalities. The MIP-DQC Tool GUI version has the following functionalities:
 - o Validating the hospital tabular (csv) data and producing a validation report and some overall statistics about the data.
 - o Data cleaning capability based on the previously performed validation report.
 - o Inference of a dataset's schema and producing a schema file in Frictionless json or the MIP Data Catalogue's excel format.
 - o Designing and performing schema mapping of an incoming hospital dataset to a certain Pathology's Common Data Element (CDE) schema.
 - o Producing a DICOM MRI validation and statistical report, based on the meta-data headers.
 - o The tabular (csv) data validation functionality has the option of downloading the pathology's CDE metadata directly from the MIP Data Catalogue's API. Please note that this option is not available in the CLI version.
 - o The schema mapping functionality is performed by the MIPMap engine packaged in a Docker container, which runs in the background. Please note that this option is not available in the CLI version.

The user guide for each of these three tools can be found in the next sections of this document. Through description of the steps required before being able to upload data on the MIP, the information of which tool can be used, is provided. The following table lists the sequential steps as well as the tool that can be used to simplify this process.

MIP Data Processing Steps and available Tools				
Data extraction and compilation			Common Data Element definition	
1	2 (optional)	3 (optional)	4	5
Extract the clinical data as a csv file	Extract the brain volumes from the nifti to the csv file	Merge the brain volumes with the clinical data	Infer the data schema from the csv file to produce the Data Model.xlsx file	For the federation, there should be a discussion around the standardization of variables in the Data Model to produce a standard Common Data Element (CDE, or Data Model) as an.xlsx file
	MRI parallel neuromorphometrics pipeline		MIP-DQC Tool	

MIP Data Processing Steps and available Tools				
Common Data Element definition		Data cleaning, validation, and upload		
Optional	6	7	8	9
The MIP metadata dictionary can be used to standardize the variables across all medical conditions	Parse the CDEs.xlsx file in the MIP Data Catalogue to produce the CDEs.JSON	Check the dataset.csv file quality against the CDEs	Perform the data cleaning and validation	Upload of the dataset.csv file on the local node
MIP-DQC Tool	MIP Data Catalogue	MIP-DQC Tool	MIP-DQC Tool	



Extraction and compilation of clinical data as a csv file

Extract variables from the Electronic Health Record (EHR) system

A dataset that can be uploaded on the local node is a unique csv file that contains all the variables. The first step is hence to extract the clinical data of interest from the EHR system to a csv file. In most cases, a hospital EHR system contains various examination results that a patient obtains during one or more hospital visits. To keep this information, the csv file must contain:

- a field with a unique ID for each PATIENT
- a field with a unique ID for each VISIT
- a "Visit Start Date" field for each VISIT

In terms of csv file structure, the following basic conditions must be met:

- Header (variable) names
 - o must not contain special characters like space, parentheses, hyphen, etc.
 - o Use only underscore (" _ ") for word separation.
 - o must not start with any number character.
- The delimiter must be the comma character (",").
- The encoding must be ASCII.

Extract Imaging Features from MRIs - OPTIONAL

Not all medical conditions require data extracted from Brain Scans (or MRIs) which explains, why this step can be skipped. However, because the MIP recognizes only one csv data format,

features from MRI images must be extracted and merged to the csv file that contains EHR variables in case they are valuable for the scientific analysis objectives.

Although multiple methodologies can be used to extract those features, we highly recommend following the Statistical Parametric Mapping SPM12 pipeline to harmonize the methodology across the hospitals.

To simplify the process of feature extraction, you can use the **MRI parallel neuromorphometrics pipeline** that was deployed by the MIP data management team. It is a python wrapper to run the Statistical Parametric Mapping SPM12 pipeline in parallel with Matlab over multiple CPU cores. See next section for the full user guide.

As soon as all variables are merged in a unique csv file, you can proceed with the definition of the Common Data Elements for your medical condition and your scientific analysis objectives.

Definition of Common Data Elements for a medical condition

Data Model (Metadata) xlsx File format

Hospitals that want to upload a dataset on the MIP need first to create the Data Model (Metadata) of its dataset.

The Data Model xlsx file contains information about the hospital variables. It corresponds to a table that lists the variables present in the hospital data set and describes them. A template can be downloaded from the Data Catalogue web-based application (see the user guide section of this tool).

The descriptions required are listed below; in **bold** the ones that cannot be left empty:

- **name**: The full name of the variable
- **code**: The code version of the variable name
- **type**: Format of the variable. Four formats are recognized in the MIP – nominal, date, real and integer.
- **values**: Only for the nominal and binominal format. It lists all the levels of the nominal variable with the correspondence between the code and the value. The format used is - {"Code","Value"},{"Code","Value"}. For example, the sex variable is coded as: {"M","Male"},{"F","Female"}.
- **unit**: Only for real and integer. Unit of the variable (e.g., km/h)
- **canBeNull**: Mention with yes/no the possibility of this variable to be null. It is mainly useful for MIP federated nodes.
- **description**: A precise description of the variable.
- **comments**: Any comment. For example, a linkage with another variable.
- **conceptPath**: Hierarchy of the variable within the dataset as MedicalCondition/Section/Sub-section/Sub-sub-section etc... This allows the MIP-Front end to create hierarchy visualization of the data. There is no specific number of hierarchy levels required and can be customized to a specific Data Model.
- **methodology**: Specification of the methodology followed to obtain this variable when different methodologies exist.

!NOTE : Although the semantics and the conceptPath can be customized to a specific Data Model, we highly recommend following the existing hierarchy and the semantic of the actual

federations to have a global harmonization. To do so, you can use the MIP metadata dictionary in the Data Quality Control Tool to standardize the variables across all medical conditions.

The **Data Quality Control Tool** allows you to infer this Data Model xlsx file based on your data csv file. In the infer option section of the Data Quality Control Tool, we indicate the number of rows that the tool will be based on for the schema inference and the maximum number of categories that a nominal variable can have. You can also include the MIP metadata dictionary to infer the global conceptPath directly for your data.

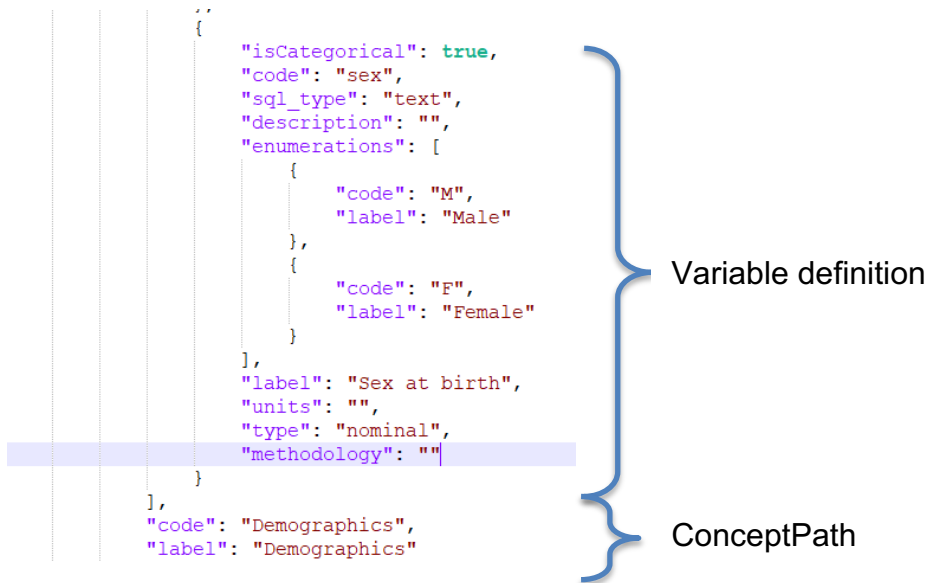
Common Data Elements

Based on the Data Model, a hospital selects the variables that will be made available on the MIP by keeping only the selected variables in the Data Model xlsx file.

In case of a MIP-Federated node, three more steps are required:

- Each variable that could lead to identification of a patient must be deleted (i.e., name, date of birth, address...) and must be replaced by a patient code. Only the data controller has access to the table of correspondence code/patient identity.
- Each hospital that is part of the federation needs to agree on the list of the variables AND on the harmonization of each variable. Indeed, variables from each hospital need to have the exact same codification and meaning for nominal variables and, should have been acquired with a similar methodology to be federated. Here is a list of discussion points:
 - Agreement on the Variable list
 - For each nominal variable, agreement of codification and meaning of this codification
 - For each real and integer variable, agreement of the unit and range
 - For each variable, agreement of the methodology followed to acquire it
- Add a nominal variable named "Dataset" that have a level for each hospital federated. A code for each hospital needs to be created (e.g. Centre Hospitalier Universitaire Vaudois is coded chuv and is written as {"chuv"," Centre Hospitalier Universitaire Vaudois "})

As soon as the CDE is defined, and in the same xlsx format as the Data Model, it can be translated to the json format that is the standard format of the MIP. Below you can see the architecture of the json file for one variable. Note, the translation between the xlsx CDEs format and the json format is made by **the Data Catalogue** web-based application. The CDE file must be named as: "Medical Condition"_cdes_"version" (e.g. "dementia_cdes_v6").



Once the CDEs have been defined, the process of preparing hospital data for ingestion to the MIP can be started.

!NOTE: In case a hospital wants to join an existing federation, it should download the pre-defined CDEs from the Data Catalogue web-based application and contact the federation leader to be able to join the federation (see already existing federations and medical conditions available on the MIP in the following sub-sections of this document).

Supported Federations and their associated Medical Conditions as of December 2021

Each CDE associated with a Medical Condition within a federation of hospitals is under the responsibility of a “Federation Leader”. The role of the “Federation Leader” is to ensure that the CDE of a Medical Condition evolves over time in a coordinated and versioned manner. The table below gives an initial view of all existing Medical Conditions with the corresponding versions of CDEs and the leader contact in use as of December 2021. For updated information about the available defined medical conditions, please refer to the Data Catalogue web-based application.

MIP-Federated node			
Medical Conditions	Leader contact	Hospital	CDE version
Dementia	Dr Thibaud Lebouvier (from 1 November 2021) Thibaud.LEBOUVIER@chu-lille.fr	CHRU Lille	V5
	Prof. Jean Francois Demonet (until 31 October 2021) Jean-Francois.Demonet@chuv.ch	Center Leenardt for Memory, CHUV, Lausanne	
Traumatic Brain Injury (TBI)	Dottore Guido Bertolini guido.bertolini@marionegri.it Stefano Finazzi stefano.finazzi@marionegri.it	Hospedale Mario Negri, Bergamo, Italy	V7
Mental Health	Prof. Pegah Sarkheil psarkheil@ukaachen.de	University Hospital RWTH, Aachen	V8
Epilepsy	Prof. Philippe Ryvlin Philippe.Ryvlin@chuv.ch	Neurosciences Department, CHUV, Lausanne	V8
NeuroNet	Lewis Killin lkillin@synapse-managers.com	SYNAPSE Research Management Partners S.L.	V1
Stroke (upcoming)			

Data cleaning, validation, and upload

Check of the dataset quality against the CDEs

Based on the CDEs defined in the previous steps, the dataset compiled as a csv, needs to be cleaned/modified and validated according to the definition of all variables. The two main violation types that need to be fixed are:

- Constraint violations which regroup:
 - minimum (for date, integer, numerical)
 - maximum (for date, integer, numerical)
 - enum (list of enumerations for nominal datatypes)

- **Datatype violations:** Case when a value in a column has a different datatype than the one that has been declared for that variable in the CDEs. This also includes the empty cells that should be truly empty and not with “NA”, “Nas”, “unknowns” etc...

Perform the data cleaning and validation

All violations highlighted before should be corrected so that each variable present in the csv file, corresponds to the one listed in the CDEs. Be careful to save the csv file in the correct format, which is csv; encoding must be ASCII with comma separation.

Data set upload

To become visible to the MIP end users, the dataset csv must be first renamed as the hospital code defined on the Dataset variable created on the CDE (e.g., “chuv.csv”). The file is then copied to the corresponding “medical condition” folder on the MIP local node and, when the hospital is part of a federation (federated hospital), to the corresponding “medical condition” folder on the MIP federated node.

The **Data Quality Control Tool** allows you to compare the CDEs json file with your dataset csv file to highlight the violations and suggests possible corrections. If the corrections suggestions are good, the DQC tool can also perform those corrections and save the corrected dataset in the correct format. In case you want to join an existing federation, the DQC tool allows you to perform a data mapping, which transforms the hospital's local variables to a set of variables of a target CDEs medical condition.

HBP-MIP MRI parallel neuromorphometrics pipeline – Wiki

The MRI parallel neuromorphometrics pipeline is a python wrapper to run the Statistical Parametric Mapping SPM12 pipeline in parallel with Matlab over multiple CPU cores. This pipeline was deployed to help hospitals participating in the MIP to process clinical MRI scans.

The main pipeline wrapper is supported by two python scripts that help the user to organize the files as below:

- **NMM Pipeline - Nifti Organiser** (<https://github.com/HBPMedical/nmm-pipeline-nifti-organizer>); This tool is meant to help data owners to organize their MRI data in such a way that it can be processed by the pipeline wrapper.
- **NMM Pipeline - Output Merge** (<https://github.com/HBPMedical/nmm-pipeline-output-merge>); This tool is to help on merging the pipeline wrapper individual MIP formatted output from the NMM pipeline into one single CSV file.

Note: The MRI parallel pipeline default configuration runs on a protocol related to T1 Nifty images, which were not pre-processed by any other tool. Therefore, if the MRI protocol is not T1, or the images were pre-processed, the results might be not accurate. Hence, in such case you will have to update the protocol definition file from the mri-preprocessing-pipeline subproject (see: <https://github.com/HBPMedical/mri-preprocessing-pipeline>).

Note II: The mri-parallel-nmm-pipeline is a python wrapper that runs Matlab and SPM 12 tools to extract data (volumes CSV) from NIFTI file only (*.nii), therefore a user must convert any MRI images DICOM files before it can be used by the wrapper. We use this open-source converter called "dcm2niix" available on GitHub: <https://github.com/rordenlab/dcm2niix/releases>

Prerequisites

- Python 3.6 (see: <https://www.python.org/>)
- Matlab (see: <https://ch.mathworks.com/products/matlab.html>)
- SPM12 ^[1] (see: <https://www.fil.ion.ucl.ac.uk/spm/software/spm12/>)
- Matlab engine for Python must be installed (see: https://www.mathworks.com/help/matlab/matlab_external/install-the-matlab-engine-for-python.html).

[1] SMP12 should be deployed in /opt folder for the wrapper to work.

Hardware requirements

- **CPU:** The wrapper has a configuration parameter where the user can limit/define the number of CPU cores to be occupied by the tool, so you can change it depends on the machine you are using and the available resources, the easy step is by:

In the python script file "mri_parallel_preprocessing.py", check line:

```
my_pool = get_context("spawn").Pool(min(4, cpu_count())) # temporarily limit to 4 cores
```

- When we tested the tools to process 1,432 .nii files it took 48 hours to extract the csv file, we occupied 4 cores out of 6 available on Intel® Core™ i7-8700K CPU @ 3.70GHz & RAM of 32 Gb, but the computer during the process was not available for any other task, so if you want to run the script you need to make the computer available for the MRI pipeline process only.
- We recommend as minimum specification to run the pipeline is to provide a machine with **16 GB of RAM x 4 cores CPU**.
- **The storage requirement** is a general rule, the available storage size should be (**12 x size of the images**), for example if you have 5 GB of images, you need a storage space of 12 x 5 = 60 GB available for the tools to process the images.

Tools Set-up:

First step: Run the following command in Linux terminal to create a parent folder to deploy all the tools inside, for the purpose of this tutorial the directory will be as below, but you change the part colored in red as you prefer:

```
mkdir /home/$USER/Downloads/"Image processing"
```

Inside the created parent folder **<Image processing>**; create 3 supporting Sub-folders to be used in each step with the following names: Organizer, Processor and Merge; or simply by running the below command in terminal.

```
mkdir /home/$USER/Downloads/"Image processing"/Organizer
mkdir /home/$USER/Downloads/"Image processing"/Processor
mkdir /home/$USER/Downloads/"Image processing"/Merge
```

Second step: Clone each tool from GitHub repositories inside the parent folder **<Image processing>** as below:

- Open the terminal inside the project folder **<Image processing>** created from the first step and run the below commands respectively:

to clone MRI_parall_organizer

```
git clone --recursive https://github.com/HBPMedical/nmm-pipeline-nifti-organizer
```

to clone MRI_parall_processing

```
git clone --recursive https://github.com/HBPMedical/mri-parallel-nmm-pipeline.git mri-full
```

to clone MRI_parall_merge

```
git clone --recursive https://github.com/HBPMedical/nmm-pipeline-output-merge
```

MRI Organizer step:

For the wrapper pipeline to work properly, the MRI Source folder must contain nifti files (*.nii) organized according to the following directory tree:

subject/visit/protocol/repetition/

MRI Organizer tool will be used to achieve such requirements, this guideline will provide a step-by-step guidance to use the tool:

Optional step: For some cases the “.nii” files were compressed on “.gz” format, as the tool only accept .nii files all .gz files should be uncompressed. This step suggests a quick command code that will uncompressed the .nii files and move them in one folder to be ready for the Organizer tool. For the purpose of this tutorial, we consider each .gz image file stored under this folder structure:

/<storage>/<subject>/<visit>/<protocol>/<filename>.nii.gz

- Open <storage> where all the images and their related sub folders are stored, then open the command terminal in that directory.
- Run the following scrip which extract all the “.gz” files inside <storage>.

Important: Make sure you run the following command in terminal in the working directory <storage>, otherwise this command can extract all .gz file found where it runs, which could lead to serious damage to the system disk space!

#Extract all '.gz' file

```
gunzip $(find . -type f -iname "*.gz") -k
```

Note: Sometimes and due to gz tool used and number of files to extract, random files will be given an error, which make the tool stop working, such files needed to be extracted manually and then rerun the command with skipping the existing file by simply pressing Enter until the skip messages stop showing.

- After finishing the extraction of all .nii files, we need to move all files to <extract folder>, you need to change <storage_device>, <username>, <dirname> and <mainfolder> in the command with the related to your project folder before you run it correctly.

#Create extract folder for the images

```
mkdir /home/$USER/Downloads/"Image processing"/extract
```

#Rename & move

```
find . -type f -iname "*.nii" -exec      rename -n 's:^./::; s:/::g; s:^:/home/$USER/Downloads/Image\ processing/extract/:' {} +
```

Note: the previous command moves all .nii files extracted to the folder <extract> from the previous step and rename each .nii file to standard name accepted by the processing tool which is in this format:

<subject>_<visit>.nii

Important: The previous command with rename -n is sat as dry run option to confirm first the results before the actual command run by removing -n from the code to avoid errors.

NMM Pipeline - Nifti Organiser step:

#Open nmm-pipeline-nifti-organizer-master folder by the terminal

```
cd /home/$USER/Downloads/"Image processing"/nmm-pipeline-nifti-organizer
```

Run the tool with the following configuration as per the previous steps:

```
# <input>: /home/$USER/Downloads/"Image processing"/extract
```

```
# <output>: /home/$USER/Downloads/"Image processing"/Organizer
```

```
./organizer.py /home/$USER/Downloads/"Image processing"/extract  
/home/$USER/Downloads/"Image processing"/Organizer
```

NMM Pipeline - mri-full:

#Open <mri-full> folder by the terminal:

```
cd /home/$USER/Downloads/"Image processing"/mri-full
```

start the image processing tool:

we use nohub command as the tool will take long time, therefore we want to assure no interruption by the system to invoke any saving power settings on the system.

```
# <input>: /home/$USER/Downloads/"Image processing"/Organizer
```

```
# <output> :/home/$USER/Downloads/"Image processing"/Processor/
```

```
nohub ./mri_parallel_preprocessing.py /home/$USER/Downloads/"Image processing"/Organizer  
/home/$USER/Downloads/"Image processing"/Processor/
```

NMM Pipeline -output-merge

#Open <nmm-pipeline-output-merge-master> folder by the terminal:

```
cd /home/$USER/Downloads/"Image processing"/nmm-pipeline-output-merge
```

start the csv files merge:

```
# <input>: /home/$USER/Downloads/"Image processing"/Processor/
```

```
# <output> : /home/$USER/Downloads/"Image processing"/Merge
```

```
./merge.py /home/$USER/Downloads/"Image processing"/Processor/  
/home/$USER/Downloads/"Image processing"/Merge
```

HBP-MIP Data Catalogue - Wiki

Github repository link

Overview

Data Catalogue (DC) is a component of the Medical Informatics Platform (MIP) for the Human Brain Project. Initially, it was designed and tasked to be the single source of truth of metadata descriptions for data residing in hospitals that have joined the MIP. In the course of things, additional needs have been recognized; therefore, DC ended up supporting the following:

- Presentation of the Medical Conditions to which data refer in the MIP.
- Presentation and visualisation of the Common Data Elements (CDEs) data models for the Medical Conditions in the MIP.
- Presentation and visualisation of the hospital local data models that have been mapped and harmonized to the CDEs data models.
- Management (create, edit, delete) of the above (global and local) data models with version control by authorized accounts. Data models are defined via DC's GUI or are imported in XLSX format.
- Generation of metadata files in JSON format according to MIP's specifications.

Users

The user of DC is the researcher who, prior to executing experiments with the MIP, needs to investigate the included Medical Conditions along with their semantics as well as what type of information is available in each hospital.

Another user is the data provider / manager of a hospital that has joined the MIP. The data provider / manager must define and manage the metadata of her hospital data.

Lastly, the MIP portal itself is a DC user as it uses its REST API to get the latest CDEs versions.

User Guide

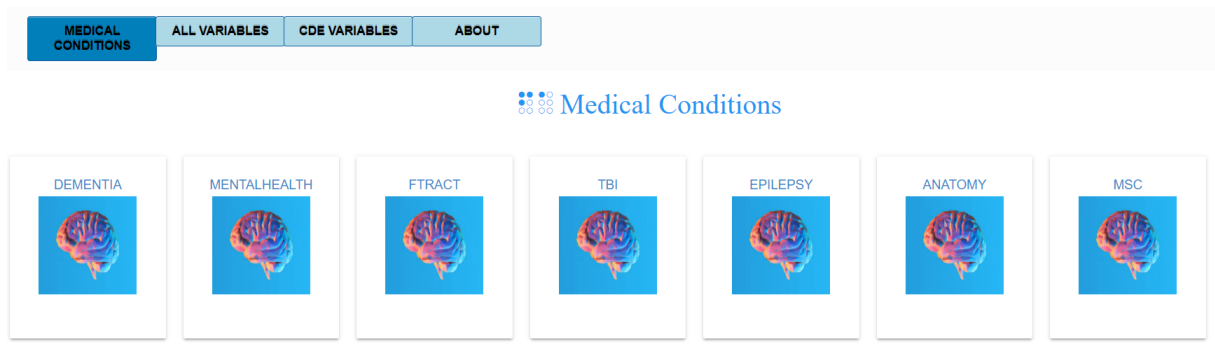
Informative Features (No login required)

The user can perform a set of actions without being logged in.

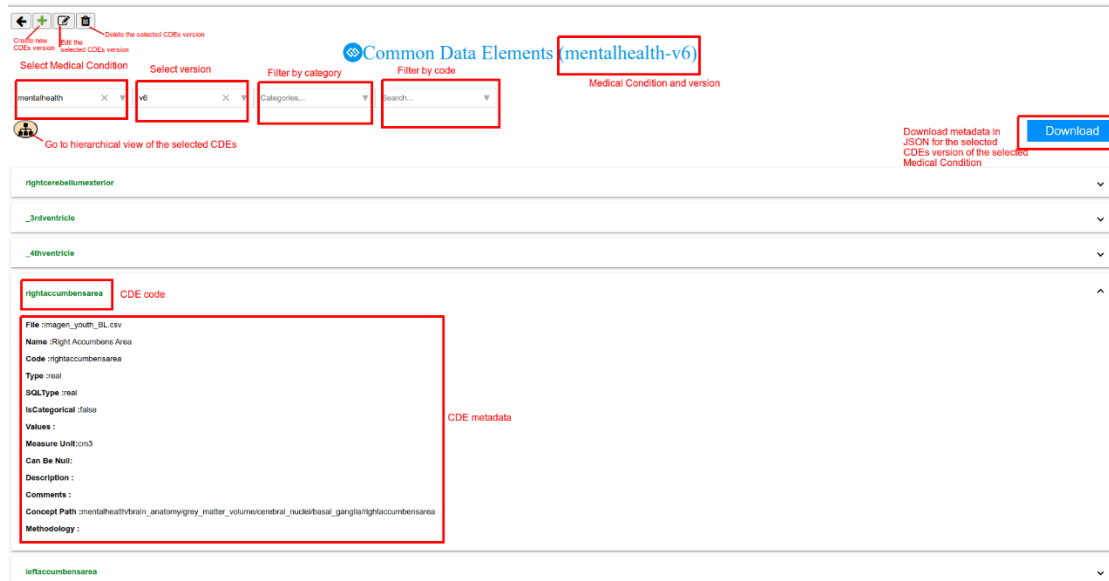
Common Data Elements (CDEs):

- View all medical conditions

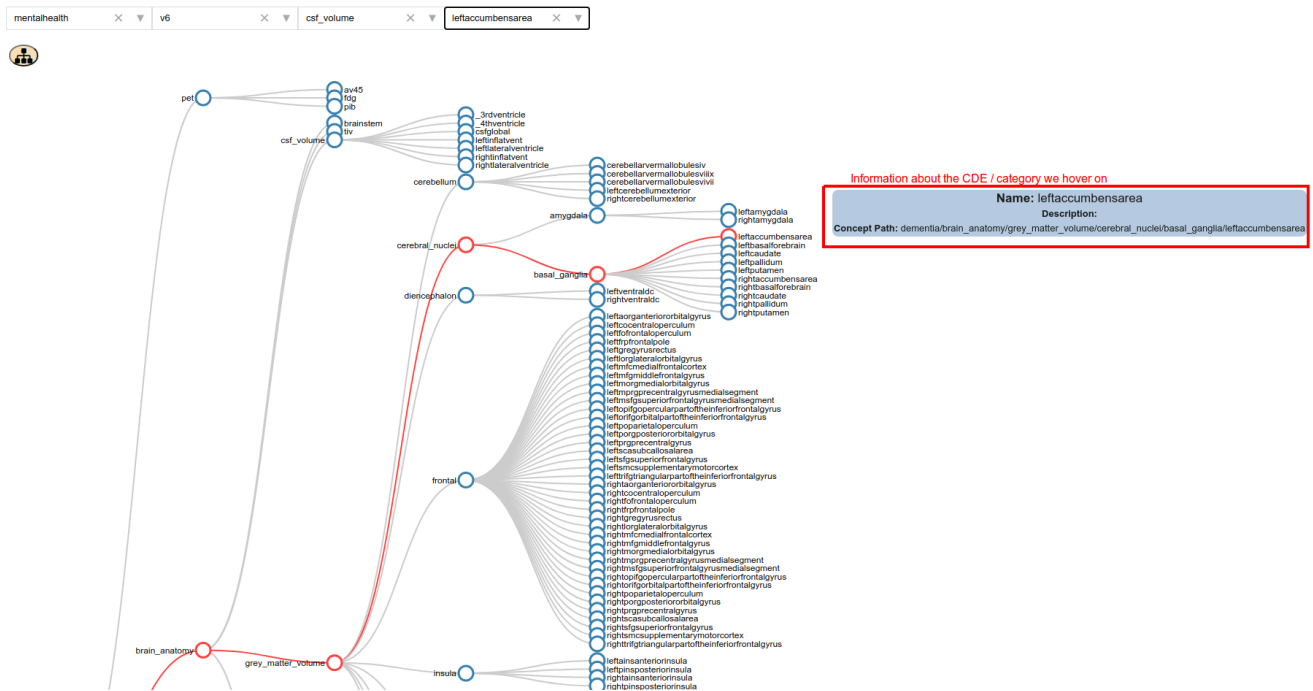
Data Catalogue



- Select the current or any previous version of the CDEs meta-data of a specific medical condition
- Search CDEs meta-data based on variables' category or code for a medical condition version
- View details about the meta-data
- Download the JSON that contains the meta-data that will be used in the MIP Federated Node for a specific medical condition



- View the hierarchy of the meta-data in an indexable tree structure



Hospital Metadata:

- View the meta-data of all hospitals combined

All Hospital Variables

Filter by category

Categories... Search... Filter by code

P_Fam_Antecedent (Lille) Meta-data variable code and hospital name

File :EHR_General.csv

Name :P_Fam_Antecedent

Code :P_Fam_Antecedent

Type :numeric

Values :

Measure Unit :

Can Be Null :

Description :Yes if there are family history or no if not

Comments :

Concept Path :/root/history/family/P_Fam_Antecedent


Methodology :

P_Fam_Autres (Lille)



- View the versions of the meta-data of each hospital and search with category and/or code

- Download the JSON that contains the meta-data of the hospital local data model

Select the version of the meta-data variables

 Hospital name and version of the meta-data variables

v1 Categories... Search...

  Filter by category Filter by code

[Download](#)

P_Fam_Antecedent Meta-data variable code

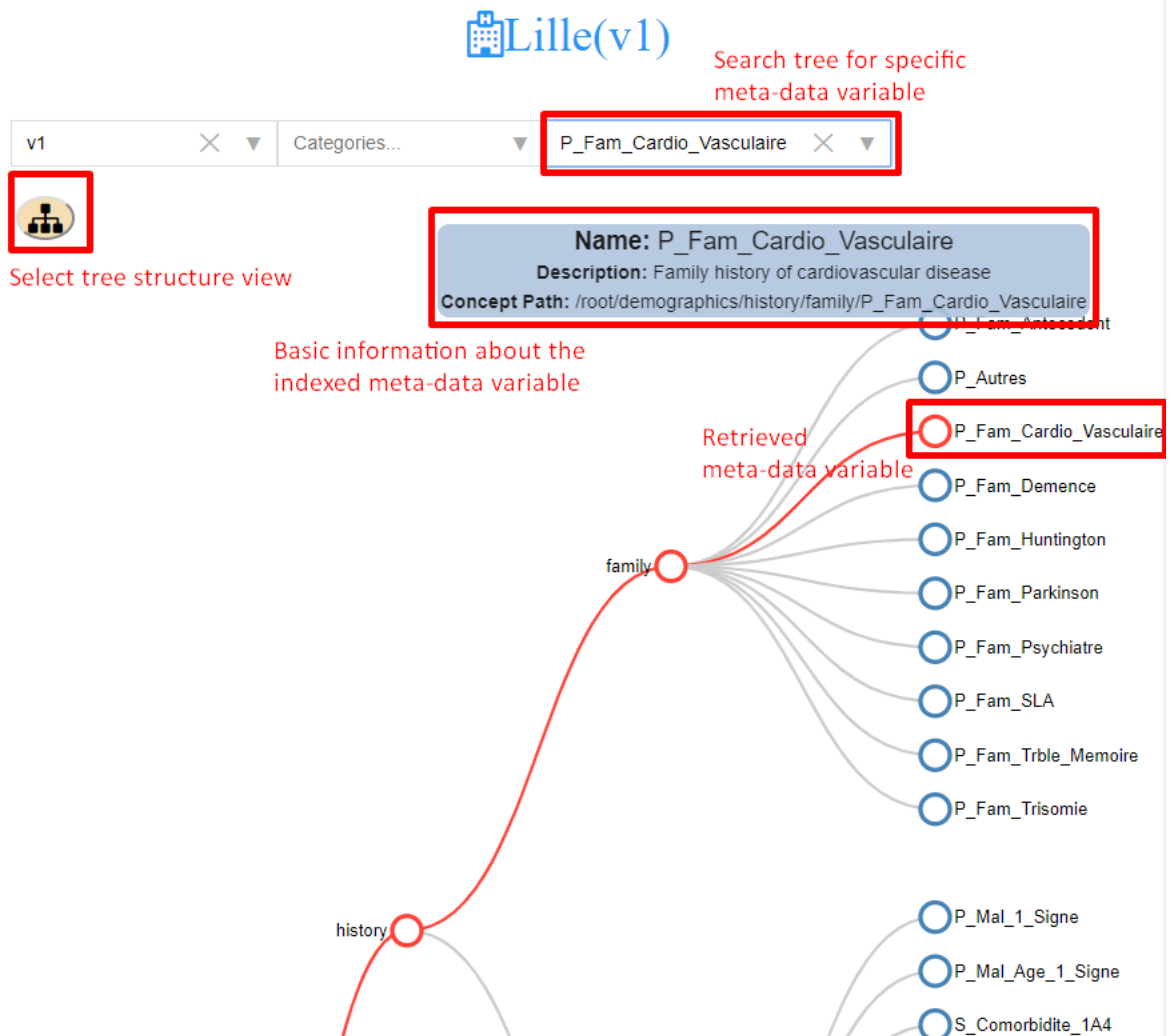
Download meta-data variables in json format ^

File :EHR_General.csv
Name :P_Fam_Antecedent
Code :P_Fam_Antecedent
Type :numeric
Values :
Measure Unit:
Can Be Null:
Description :Yes if there are family history or no if not
Comments :
Concept Path :/root/history/family/P_Fam_Antecedent
Methodology :

Meta-data variable details

P_Fam_Autres v



- View the hierarchy of the meta-data in an indexable tree structure



- View / Download / Index the results of the Quality Control Tool for the meta-data

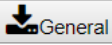
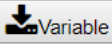
Lille(v1)

v1 × ▼ Categories... ▼ Search... ▼

  Report view selected

Report for the meta-data variables as a whole

Download general or individual report in csv file

 General  Variable

General Reports

Batch Number	File Name	Date Qc Tool Ran	QC Tool Version	Total Variables	Total Rows	Rows With Only Id	Rows With No Id	Rows With All Columns Filled	100% Coverage	80-100% Coverage	60-80% Coverage	40-60% Coverage	20-40% Coverage
1	EHR_Followup_Frozen1.csv	2019-03-14 13:09:48	0.1.1	26	14137	0	0	0	17	1	2	0	2
	EHR_Followup_Frozen2.csv	2019-03-14 13:10:14	0.1.1	13	14194	0	0	44	3	0	1	0	0
	EHR_General.csv	2019-03-14 13:10:50	0.1.1	17	1436	1	0	17	1	13	0	0	2
	EHR_General_Frozen2.csv	2019-03-14 13:11:25	0.1.1	3	1364	0	0	1364	3	0	0	0	0
	EHR_General_Frozen3.csv	2019-03-14 13:11:41	0.1.1	8	201	0	0	79	5	1	0	2	0

Report for each individual meta-data variable


Variable Reports

Variable Name	Batch Number	Type Declared	Type Estimated	Category Values	Number Of Category Values	Unique Values	Most Frequent Value	Count Of Most Frequent Value	Records Filled	Non Null Rows(%)	Mean	Stai Dev
P_Fam_Antecedent	1		nominal	'O', 'N', 'NR'	3	3	O	1049	1049	1435	99.93	
P_Fam_Autres	1		numerical			136	K	51	51	345	24.03	
P_Fam_Cardio_Vasculaire	1		nominal	'N', 'O', 'NR'	3	3	N	719	719	1435	99.93	
P_Fam_Demence	1		nominal	'N', 'O', 'NR'	3	3	N	742	742	1435	99.93	

- View the graphical representation of how variables are mapped to CDEs and using which rule

Lille(v1)

v1 Categories... Search...

 Mapping view selected

Meta-data variable
that maps to a CDE

Hospital Variables (v1) ---> CDE Variables

CDE

P_Lateralite	G→L D→R	Rule that should be applied for the transformation	handedness
(CurrentDate-P_Ne_Le) cc	P_Lateralite → handedness Mapping Rule: G→L D→R	roups: {"-50y"}, {"50-59y"}, {"60-69y"}, {"70-79y"}, {"+80y"}	agegroup
P_Ne_Le	(CurrentDate-P_Ne_Le		subjectage
	(CurrentDate-P_Ne_Le		subjectageyears
P_Sexe	stays the same		gender
S_Exa_UPDRS	stays the same		updrstotal
S_MMS	stays the same		minimentalstate
S_Moca	stays the same		montrealcognitiveassessment

Management Features (Login and appropriate role required)

Data Catalogue uses a keycloak instance setup for the MIP to authorize users. There are 3 role types available

- administrator - has access to everything
- pathology owner - has access to everything that is included in a pathology (CDEs_version,hospital,local_version)
- hospital owner - has access to everything that is included in a hospital (hospital,local version)

The extra actions that the user can do with appropriate role are the following:

- Create a new medical condition (administrator/pathology owner)
- Create/delete hospital (administrator/pathology owner/hospital owner)
- Create/edit/delete a meta-data for global model using the GUI or uploading a file (administrator/pathology owner)
- Create/edit/delete a meta-data for local hospitaldata model version using the GUI (administrator/pathology owner/hospital owner)

  Create new version

 Lille(v1)

v1  Categories...  Search... 



Download

P_Fam_Antecedent



P_Fam_Autres



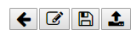
P_Fam_Cardio_Vasculaire



P_Fam_Demence



P_Fam_Huntington



 New Version for Lille

New Variable



Delete suggested meta-data variable

 P_Fam_Antecedent



File : EHR_General.csv

Name : P_Fam_Antecedent

Code : P_Fam_Antecedent

Type : numeric

Values :

Measure Unit:

Can Be Null:

Description : Yes if there are family his

Comments :

Concept Path : /root/history/family/P_Fam

Methodology :

Mapping Function :

Map CDE :

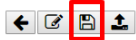
Change the details of the suggested meta-data variable

Save

Save the meta data variable to the current version

 P_Fam_Autres





Save all the version

 New Version for Lille

New Variable

Create a new variable from scratch

File Name

Name

Code

Type

Values

Measure Unit

Can Be Null

Description

Comments

Concept Path

Methodology


Mapping Function


Map CDE

Fill in all the information about the variable

save

Save the variable to the current version

 P_Fam_Antecedent

 P_Fam_Autres

- OR Create/edit/delete a meta-data for local hospital data model version uploading a file (administrator/pathology owner/hospital owner)



Upload file to create a new version

 New Version for Lille

New Variable

 P_Fam_Antecedent

 P_Fam_Autres

 P_Fam_Cardio_Vasculaire

 P_Fam_Demence

 P_Fam_Huntington



Description of the details of the meta-data variables,
that should be completed in the excel file

In order to create a new version the data have to be provided in a specific file format. The columns and the file name have to be kept unchanged. Download the XLSX template provided [below](#) and fill in the appropriate information and afterwards upload it in this platform. Fill in the rows with the variables' information as explained above and upload the file:

1. **csvFile**: The name of the dataset file the variable is in.
2. **name**: The name of the variable.
3. **code**: The variable's code.
4. **type**: The variable's type.
5. **values**: The variable's values. It may have an enumeration or a range of values. For enumerations, for every value please provide code and label in brackets. Example (for ADNI category): {"AD", "Alzheimer's Disease"}, {"MCI", "Mild Cognitive Impairment"}, {"CN", "Cognitively Normal"}. For range of values simply state the min and max values with '-' in between. Example (for MMSE Total scores): 0-30.
6. **unit**: The variable's measurement unit.
7. **canBeNull**: Whether the variable is allowed to be null or not: Y/N.
8. **description**: The variable's description.
9. **comments**: Comments about the variable's nature.
10. **conceptPath**: The variable's concept path. Example (for ApoE4): /root/genetic/polymorphism/apoe4.
11. **methodology**: The methodology the variable has come from. Example (for rs10498633_T): Iren-nmm-volumes.
12. **mapFunction**: The function that transforms the variable's value into the value of its corresponding CDE. Use a simple way to express values transformations. Example (for Handedness): 1->'R' 2->'L'. When value stays the same fill in with "stays the same".
13. **mapCDE**: The corresponding CDE's code.

XLSX template

Download

Download a sample excel file that
contains all the appropriate columns

Choose File No file chosen

Upload

Choose an excel file from local disc to upload

All previous uploads.

Show Files

Show all previous files that were uploaded

- Download a template file that contains the schema of the meta-data that should be completed (anyone logged in)



Upload file to create a new version

New Version for Lille

New Variable	
P_Fam_Antecedent	▼
P_Fam_Autres	▼
P_Fam_Cardio_Vasculaire	▼
P_Fam_Demence	▼
P_Fam_Huntington	▼



Description of the details of the meta-data variables,
that should be completed in the excel file

In order to create a new version the data have to be provided in a specific file format. The columns and the file name have to be kept unchanged. Download the XLSX template provided [below](#) and fill in the appropriate information and afterwards upload it in this platform. Fill in the rows with the variables' information as explained above and upload the file:

1. **csvFile:** The name of the dataset file the variable is in.
2. **name:** The name of the variable.
3. **code:** The variable's code.
4. **type:** The variable's type.
5. **values:** The variable's values. It may have an enumeration or a range of values. For enumerations, for every value please provide code and label in brackets. Example (for ADNI category): [{"AD", "Alzheimer's Disease"}, {"MCI", "Mild Cognitive Impairment"}, {"CN", "Cognitively Normal"}]
For range of values simply state the min and max values with '-' in between. Example (for MMSE Total scores): 0-30.
6. **unit:** The variable's measurement unit.
7. **canBeNull:** Whether the variable is allowed to be null or not: Y/N.
8. **description:** The variable's description.
9. **comments:** Comments about the variable's nature.
10. **conceptPath:** The variable's concept path. Example (for ApoE4): /root/genetic/polymorphism/apoe4.
11. **methodology:** The methodology the variable has come from. Example (for rs10498633_T): lren-nmm-volumes.
12. **mapFunction:** The function that transforms the variable's value into the value of its corresponding CDE. Use a simple way to express values transformations. Example (for Handedness): 1->'R' 2->'L' When value stays the same fill in with "stays the same".
13. **mapCDE:** The corresponding CDE's code.

XLSX template

Download

Download a sample excel file that
contains all the appropriate columns

Choose File No file chosen

Upload

Choose an excel file from local disc to upload

All previous uploads.

Show Files

Show all previous files that were uploaded

Installing / Getting started

General Architecture



Database
(PostgreSQL 9.5)



Back End
(Spring Boot 2.1)



Front End
(Angular 6.2)

Prerequisites

Required installed packages:

- Java 1.8 (Oracle)
- PostgreSQL 9.5
- Spring Boot 2.1
- Angular 6.2
- Angular CLI 6.2
- Typescript 2.9
- D3 5.7

Installation

Clone the repo:

```
git clone https://github.com/aueb-wim/DataCatalogue.git
```

After logging in PostgreSQL shell we create an empty database named datacatalog:

```
create database datacatalog;
```

Backend

Install Maven. In a terminal:

```
sudo apt install maven
```

Fill in the credentials that we use to login to PostgreSQL and the IP / port of the server at application.properties file in DataCatalogue:

- spring.datasource.username= yourUser
- spring.datasource.password= yourPassword
- server.port=8086
- server.address=172.16.10.138

Change the URL of the Authentication Entry Point in service/MIPSecurity.java line 150 if necessary. Run spring boot:

```
mvn spring-boot:run
```

This will initiate the back end services.

Frontend

Need to have Angular and Angular CLI installed. Having Angular already installed, inside the frontend directory run:

```
npm install
```

To configure frontend, change the following files with the correct URLs:

- frontend/proxy.conf.js
- frontend/proxy.conf.json
- frontend/src/app/shared/hospital.service.ts
- frontend/src/app/components/form-upload.component.ts
- frontend/src/app/components/form-upload-CDEs.component.ts

To initiate the frontend services, we use angular CLI:

```
cd DataCatalogue/frontend  
  
ng serve --host 0.0.0.0
```

For developers

Open the DataCatalogue project with any IDE (Intelij recommended). To run Spring boot, execute the class DataCatalogueSpringBootApplication. Frontend is initiated as said with ng serve.

HBP-MIP Data Quality Control Tool - Wiki

Github repository link

Here you could find some useful explanations and technical details about the the Data Quality Control features.

In GUI Manual you can find the basic operations of the Data Quality Control Tool, both for tabular and DICOM (MRI) data.

In Reports - Descriptions-and-Details page there is a thorough explanation of the Reports produced in pdf format, both for a csv dataset and an MRI dataset.

In Files Specifications, there is a short description about the "modified" Frictionless Data Table Schema JSON object, that we use for describing the schema of a csv dataset file. Also, there is a short description about the exported Excel file which contains the dataset's schema description in a format that is acceptable by the Data Catalogue.

In Data Validation & Cleaning functionality (CSV), we describe the Data Validation and Cleaning functionalities per datatype that are supported by the tool.

Installation

Linux

Prerequisites

- Ubuntu 18.04 or newer
- docker version 19 or newer
- docker-compose 1.22 or newer

Note: The User MUST be in the user group docker. To do that:

```
sudo gpasswd -a $USER docker
```

Installation through deb file

First download the proper deb package for your system version (currently the deb packages are for ubuntu 16.04, 18.04 and 20.04) then install the deb package in terminal by giving the below command (The deb file can be):

```
sudo apt-get update
```

```
sudo dpkg -i path_to_deb_file
```

Selecting previously unselected package mipqctool.

(Reading database ... 32232 files and directories currently installed.)

```
Preparing to unpack mipqctool_4.0_amd64_ubuntu_20.04.deb ...
Unpacking mipqctool (4.0) ...
dpkg: dependency problems prevent configuration of mipqctool:
  mipqctool depends on python3-tk; however:
    Package python3-tk is not installed.
  mipqctool depends on libcairo2; however:
    Package libcairo2 is not installed.
  mipqctool depends on libpango-1.0-0; however:
    Package libpango-1.0-0 is not installed.
  mipqctool depends on libpangocairo-1.0-0; however:
    Package libpangocairo-1.0-0 is not installed.
  mipqctool depends on libgdk-pixbuf2.0-0; however:
    Package libgdk-pixbuf2.0-0 is not installed.
  mipqctool depends on libffi-dev; however:
    Package libffi-dev is not installed.

dpkg: error processing package mipqctool (--install):
  dependency problems - leaving unconfigured
Processing triggers for libc-bin (2.31-0ubuntu9.2) ...
Errors were encountered while processing:
  mipqctool
```

If some dependencies haven't been installed successfully.

```
sudo apt-get update --fix-missing
sudo apt-get install -f
```

Manual Installation

Required installed packages for Debian based distros

- python3, python3-pip, python3-tk
- cairo
- Pango
- GDK-PixBuf
- Git

To install the above packages, give in a terminal the below commands:

```
sudo apt-get update
```

```
sudo apt-get install build-essential python3-dev python3-pip python3-  
setuptools python3-wheel python3-cffi python3-venv python3-tk libcairo2  
libpango-1.0-0 libpangocairo-1.0-0 libgdk-pixbuf2.0-0 libffi-dev shared-  
mime-info git
```

In a terminal we run:

```
git clone https://github.com/HBPMedical/DataQualityControlTool.git  
  
cd DataQualityControlTool  
  
python3 -m venv venv/qctool  
  
source venv/qctool/bin/activate  
  
pip install --upgrade pip  
  
sh install.sh
```

Windows (with WLS 2)

A native windows version of DQC tool does not exist at the moment, but there is a hybrid solution by taking advantage the Windows Linux Subsystem (WLS). The WLS allows to run native Linux applications in Windows smoothly. Although currently the Windows 10 and WLS does not support Graphical applications out of the box, there are some walk around solutions by using the GWLS or the Xming software. The good news is that WLS version 2 will obtain a native support for Linux GUI applications in the near future, so it won't be necessary to use neither GWLS nor Xming.

Prerequisites

- Windows 10, version 1903 or higher.

- WLS version 2 (Windows Linux Subsystem).
- Ubuntu 20.03 - Installation through the Microsoft Store.
- Docker Desktop for WLS.
- GWSL app - Installation through the Microsoft Store.
- Xming (an alternative to GWSL - not recommended) - X server for Windows.

Note: The Data Quality Control tool could be run in Windows with WLS 1 without docker client, but it won't have the data mapping functionality.

Setup WLS

Step 1 - Enable the Windows Subsystem for Linux

Open PowerShell as Administrator and run:

```
dism.exe /online /enable-feature /featurename:Microsoft-Windows-Subsystem-Linux /all /norestart
```

If you wish to only install WSL 1, you can now restart your machine and move on to Step 6 - Install your Linux distribution of choice

Step 2 - Check requirements for running WSL 2

To update to WSL 2, you must be running Windows 10.

- For x64 systems: Version 1903 or higher, with Build 18362 or higher.
- For ARM64 systems: Version 2004 or higher, with Build 19041 or higher.
- Builds lower than 18362 do not support WSL 2. Use the Windows Update Assistant to update your version of Windows.

To check your version and build number, select Windows logo key+ R, type winver, select OK.

Step 3 - Enable Virtual Machine feature

Open PowerShell as Administrator and run:

```
dism.exe /online /enable-feature /featurename:VirtualMachinePlatform /all /norestart
```

Restart your machine to complete the WSL install and update to WSL 2.

Step 4 - Download the Linux kernel update package

- 1 Download the latest package: WSL2 Linux kernel update package for x64 machines
- 2 Run the update package downloaded in the previous step. (Double-click to run - you will be prompted for elevated permissions, select 'yes' to approve this installation.)

Step 5 - Set WSL 2 as your default version

Open PowerShell as Administrator and run:

```
wsl --set-default-version 2
```

Step 6 - Install Ubuntu 20.04 LTS distribution

- 1 Open the Microsoft Store and install Ubuntu 20.04 LTS Linux distribution.
- 2 Set your distribution version to WLS 2

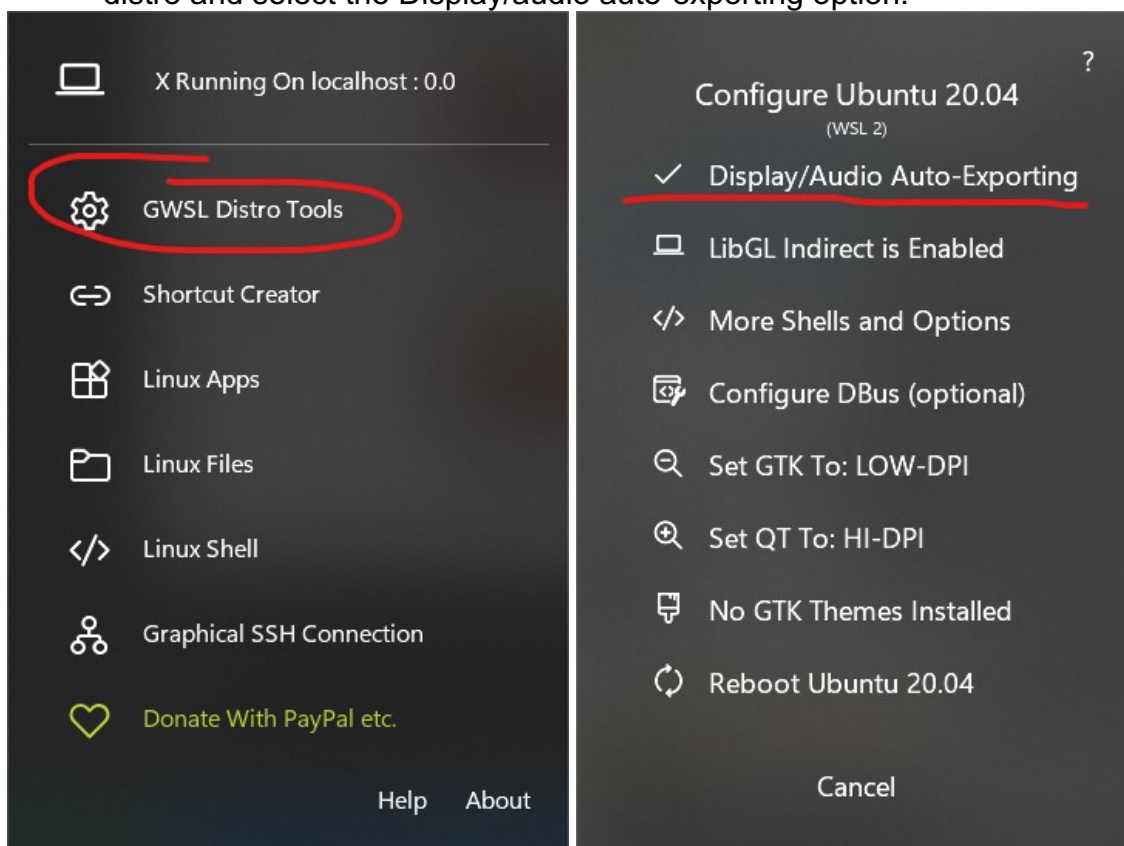
Open PowerShell as Administrator and run:

```
wsl --set-version Ubuntu-20.04 2
```

The above instructions are taken from the Official Microsoft site. Please, refer to it for more details or troubleshooting.

Setup GWLS

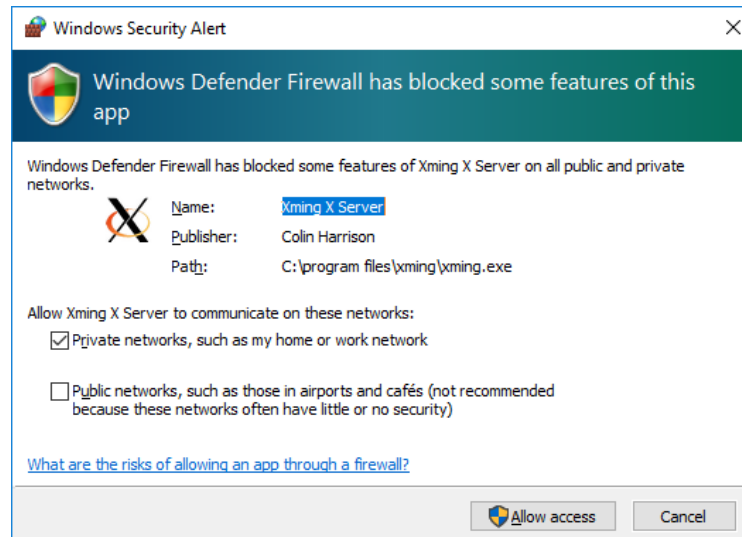
- 1 Install the application through the Microsoft Store
- 2 When installing, in Windows Defender Firewall pop up, please check both boxes for Private and Public networks.
- 3 In GWSL click the GWSL Distro Tools and select the previously installed Ubuntu distro and select the Display/audio auto-exporting option.



- 4 In the same distro menu, click the Reboot option, so the changes would take effect.

Setup Xming (case of not using the GWLS)

- 1 When installing or running Xming for the first time, in Windows Defender Firewall pop up, please check both boxes, for Private and Public networks.



- 2 In the Xming shortcut properties, edit the target field by adding -ac at the end. An example of the final string maybe will be like this: "C:\Program Files

`(x86)\Xming\Xming.exe" :0 -clipboard -multiwindow -ac`

Setup of Docker Desktop

- 1 Start Docker Desktop from the Windows Start menu.
- 2 From the Docker menu, select Settings > General.
- 3 Select Use WLS2 based engine check box.
- 4 Click Apply & Restart.
- 5 When Docker Desktop restarts, go to Settings>Resources>WLS Integration. The Docker-WSL integration will be enabled on your default WSL distribution.
- 6 Add your ubuntu user to the docker user group. To do that, open an Ubuntu terminal and give `sudo gpasswd -a $USER docker`

For more details or troubleshooting, please refer to the official docker webpage.

Installing the DQC tool

Please refer to the Linux Installation section, ignoring the prerequisites.

Running the DQC tool

- 1 Start the GWLS (or Xming) from Windows' Start Menu
- 2 (Case of Xming) Set up the X Server by giving the following command in WLS terminal


```
export DISPLAY=$(awk '/nameserver / {print $2; exit}' /etc/resolv.conf  
2>/dev/null):0
```

```
export LIBGL_ALWAYS_INDIRECT=1
```

3 Launch the application by giving in the Ubuntu terminal the command:

```
qctoolgui
```

MacOS

Prerequisites

- Docker Desktop
- HomeBrew Package Manager
- Python 3.6 or newer
- Git
- Pango and libffi packages

For installing Docker Desktop please refer to Official Docker documentation page

To install HomeBrew open a terminal and give the following command:

```
/bin/bash -c "$(curl -fsSL  
https://raw.githubusercontent.com/Homebrew/install/HEAD/install.sh)"
```

For installing the prerequisites packages, in terminal we give:

```
brew install python3 pango libffi git
```

Manual installation

In a terminal we run:

```
git clone https://github.com/HBPMedical/DataQualityControlTool.git  
cd DataQualityControlTool  
python3 -m venv venv/qctool  
source venv/qctool/bin/activate  
pip install --upgrade pip  
sh install.sh
```

Files Specifications

MIP "Frictionless data" table-schema

For describing the schema of a csv file, we use a modified version of the Frictionless Data Table Schema specifications designed to be expressible in JSON. More specifically, we modified the Field object by adding an additional property named `MIPType`. This property can take the following values:

- text
- nominal
- numerical
- integer
- date

Those are the main Datatypes that are used in the HBP-MIP platform.

The modified schema of the Table Schema JSON object can be found [HERE](#).

CDEs Dictionary Excel file

CDEs Dictionary Excel columns

The spreadsheet **MUST** have the following columns with the same order:

- 1 mipname: The name of the variable.
- 2 mip_code: The variable's code.
- 3 mip_type: The variable's type.
- 4 mip_values: The variable's values. It may have an enumeration or a range of values. For enumerations, for every value please provide code and label in brackets. Example (for ADNI category): {"AD", "Alzheimer's Disease"}, {"MCI", "Mild Cognitive Impairment"}, {"CN", "Cognitively Normal"} For range of values simply state the min and max values with '-' in between. Example (for MMSE Total scores): 0-30.
- 5 unit: The variable's measurement unit.
- 6 description: The variable's description.
- 7 comments: Comments about the variable's nature.
- 8 conceptPath: The variable's concept path. Example (for ApoE4): /root/genetic/polymorphism/apoe4.
- 9 variable_lookup: list of alternative names separated by commas
- 10 enum_lookup: list of alternative enumerations in the form {code: actual value} separated by commas. Example: {"F": "Famme"}, {"M", "Homme"}.
- 11 domain: The main scientific domain category that the variable belongs to.

Data Catalogue Excel file

Necessary Conventions

!NOTE: the name of the file should be in the following format: "pathology name"CDEs"version number".xlsx (e.g. demencia_CDEss_v1.xlsx).

!NOTE: all the columns described below should be present in the uploaded file.

!NOTE: a template with the appropriate file name and columns is available here. In order to avoid naming inconsistencies, we encourage users to use the provided template and fill-in their data.

DC Excel columns

The spreadsheet **MUST** have the following columns with the same order:

- 1 csvFile: The name of the dataset file the variable is in.
- 2 name: The name of the variable.
- 3 code: The variable's code.
- 4 type: The variable's type.
- 5 values: The variable's values. It may have an enumeration or a range of values. For enumerations, for every value please provide code and label in brackets. Example (for ADNI category): {"AD","Alzheimer's Disease"}, {"MCI","Mild Cognitive Impairment"}, {"CN","Cognitively Normal"} For range of values simply state the min and max values with '-' in between. Example (for MMSE Total scores): 0-30.
- 6 unit: The variable's measurement unit.
- 7 canBeNull: Whether the variable is allowed to be null or not: Y/N.
- 8 description: The variable's description.
- 9 comments: Comments about the variable's nature.
- 10 conceptPath: The variable's concept path. Example (for ApoE4): /root/genetic/polymorphism/apoe4.
- 11 methodology: The methodology the variable has come from. Example (for rs10498633_T): Iren-nmm-volumes.

Data Validation & Cleaning functionality (CSV)

Validation & Cleaning functionality per datatype

The tool can perform data validation and cleaning to the following datatypes:

- numerical
- integer
- date
- nominal

We categorize the violations into two types:

- 1 Constraint violations
- 2 Datatype violations

Types of Constraint violations that are supported currently by the tool are:

- 1 minimum (for date, integer, numerical)
- 2 maximum (for date, integer, numerical)
- 3 enum (list of enumerations for nominal datatypes)

Datatype violation is the case when a value in a column, has a different datatype from the one that has been declared for that column in the dataset's schema json file.

Below we can see the constraint violations per datatype that currently are supported by the tool, and the corresponding suggestions for the data cleaning operation. Also, in another table, we can see the datatype violations per datatype and the corresponding suggestions for the data cleaning operation.

Numerical Type column

Constraint Violation

Constraint	Suggested replacement
Minimum	-> Null
Maximum	-> Null

Datatype Violation

wrong Datatype	Suggested replacement
date	-> Null
text	-> Null

Integer Type column

Constraint Violation

Constraint	Suggested replacement
Minimum	-> Null
Maximum	-> Null

Datatype Violation

wrong Datatype	Suggested replacement
numerical(float)	-> integer(numerical)
date	-> Null
text	-> Null

Date Type column

Constraint Violation

Constraint	Suggested replacement
Minimum	-> Null
Maximum	-> Null

Datatype Violation

wrong Datatype	Suggested replacement
numerical	-> Null
integer	-> Null
text	try infer(Date) else Null

Nominal Type column

Constraint Violation

Constraint	Suggested replacement
enum	try spell-correction* else Null

*We calculate the Levenshtein distances between the given mis-spelled value with all the enumerations declared in the data schema. We suggest as corrected value the enumeration with distance smaller or equal to 3.

Datatype Violation

wrong Datatype	Suggested replacement
numerical	-> Null
integer	-> Null

GUI Manual

The tool has a tab for each data type (tabular, DICOM) and the usage of the GUI is a straightforward procedure.

Data Validation and Cleaning



Data Validation

First, we must load the dataset csv file.

If we need to validate the data, we need to have a json file containing the table schema (metadata) of the dataset csv file. This json must follow either a modified frictionless data table-schema specifications or MIP's Data-Catalogue specification schema.

About frictionless schema, the modified schema of that json file can be found here. In short, it is a simple modification that adds `MIPType` property in the field json object. The acceptable values for this property are `text`, `numerical`, `integer`, `nominal` and `date`, depending on the original field object type property value.

Also, in the case where the dataset belongs to a certain MIP's pathology, there is the option to download the dataset's CDEs schema from MIP Data Catalogue. The user may save the schema file in a local drive as a json file.

For the report, there are two options file formats:

- pdf
- excel (xlsx)

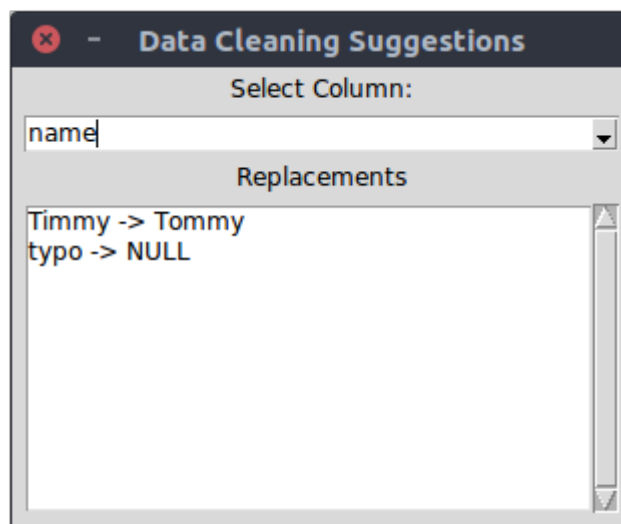
For more details about the content of the above files, please refer to the Report Files Descriptions and Details wiki section.

outlier threshold input field is related with the outlier detection for numerical variables of the incoming dataset. The way that the Data Quality Control tool handles the outlier detection of a certain numerical variable, is that first calculates the mean and the standard deviation based on the valid values of that column and then calculates the upper and the lower limit by the formula: $\text{upper_limit} = \text{mean} + \text{outlier threshold} * \text{standard deviation}$, $\text{lower_limit} = \text{mean} - \text{outlier threshold} * \text{standard deviation}$. If any value is outside those limits then it is considered as an outlier.

The report file will be saved in the given output folder, by clicking Create Report button. The name of the output file will be:

- <dataset>_report.pdf

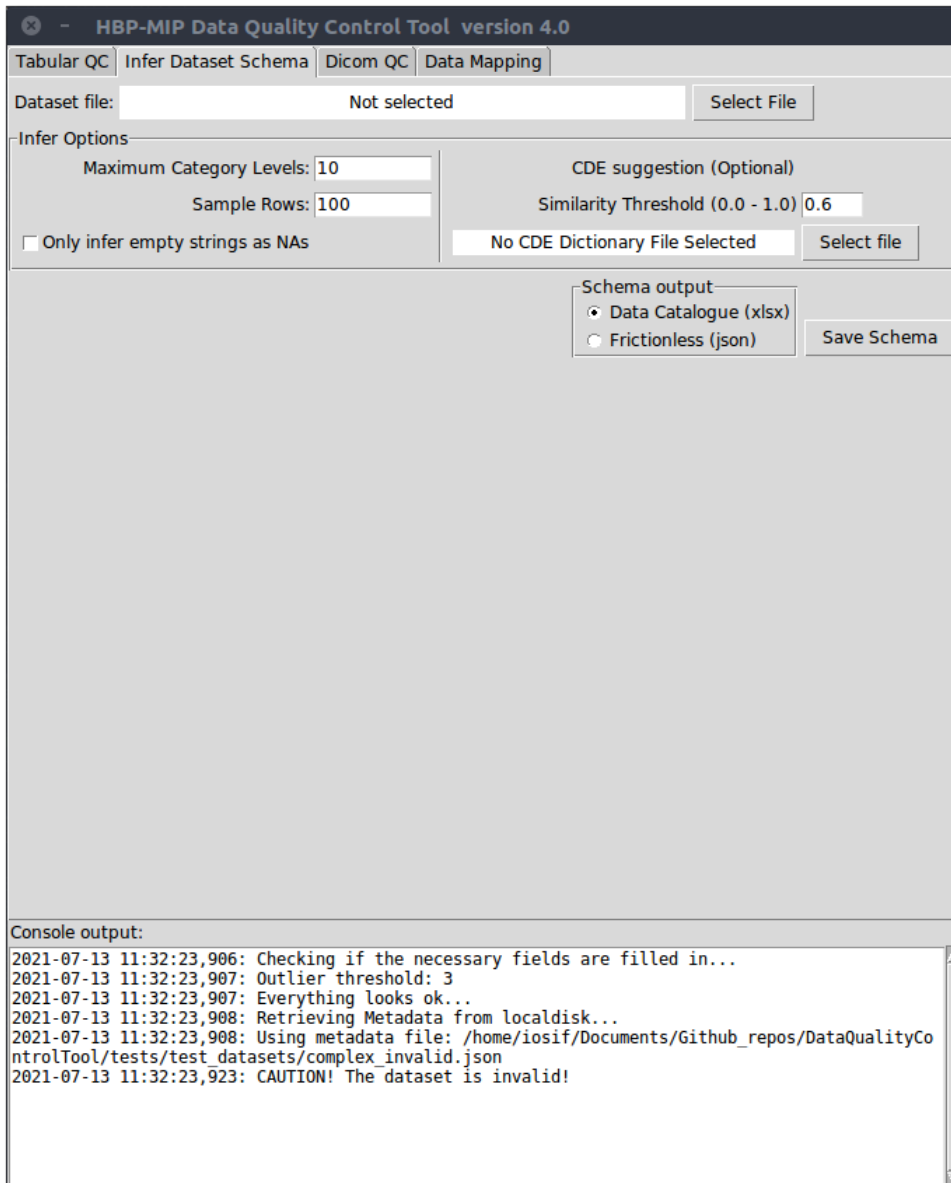
Data Cleaning



After reviewing the Cleaning Suggestions either by clicking the button Show cleaning suggestions or by referring the Suggested corrections section in the Data Validation report created in the previous step (Please refer to the Report Files Descriptions and Details wiki section for further details), we can proceed with the data cleaning operation by clicking the Perform Cleaning button. The cleaned dataset file will be saved in the report folder using the original dataset name with the addition of the suffix '_corrected'.

For further details about the validating and cleaning procedure of the DQC tool please refer to the Validation & Cleaning functionality per datatype wiki section.

Dataset Schema Inference



HBP-MIP Data Quality Control Tool version 4.0

Tabular QC | **Infer Dataset Schema** | Dicom QC | Data Mapping

Dataset file:

Infer Options

Maximum Category Levels:

Sample Rows:

☐ Only infer empty strings as NAs

CDE suggestion (Optional)

Similarity Threshold (0.0 - 1.0)

Schema output

☒ Data Catalogue (xlsx)

☐ Frictionless (json)

Console output:

```

2021-07-13 11:32:23,906: Checking if the necessary fields are filled in...
2021-07-13 11:32:23,907: Outlier threshold: 3
2021-07-13 11:32:23,907: Everything looks ok...
2021-07-13 11:32:23,908: Retrieving Metadata from localdisk...
2021-07-13 11:32:23,908: Using metadata file: /home/iosif/Documents/Github_repos/DataQualityControlTool/tests/test_datasets/complex_invalid.json
2021-07-13 11:32:23,923: CAUTION! The dataset is invalid!
  
```

In this tab we can infer a dataset's schema and save it to the local disk. The schema could be saved in two formats:

- 1 Frictionless spec json
- 2 Data Catalogue's spec Excel (xlsx) file, that can be used for creating a new CDEs pathology version.

In the infer option section, we give the number of rows that the tool will be based on for the schema inference. Also, we declare the maximum number of categories that a nominal MIPTYPE variable can have.

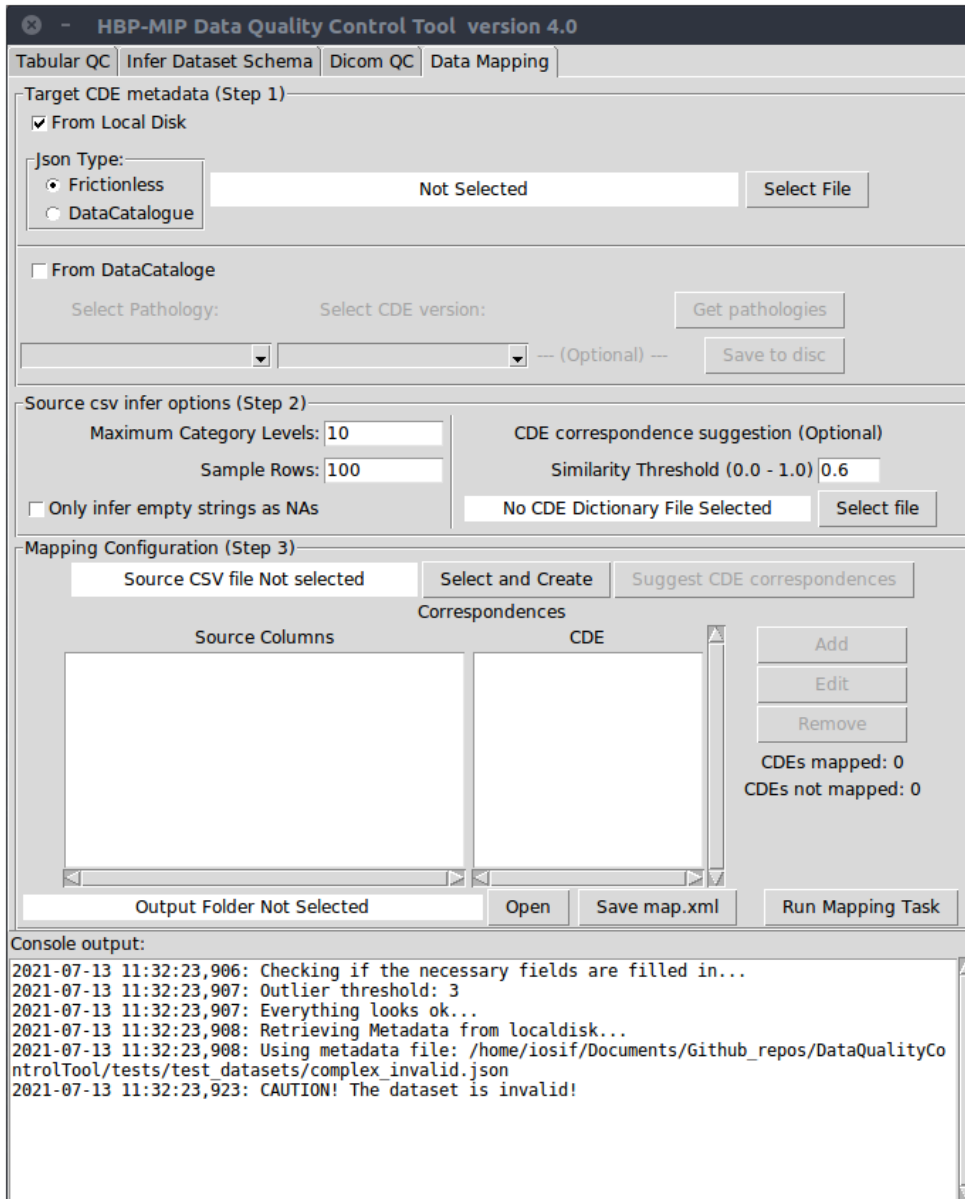
If we choose the Data Catalogue's excel as an output, the tool offers the option of suggesting CDEs variables for each column of the incoming dataset. This option is possible, only when a CDEs dictionary is provided. This dictionary is an excel file that contains information for all the CDEs variables that are included or will be included in the MIP (this dictionary will be available in the Data Catalogue in the near future). The tool calculates a similarity measure for each column based on the column name similarity (80%) and the value range similarity (20%). The similarity measure takes values between 0 and 1. In the field similarity threshold we can define the minimum similarity measure between an incoming column and a CDEs variable that need to be met in order the tool to suggest that CDEs variable as a possible correspondence. The tool stores those CDEs suggestions in the excel file in the column named CDEs and stores the corresponding concept path under the column conceptPath.

NOTE The DQC tool handles the below strings as null (NAs):

- " (empty string)
- #N/A
- #N/A N/A
- #NA
- -1.#IND
- -1.#QNAN
- -NaN
- -na
- 1.#IND
- 1.#QNAN
- N/A
- NA
- NULL
- NaN
- n/a
- nan
- null

If we want the DQC tool to handles those values as normal strings, we select the 'Only infer empty strings as NAs' box. In that case only the " (empty string) will be handled as a null value.

Data Mapping



The screenshot shows the 'Data Mapping' tab of the 'HBP-MIP Data Quality Control Tool version 4.0'. The interface is divided into several sections:

- Target CDE metadata (Step 1):**
 - ☒ From Local Disk: Includes a 'Json Type' dropdown (Frictionless selected), a 'Not Selected' text field, and a 'Select File' button.
 - ☐ From DataCatalogue: Includes 'Select Pathology:' and 'Select CDE version:' dropdowns, a 'Get pathologies' button, and a 'Save to disc' button.
- Source csv infer options (Step 2):**
 - Maximum Category Levels: 10
 - Sample Rows: 100
 - ☐ Only infer empty strings as NAs
 - CDE correspondence suggestion (Optional): Similarity Threshold (0.0 - 1.0) 0.6
 - No CDE Dictionary File Selected, with a 'Select file' button.
- Mapping Configuration (Step 3):**
 - Source CSV file Not selected, with 'Select and Create' and 'Suggest CDE correspondences' buttons.
 - Correspondences table with 'Source Columns' and 'CDE' columns.
 - Buttons: Add, Edit, Remove.
 - Status: CDEs mapped: 0, CDEs not mapped: 0.
 - Output Folder Not Selected, with 'Open', 'Save map.xml', and 'Run Mapping Task' buttons.
- Console output:**

```

2021-07-13 11:32:23,906: Checking if the necessary fields are filled in...
2021-07-13 11:32:23,907: Outlier threshold: 3
2021-07-13 11:32:23,907: Everything looks ok...
2021-07-13 11:32:23,908: Retrieving Metadata from localdisk...
2021-07-13 11:32:23,908: Using metadata file: /home/iosif/Documents/Github_repos/DataQualityControlTool/tests/test_datasets/complex_invalid.json
2021-07-13 11:32:23,923: CAUTION! The dataset is invalid!

```

If a hospital wants to participate in a MIP Federation of a certain pathology, the hospital's data must be harmonized accordingly. The goal of data mapping is transforming the hospital's local variables to a set of variables of a target CDEs pathology.

Step 1 - Target Schema Selection

Target CDE metadata (Step 1)

☒ From Local Disk

Json Type:

☒ Frictionless ☐ DataCatalogue

Not Selected

☐ From DataCatalogue

Select Pathology: Select CDE version:

-- (Optional) --

First, we select the target CDEs schema (Pathology). This can be loaded from a json file stored in a local disk or can be downloaded from MIP Data Catalogue API, which is a more convenient option.

Target CDE metadata (Step 1)

☐ From Local Disk

Json Type:

☒ Frictionless ☐ DataCatalogue

Not Selected

☒ From DataCatalogue

Select Pathology: Select CDE version:

dementia v4 -- (Optional) --

If we choose the latter, then we select the From DataCatalogue tick box and then click the Get pathologies button to retrieve all Pathologies metadata that are currently stored in the Data Catalogue. Then, we select first a Pathology from the drop-down menu, and then a CDEs version for the selected pathology from the drop down menu next to the first one. Optionally, we can save the selected CDEs schema in a Data Catalogue's spec json file.

Step 2 Source csv infer options

Source csv infer options (Step 2)

Maximum Category Levels: 10

Sample Rows: 100

☐ Only infer empty strings as NAs

CDE correspondence suggestion (Optional)

Similarity Threshold (0.0 - 1.0) 0.6

No CDE Dictionary File Selected

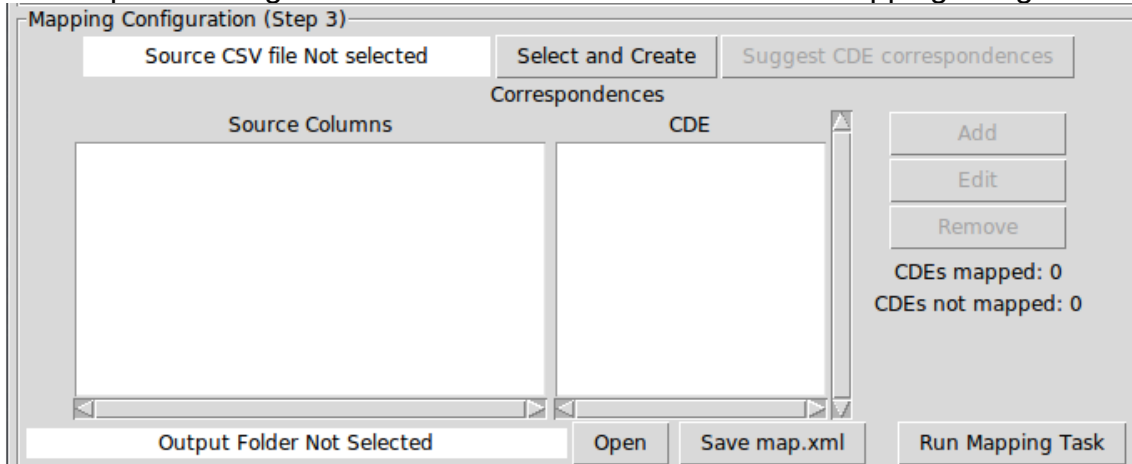
Currently, the tool does not support loading an existing source csv schema from a file, thus the source schema must be inferred first. As in the Dataset Schema Inference mentioned above, here we give the number of rows that the tool will based on for the source csv schema inference. Also, we declare the maximum number of categories that a nominal MIPTYPE variable can have.

Also there is the option to load a CDEs Dictionary, as in the Dataset Schema Inference. Here, with the use of a CDEs Dictionary the tool can suggest mappings between the incoming hospital variables and CDEs variables. Note Among the suggestions may be CDEs variables

that are not included in the selected Pathology from the previous step, in that case a warning message will be appeared in the console output.

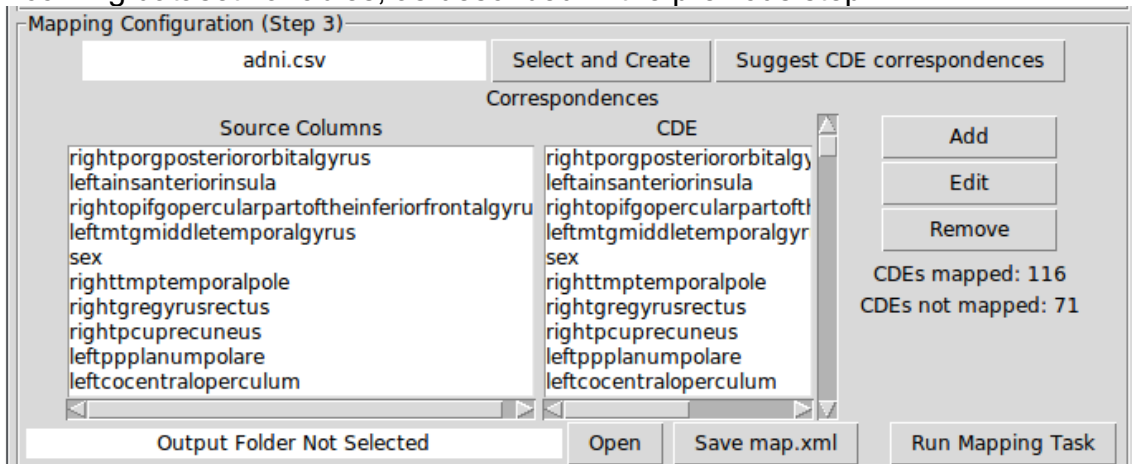
Step 3 Data Mapping Configuration

This step is the longest from the three and here the actual mapping design occurs.



The interface shows the 'Mapping Configuration (Step 3)' window. At the top, there are three buttons: 'Source CSV file Not selected', 'Select and Create', and 'Suggest CDE correspondences'. Below these are two large empty boxes labeled 'Source Columns' and 'CDE'. To the right of these boxes are three buttons: 'Add', 'Edit', and 'Remove'. Below these buttons, it says 'CDEs mapped: 0' and 'CDEs not mapped: 0'. At the bottom, there are four buttons: 'Output Folder Not Selected', 'Open', 'Save map.xml', and 'Run Mapping Task'.

First, we select the Source CSV file by clicking the **Select and Create** button. If we have loaded a CDEs Dictionary in the previous step, then the **Suggest CDEs correspondences** will be activated. If we click that button, the tool will try to find CDEs correspondences for the incoming dataset variables, as described in the previous step.



The interface shows the 'Mapping Configuration (Step 3)' window after running the suggestion task. The 'Source CSV file' button now shows 'adni.csv'. The 'Suggest CDE correspondences' button is now active. The 'Source Columns' box contains a list of brain regions: rightporgposteriororbitalgyrus, leftainsanteriorinsula, righttopifgopercularpartoftheinferiorfrontalgyru, leftmtgmiddletemporalgyrus, sex, righttmttemporalpole, rightgregyrusrectus, rightpcuprecuneus, leftppplanumpolare, and leftcocaloperculum. The 'CDE' box contains the same list of brain regions. To the right of these boxes are three buttons: 'Add', 'Edit', and 'Remove'. Below these buttons, it says 'CDEs mapped: 116' and 'CDEs not mapped: 71'. At the bottom, there are four buttons: 'Output Folder Not Selected', 'Open', 'Save map.xml', and 'Run Mapping Task'.

If we have run the CDEs suggestions and some CDEs correspondences (or mappings) have been found, those will be appeared in the CDEs selection box. In the 'Source Columns' box we can see the source variables for each mapping.

Next to these selection boxes there are three buttons:

- Add, for creating a new mapping
- Edit, for editing an existing mapping
- Remove, for deleting an existing mapping

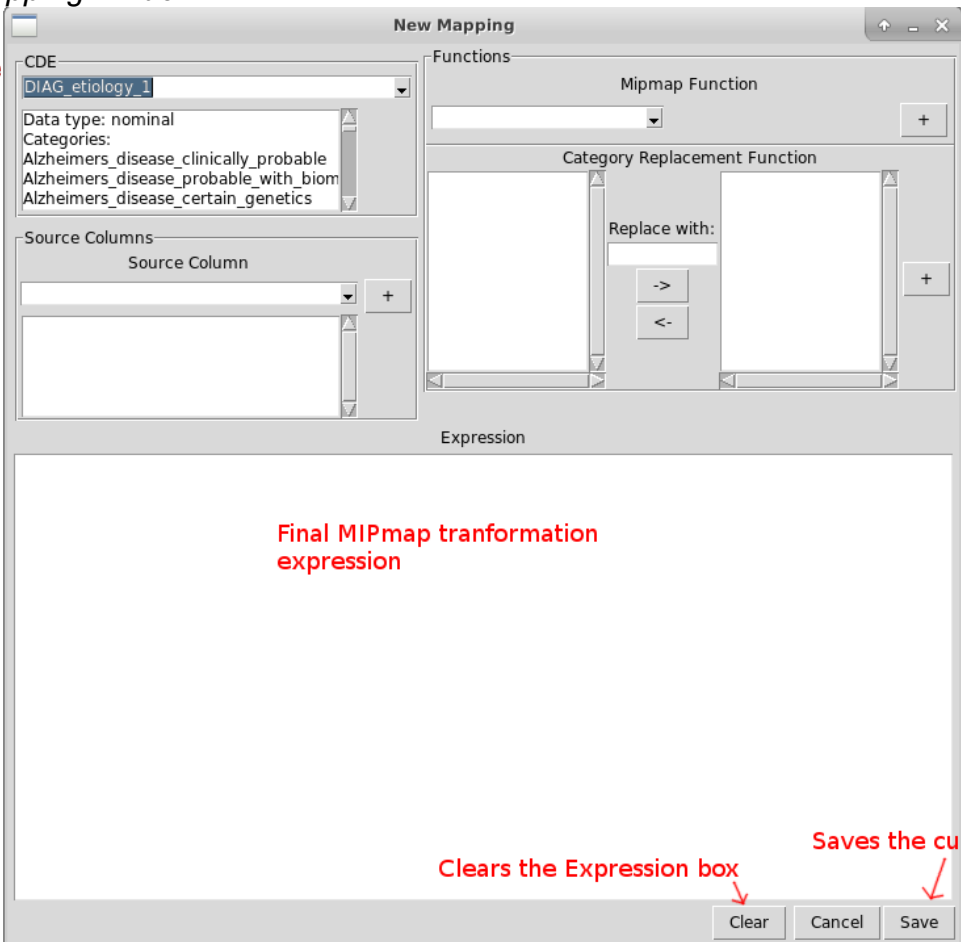
In the last two cases, to select a mapping we **MUST** select the CDEs name from the CDEs select box and **NOT** from the Source Column select box.

NOTE Due to GUI limitations, the scrolling in the mappings boxes by using the mouse wheel is not synchronized. For synchronized scrolling, please use the vertical scrolling bar.

Please note the labels below the buttons:

- CDEs mapped is the number of CDEs that has been mapped to one or more source variables.
- CDEs not mapped is the number of CDEs that are not mapped.

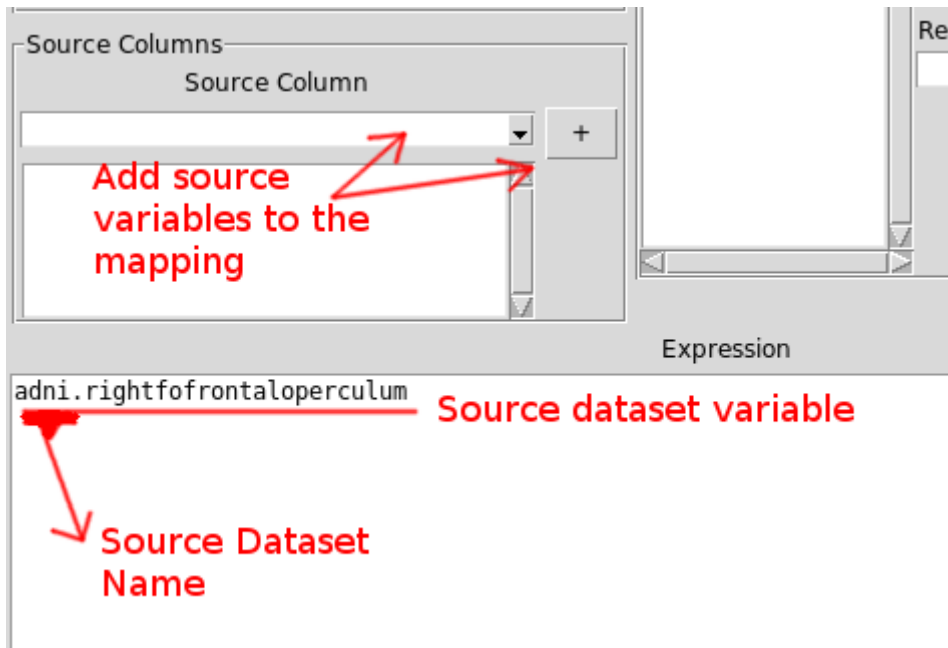
The new mapping window



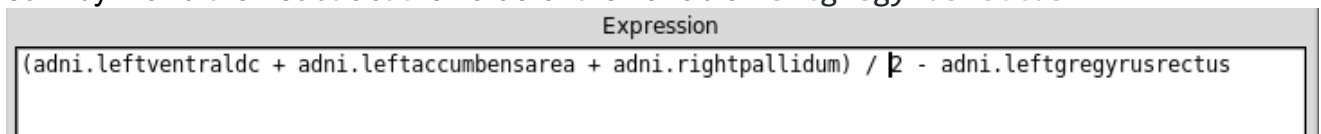
The screenshot shows the 'New Mapping' window with the following sections and annotations:

- CDE variable section (only one):** Points to the 'CDE' dropdown menu.
- Source variable section (multiple):** Points to the 'Source Columns' section.
- Transformation Function Section:** Points to the 'Mipmap Function' and 'Category Replacement Function' sections.
- Final MIPmap tranformation expression:** Points to the large 'Expression' text area.
- Clears the Expression box:** Points to the 'Clear' button.
- Saves the current mapping:** Points to the 'Save' button.

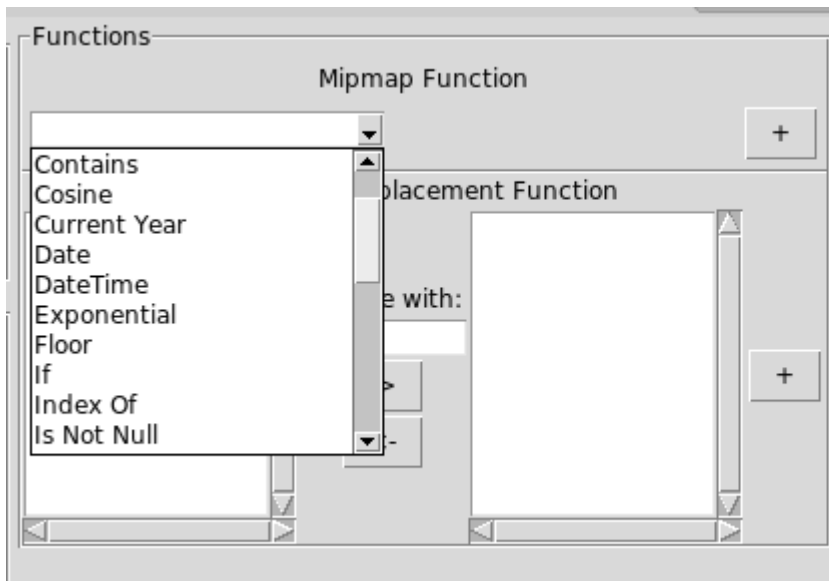
In the CDEs section, we select the CDEs variable that we are going to map. Then, on the Source Column section we can select variables from the source schema and by pressing the plus button we can add this variable to the expression box. Please note how the source column names are represented in the expression box below.



Below, we can see an example expression involving 3 source columns from the source dataset named 'adni'. The transformation is a quite simple one. We add the values of the variables `leftventraldc`, `leftaccumbensarea` and `rightpallidum` together and divide the sum by 2 and then subtract the value of the variable `leftgregyrusrectus`.



We can create more complicated expressions by inserting functions supported by MIPMap from the Functions section. By pressing the plus button, the selected function expression will be inserted at the expression box with some dummy arguments. We MUST replace those arguments with the ones that we want we use, for example a source variable or a number. Caution, the argument MUST be of compatible data type (str, number, date) with the ones that are supported by the selected function. Please refer to the MIPMap supported functions wiki section for further details.

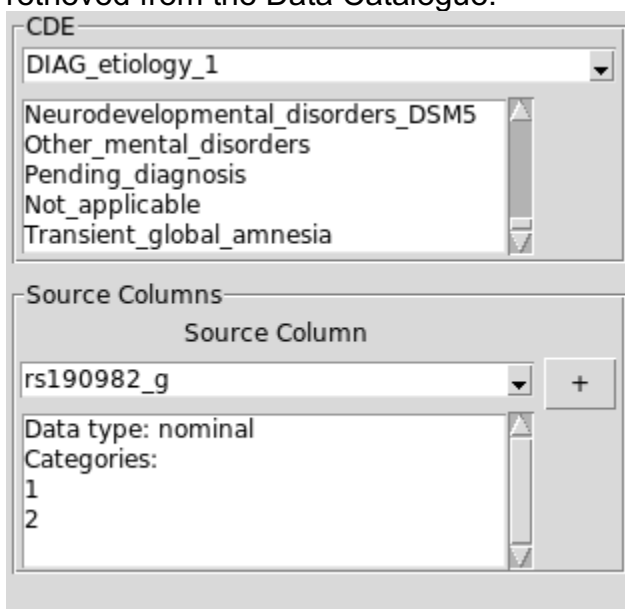


adds the selected function to the expression box

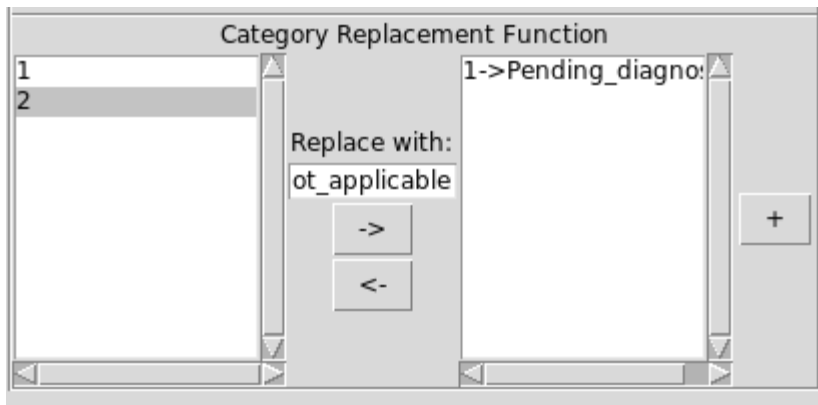
The Category Replacement Function is a special case of function that is used for in a mapping between two nominal variables (variables that has enumerated values). Let's see the following example to understand how it works.

Example of creating 1 to 1 mapping with nominal variables

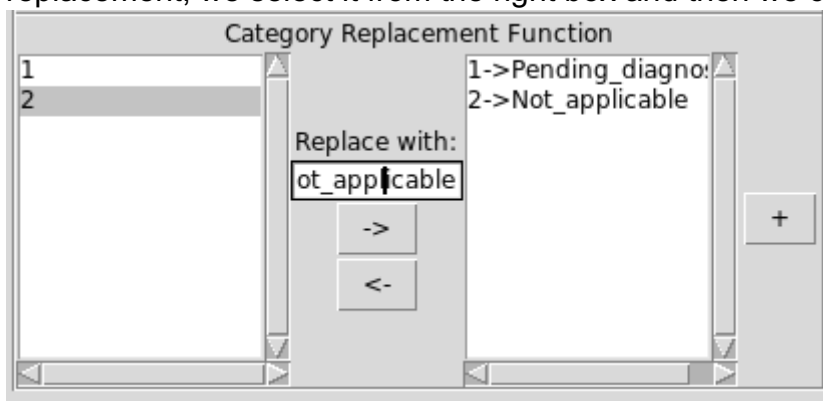
Let's try to create a new mapping between a nominal source variable and a nominal CDEs variable. In our example we used as a target schema the Dementia pathology version 5, retrieved from the Data Catalogue.



The CDEs variable that we are going to map is the DIAG_etiology_1, so we select it from the CDEs drop-down box. Then we select the nominal source variable rs190982_g. Please note, the mapping between these two variables doesn't not have any scientific point, we have just picked two nominal variables for the needs of this example. In the text boxes below the variable names, we can see the enumerations of each variable. Because we have a mapping between two nominal variables, we are going to use the Category Replacement Function.



First, we select the source variable enumeration (the number 1 in our case) from the left box that we want to replace, then we type in the box Replace with the target enumeration. In our example the target enumeration is Pending_diagnosis. Then click the -> button. After that we can proceed with the second enumeration (the number 2) which is going to be replaced with the target enumeration Not_applicable. Note, if we want to delete an enumeration replacement, we select it from the right box and then we click on the <- button.



After creating all the possible enumeration replacements, we can click the + button to produce the MIPMap expression. The outcome of the Category Replacement Function is consisted of multiple 'if' statements. Caution, it is not recommended changing it. After that, we can press the save button to save the current mapping.

Expression

```
if(adni.rs190982_g == "1", "Pending_diagnosis", if(adni.rs190982_g == "2", "Not_applicable", null()))
```

Clear Cancel Save

Example of editing an existing mapping

Now, let's edit an existing mapping that was created automatically by the Suggest CDEs correspondences functionality. We select the rightfofrontaloperculum CDEs mapping and we click the edit button.

Mapping Configuration (Step 3)

adni.csv

Select and Create
Suggest CDE correspondences

Source Columns

- leftmprgprecentralgyrusmedialsegment
- rightfofrontaloperculum
- lefttmptemporalpole
- rightofugoccipitalfusiformgyrus
- rightmfcmedialfrontalcortex
- rightputamen
- rightcaudate
- lefthippocampus
- rightioginferioroccipitalgyrus
- righttententorhinalarea

CDE

- leftmprgprecentralgyrusm
- rightfofrontaloperculum
- lefttmptemporalpole
- rightofugoccipitalfusiformg
- rightmfcmedialfrontalcorte
- rightputamen
- rightcaudate
- lefthippocampus
- rightioginferioroccipitalgyr
- righttententorhinalarea

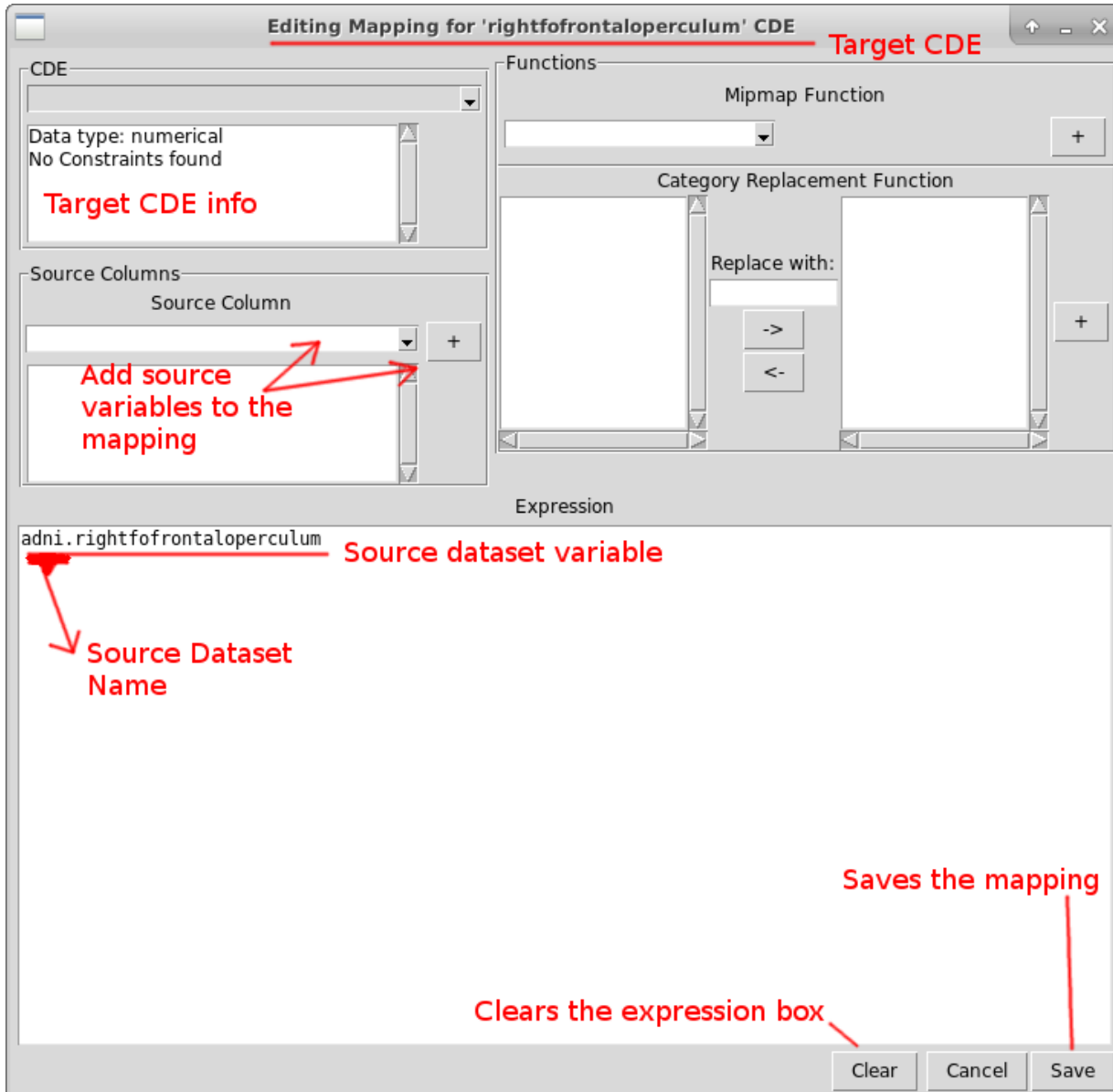
Add
Edit
Remove

CDEs mapped: 116
CDEs not mapped: 77

Output Folder Not Selected

Open
Save map.xml
Run Mapping Task

At the editing window, we observed that the CDEs section is grey out.



Editing Mapping for 'rightfofrontaloperculum' CDE Target CDE

CDE
 Data type: numerical
 No Constraints found
Target CDE info

Source Columns
 Source Column
 Add source variables to the mapping

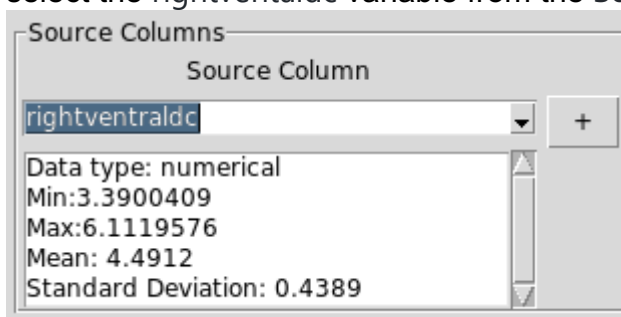
Functions
 Mipmap Function
 Category Replacement Function
 Replace with:
 ->
 <-

Expression
 adni.rightfofrontaloperculum Source dataset variable
Source Dataset Name

Clears the expression box Saves the mapping

Clear Cancel Save

At the moment, we observed that only one source variable is participating in the mapping and its value will be assigned to the target CDEs unchanged. Let's assume that the final value for the CDEs rightfofrontaloperculum will be the result of two source variables. To do so, we select the rightventraldc variable from the Source Column section click the + button.

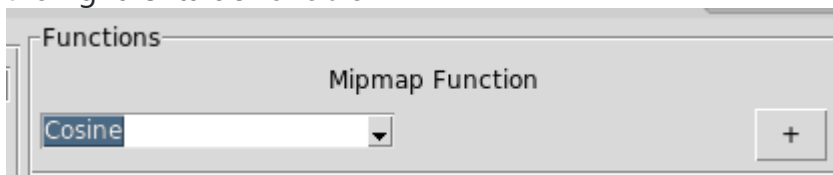


Source Columns
 Source Column
 rightventraldc
 Data type: numerical
 Min: 3.3900409
 Max: 6.1119576
 Mean: 4.4912
 Standard Deviation: 0.4389

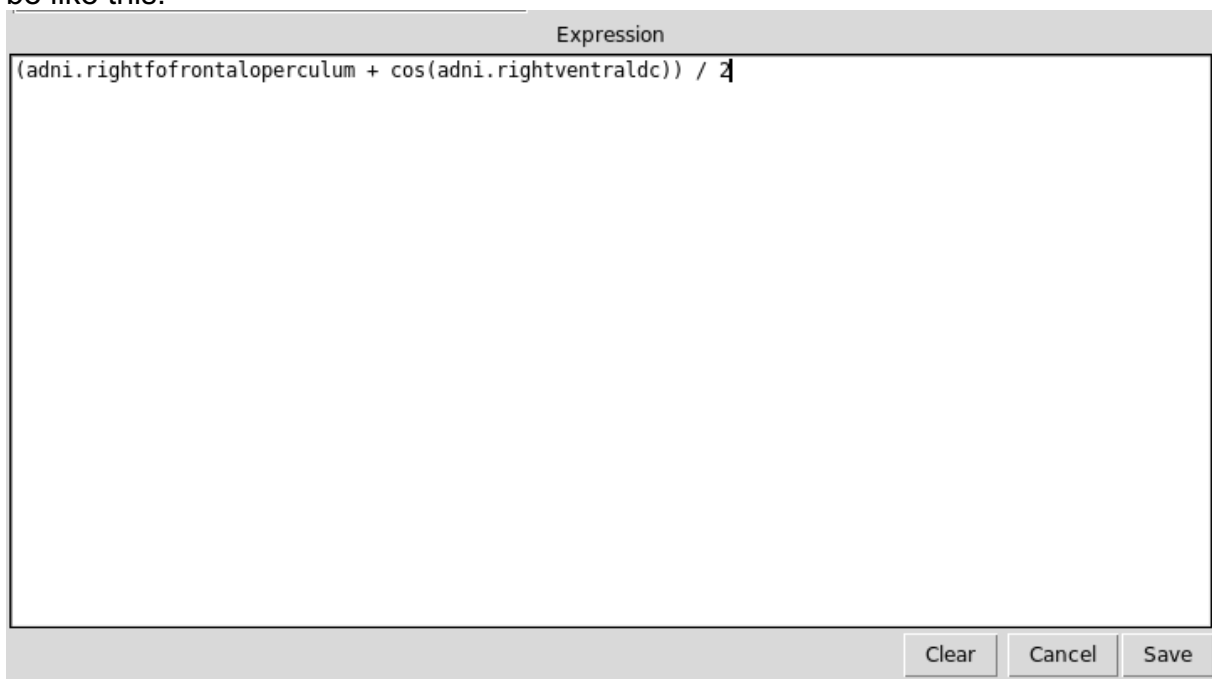
Now, let's say that we want the final value of the CDEs will be the result from the following formula:

$(\text{rightfofrontaloperculum} + \cos(\text{rightventalcdc})) / 2$

To do so, we select the cosine function from MIPMap Function and press the + button to add the function expression to the Expression box. Note, that the expression will be insert in the position that the text cursor currently is. After that, we delete the x default argument from the cosine expression. Leave the text cursor inside the parentheses. Then, we click on the + button at the Source Column section which previously selected the rightventalcdc variable.



After making the necessary edits in the Expression box, the final MIPMap expression would be like this:



We click the save button to save the modified mapping.

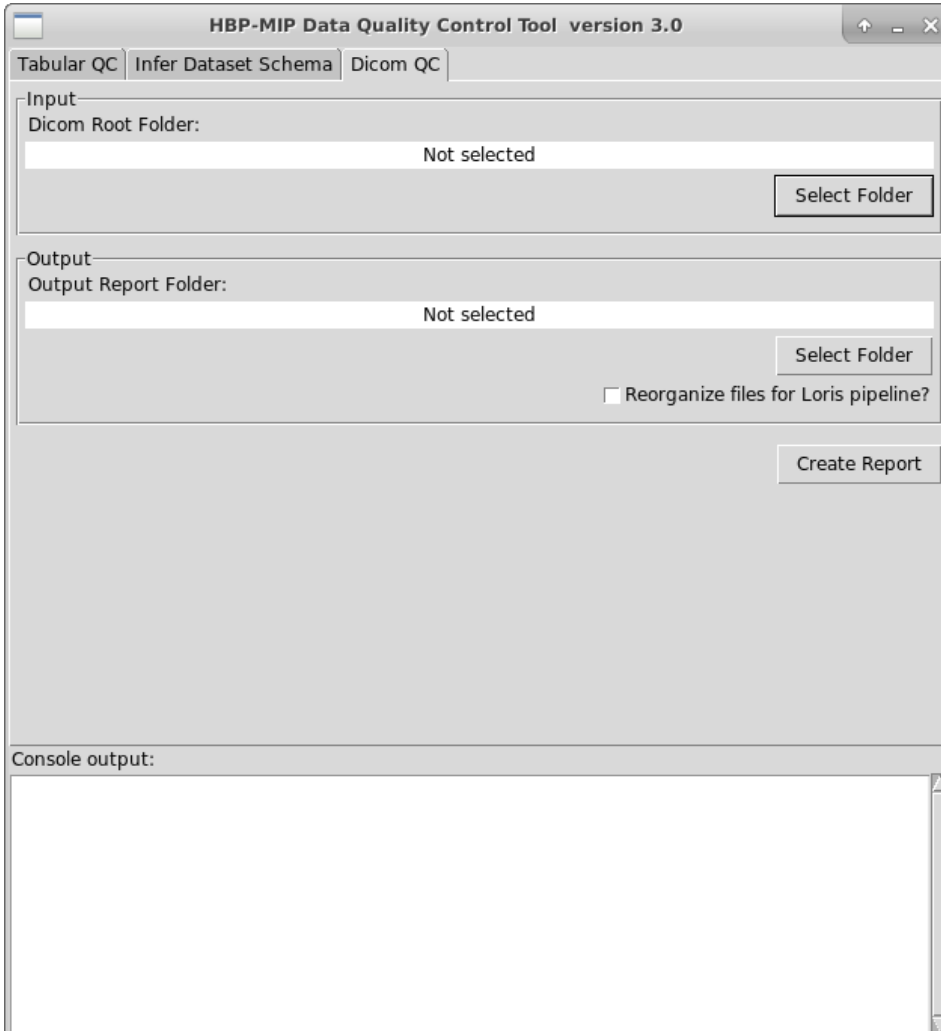
Final step - Executing the mapping / Data Transformation

If we are satisfied with the mappings, we can proceed to the actual data transformation. We select the local folder that we want to save the transformed csv file. NOTE, we must have WRITE rights on the selected folder. Then, we click the Run Mapping Task button to execute the transformation.

There is the option to save the mappings to an XML file compatible with the MIPMap format if the user wants to edit the mappings by using the MIPMapGUI application and execute the data transformation there.

NOTE The Data Transformation is done by a dockerized MIPMap, a software written in java, on the background.

DICOM Tab



Data Validation

We select the Dicom Root Folder where all the DICOM are stored. It is assumed that for each patient there is a subfolder containing all the MRI dcm files, note that a patient could have more than one MRI. Then, we select the Output Report Folder where the report files will be placed. If the folder does not exist, the tool will create it. Then, we press the Create Report button.

The tool creates in the <report folder>, the pdf report file dicom_report.pdf and, depending on the results, also creates the following csv files:

- validsequences.csv
- invalidsequences.csv
- invaliddicoms.csv

- notprocessed.csv
- mri_visits.csv

The above files are created even if no valid/invalid sequences/dicom files have been found. In such case, the files will be empty. Detailed description for the content of these files can be found on the Report Files - Description and Details wiki section.

Data Cleaning

If we want to filter out the invalid MRI sequences and reorganize the dcm files of the valid MRIs in a suitable folder structure for importing them into LORIS-for-MIP, repeat the previous step and select the Reorganize files for Loris pipeline check button.

For the LORIS pipeline the dcm files are reorganized and stored in a folder structure `<output_report_folder>/<patientid>/<patientid_visitcount>`. All the dcm sequence files that belong to the same scanning session (visit) are stored in the common folder `<patientid_visitcount>`.

MIPMap supported functions

label	expression	Description
Abs	<code>abs(x)</code>	Returns the absolute value of the same type / Returns the Modulus for Complex
Append	<code>append(str1, str2)</code>	Appends second parameter <code>str2</code> to first parameter <code>str</code> as string
Ceiling	<code>ceil(x)</code>	The smallest integer above the number <code>x</code>
Contains	<code>contains(str, subStr)</code>	Returns True if <code>str</code> contains the <code>subStr</code> at least once
ContainCount	<code>containCount(str, subStr)</code>	Returns the number of occurrences of <code>subStr</code> within <code>str</code>
Cosine	<code>cos(x)</code>	Cosine of angle <code>x</code>
Current Year	<code>currentYear()</code>	Returns current year
Date	<code>date()</code>	Returns current date
DateTime	<code>datetime()</code>	Returns current date in datetime format
Exponential	<code>exp(x)</code>	The result of the exponential function (e^x)
Floor	<code>floor(x)</code>	The smallest integer below that number <code>x</code>
If	<code>if(cond, trueval, falseval)</code>	The if function: <code>trueval</code> will be returned if <code>cond > 0</code> or True, and <code>falseval</code> if <code>cond <= 0</code> or False

label	expression	Description
Index Of	<code>indexof(str, subStr)</code>	Returns the position of the first occurrence of the subStr in the str.
Is Not Null	<code>isNotNull(arg)</code>	Tests if the arg is not null
Is Null	<code>isNull(arg)</code>	Tests if the arg is null
Is Numeric	<code>isNumeric(str)</code>	Test if the str is numeric
Length	<code>len(str)</code>	Returns the length of a string
Log	<code>log(x)</code>	Logarithm base 10
Ln	<code>ln(x)</code>	Natural logarithm
Modulus	<code>mod(x,y)</code>	Calculates the modules of $x \% y$ of the arguments
New Id	<code>newId()</code>	Gets the next number in a sequence
Null	<code>null()</code>	Returns null value
Power	<code>pow(x,y)</code>	Computes the y-th power of a number x (x^y)
Replace	<code>replace(str, text1, text2)</code>	In the first argument str , find all occurrences of the text1 and and replace them with text2
Round,	<code>round(x,[y])</code>	The closest integer to the argument, the second argument is optional and refers to decimal places
Sine	<code>sin(x)</code>	The Sine of angle x
Square Root	<code>sqrt(x)</code>	The Squared root of x
Substring	<code>substring(str, start, [end])</code>	Extracts a substring of str, start is starting index, end is optional for ending index
Tangent	<code>tan(x)</code>	Tangent of angle x
To Date	<code>todate(str, pattern)</code>	Converts a str to date format given pattern, e.g. "DD/MM/YYYY"
To Double	<code>todouble(x)</code>	Returns the double value of x
To Integer	<code>toint(x)</code>	Returns the integer value of x
To Lowercase	<code>tolower(str)</code>	Converts the str to lower case letters
To String	<code>tostring(data)</code>	Converts data type to string

label	expression	Description
To Timestamp	<code>totimestamp(str, pattern)</code>	Converts a str to datetime format given pattern, e.g. "DD/MM/YYYY hh:mm:ss"
To Uppercase	<code>toupper(str)</code>	Converts the str to upper case letters

Report Files Descriptions and Details

Data Validation Report of a CSV file (PDF)

The Data Validation report has 2 major report types:

- 1 Report with dataset's overall statistics and rows percentages about data completion and validation.
- 2 Report for each dataset column with statistics about the values, validation results and suggestions for correcting the invalid data.

If Data Cleaning has been performed, then both reports are slightly different from the original ones, containing additional information about the data cleaning operation.

Here are the templates of those 2 types of reports with explanations for each subsection

Dataset's Report

Section with dataset's overall statistics

This QC Report is created on: Date we run the data validation

Version of Data Quality Control Tool: version

File path: the file path location of the dataset csv file

Total number of rows: integer, Total number of columns: integer

Is metadata JSON file with table schema provided? Yes/No

Section with data validation and data completeness information

Data Cleaning performed? Yes/No

Number of rows with invalid values: the total number of rows having at least one column with invalid data

Rows Data Completeness Overall Statistics Table with distribution of number of rows against number of filled columns per row

Rows Data Validation Overall Statistics Table with distribution of number of rows against number of filled -with valid data- columns per row

Column's Report

Table with basic information about the column

Type	Text/Integer/Numerical/Nominal/Date
Total number of rows	column size in rows
Number of rows with data	how many filled values the column has
Completion percentage	The % ration of (filled values/column size)
Number of rows with constrain violation	how many rows contain a value that violates a specified value restriction (like min, max, enum etc) which is described in the dataset schema
Number of rows with datatype violation	how many rows contain a value that its type differs from the type specified for this particular column in the dataset's schema.
Data Cleansing applied?	Yes/No

Table with statistics about the column values

The statistics vary depending on the column datatype.

Statistic table for Integer Datatype

statistic field	explanation
Mode	most frequent value in the column
Number of occurrences for the most frequent value (mode)	
Minimum value	
Maximum value	
25% of records are below this value (limit value of the first quartile)	the middle number between the smallest number and the median of the set of values
50% of records are below this value (median)	the median of the set of values / 50% of the data lies below this point
75% of records are below this value (limit value of the third quartile)	the middle value between the median and the highest value of the set of values

Statistic table for Numerical Datatype

statistic field	explanation
Mean	the expected value or average
Standard deviation	measure of the amount of variation of a set of values
Minimum value	
Maximum value	
25% of records are below this value (limit value of the first quartile)	the middle number between the smallest number and the median of the set of values
50% of records are below this value (median)	the median of the set of values / 50% of the data lies below this point
75% of records are below this value (limit value of the third quartile)	the middle value between the median and the highest value of the set of values
Outlier upper bound	mean + 3 * standard deviation
Outlier lower bound	mean - 3 * standard deviation
Total number of outliers (outside 3 std.dev)	
Rows with outliers	list of [row number, value]

Statistic table for Date Datatype

statistic field	explanation
Mode	most frequent value in the column
Number of occurrences for the most frequent value (mode)	
Minimum value	
Maximum value	

Statistic table for Text Datatype

statistic field	explanation
Count of unique values (for text variables)	
Most frequent value	most frequent value in the column

statistic field	explanation
Number of occurrences for most frequent value	
5 most frequent values	
5 least frequent values	

Statistic table for Nominal Datatype

statistic field	explanation
Most frequent value	most frequent value in the column
Number of occurrences for most frequent value	
List of category values	
Number of categories	

Suggested corrections for datatype violations

If there are datatype violations and the tool succeeds to find corrections for some of them, then the suggested corrections are presented in a table with the following structure.

invalid value	proposed correction
---------------	---------------------

Suggested corrections for constraint violations

If there are constraint violations and the tool succeeds to find corrections for some of them, then the suggested corrections are presented in a table with the following structure.

invalid value	proposed correction
---------------	---------------------

Values that will be replaced with null

The invalid values that the tool has not been able to propose any corrections, will be replaced with null. Those invalid values are presented in this section in a list.

DICOM MRI Metadata report files

dicom_report.pdf - DICOM Metadata Validation Report

The report has 3 major sections:

- 1 Section with general information about the tool and execution parameters.
- 2 Section with General and Validation Statistics
- 3 Section with of MRI protocols

Here is the template of the MRI Sequences Report along with some comments and explanations.

General Information Section

This Report is created on: * Date the report is created* Version of Data Quality Control Tool: Version of the tool Main folder path where DICOM files are stored: folder path of the Dicom Root Folder Total subfolders scanned: number of subfolders that the tool has scanned and read the including dcm files, note there is an assumption of one subfolder per patient

General and Validation Statistics Section

Total number of patients with valid MRI sequences: Total number of valid MRI sequences: Total number of invalid MRI sequences: Total number of invalid DICOM (.dcm) files:

MRI protocols Section

Here, all the distinct MRI protocols (of all valid and invalid Sequences) are listed in a table. Those protocols are retrieved from the SeriesDescription Dicom tag.

validsequences.csv

If there are valid sequences, then the tool will create this csv file. This file contains all the valid MRI sequences that found in given DICOM folder with the following headers describing each sequence:

PatientID, StudyID, SeriesNumber, SeriesDescription, SeriesDate

The value of the sequence tags SeriesDescription and SeriesDate are derived from the headers in the dicom files - more specifically, the value of a sequence tag is the most frequent value of this particular tag found in the sequence's dicom files.

invalidsequences.csv

If there are invalid sequences the tool will create this csv file with the following headers:

PatientID, StudyID, SeriesNumber, Slices, Invalid_dicoms, SeriesDescription, Error1, Error2, Error3, Error4, Error5, Error6

- Slices is the number of dicom files that the current sequence is consist of (sum of valid and invalid dicoms).
- Invalid_dicoms is the number of invalid dicom files the current sequence.
- Error1 - Error6 is an error description that explains the reason why the sequence is characterized as 'invalid'

invaliddicoms.csv

If a dicom file does not have at least one of the mandatory tags as described in the MIP specification found here, then it will be characterized as 'invalid'. If there are invalid dicoms in the DICOM dataset, the tool will create this csv file with the following headers:

Folder, File, PatientID, StudyID, SeriesNumber, InstanceNumber, MissingTags

- MissingTags is a list of the missing mandatory DICOM tags.

notprocessed.csv

If in the given root folder are some files that the QC tool cannot process (not dicom files, corrupted dicom files etc), the tool will create this csv file with the following headers describing the location of those files:

Folder, File

mri_visits.csv

This file contains MRI visit information for each patient. This file is necessary for the HBP MIP DataFactory's Step3_B (not in use any more) and it has the following headers:

PATIENT_ID, VISIT_ID, VISIT_DATE

CLI Manual

Profiling and Validating a CSV dataset:

```
Usage: qctool csv <options> <csv file> <schema json>
```

This command produces a validation report for <csv file>.

The report file is stored in the same folder where <csv file> is located.

<schema json> file MUST be compliant with frictionless data table-schema specs(<https://specs.frictionlessdata.io/table-schema/>) or with Data Catalogue json format.

Options:

<code>--clean</code>	Flag for performing data cleaning. The cleaned
file will	
	be saved in the report folder.

```
-m, --metadata [dc|qc]  Select "dc" for Data Catalogue spec json
                        or "qc" for frictionless spec json.

-r, --report [xls|pdf]  Select the report file format.

-o, --outlier FLOAT      outlier threshold in standard deviations.

--help                  Show this message and exit.
```

Options further explanation

-o, --outlier, Outlier Threshold

This input field is related with the outlier detection for numerical variables of the incoming dataset. The way that the Data Quality Control tool handles the outlier detection of a certain numerical variable, is that first calculates the mean and the standard deviation based on the valid values of that column and then calculates the upper and the lower limit by the formula:

$$\text{upper_limit} = \text{mean} + \text{outlier threshold} * \text{standard deviation}$$
$$\text{lower_limit} = \text{mean} - \text{outlier threshold} * \text{standard deviation}$$
 If any value is outside those limits, then it is considered as an outlier.

The report file will be saved in the folder where the incoming dataset file is located.

Data Cleaning

After reviewing the Data Validation report created in the previous step (Please refer to the Data Validation Report wiki section for further details), we can proceed with the data cleaning operation. The cleaned dataset file will be saved in same folder where the incoming dataset is located by using the original dataset name with the addition of the suffix '_corrected'.

Inference of a CSV dataset's schema

Usage: `qctool infercsv <options> <csv file>`

This command infers the schema of the <csv file> it and stored in <output file>.

The <output file> either a json file following the frictionless data specs(<https://specs.frictionlessdata.io/table-schema/>) or an xlsx file

following MIP Data Catalogue's format.

Options:

<code>--max_levels INTEGER</code>	Max unique values of a text variable that below that will be inferred as nominal [default: 10]
<code>--sample_rows INTEGER</code> sample	Number rows that are going to be used as for inferring the dataset metadata (schema) [default: 100]
<code>--schema_spec [dc qc]</code> file	Select "dc" for Data Catalogue spec xlsx or "qc" for frictionless spec json.
<code>--CDEs_file PATH</code>	CDEs dictionary Excel file (xlsx)
<code>-t, --threshold FLOAT RANGE</code>	CDEs similarity threshold.
<code>--help</code>	Show this message and exit.

Options further explanation

--CDEs_file,--threshold, CDEs Dictionary support

If we choose the Data Catalogue's excel as an output, the tool offers the option of suggesting CDEs variables for each column of the incoming dataset. This option is possible, only when a CDEs dictionary is provided. This dictionary is an excel file that contains information for all the CDEs variables that are included or will be included in the MIP (this dictionary will be available in the Data Catalogue in the near future).

The tool calculates a similarity measure for each column based on the column name similarity (80%) and the value range similarity (20%). The similarity measure takes values between 0

and 1. With the option similarity threshold we can define the minimum similarity measure between an incoming column and a CDEs variable that need to be met in order the tool to suggest that CDEs variable as a possible correspondence. The tool stores those CDEs suggestions in the excel file in the column named CDEs and also stores the corresponding concept path under the column conceptPath.

Supported schema formats

The schema could be saved in two formats:

- 1 Frictionless spec json
- 2 Data Catalogue's spec Excel (xlsx) file, that can be used for creating a new CDEs pathology version.

DICOM MRI metadata validation

The metadata validation is performed following the HBP MIP's minimum metadata requirements for MRIs, which can be found [here](#).

Usage: `qctool dicom <options> <dicom folder> <report folder>`

This command produces a validation report for MRIs in <dicom folder>.

All MRI dcm files belonging to the same Patient MUST be in the same subfolder in <dicom folder>.

The validation report files are stored in <report folder>.

Options:

`--loris_folder <loris input folder>`

in LORIS input folder where the dcm files

<dicom folder> will be reorganized

`--help` Show this message and exit.

Options further explanation

<dicom folder>

is the root folder where all DICOM files are stored. It is assumed that each subfolder corresponds to one patient.

<report folder>

is the folder where the report files will be placed. If the folder does not exist, the tool will create it.

--loris_folder

folder path where the dcm files are reorganized for LORIS pipeline

For the LORIS pipeline the dcm files are reorganized and stored in a folder structure <loris_folder>/<patientid>/<patientid_visitcount>. All the dcm sequence files that belong to the same scanning session (visit) are stored in the common folder <patientid_visitcount>.

Output files

The tool creates in the <report folder>, the pdf report file dicom_report.pdf and, depending on the results, also creates the following csv files:

- validsequences.csv
- invalidsequences.csv
- invaliddicoms.csv
- notprocessed.csv
- mri_visits.csv

The above files are created even if no valid/invalid sequences/dicoms files have been found. In such case, the files will be empty. Detailed description for the content of these files can be found on the Report Files - Description and Details wiki section.

Developers Notes

Packaging (not applicable to version 4.0)

Windows

Prerequisites

- python3 latest version (not Microsoft Store version)
- include python executable in PATH system
- NSIS - Nullsoft Scriptable Installer for making a self-extract installer

Packaging

- 1 Clone or unzip the code in a Windows machine folder.
- 2 Create a virtual environment

```
python -m venv <name of virtualenv>
```

- 3 activate the virtualenv
- 4 install wheel package, the requirements.txt (remove the pytest and weasyprint from it) and pyinstaller

```
pip install wheel  
  
pip install -r requirements.txt  
  
pip install pyinstaller
```

- 5 update paths (included_files and pathex) in added_files list .spec file
- 6 run pyinstaller

```
pyinstaller qctoolgui_windows.spec
```

- 7 Zip the dist\MipDataQualityControlTool folder
- 8 use the NSIS to create a self-extract installer

MacOS

Prerequisites

- 1 install homebrew

```
/usr/bin/ruby -e "$(curl -fsSL  
https://raw.githubusercontent.com/Homebrew/install/master/install)"
```

- 2 Download and install the latest python3 version from the official python.org website. There are some issues with tkinter with the python versions installed through the apple store. Also, don't install python through homebrew.
- 3 Install cairo, Pango and GDK-PixBuf using Homebrew:

```
brew install cairo pango gdk-pixbuf libffi
```

Packaging

- 1 Clone or unzip the code in a Windows machine folder.
- 2 Create a virtual environment

```
python3 -m venv <name of virtualenv>
```

- 3 activate the virtualenv
- 4 install wheel package, the requirements.txt (remove the pytest and weasyprint from it) and pyinstaller

```
pip3 install wheel  
pip3 install -r requirements.txt  
pip3 install pyinstaller
```

- 5 update paths (included_files and pathex) in added_files list .spec file
- 6 run pyinstaller

```
pyinstaller qctoolgui_macos.spec
```

Ubuntu

- Docker client
- dh-virtualenv

Instructions in dh-virtualenv cookbook

<https://sourceforge.net/projects/vcxsrv/>