Paper RE02

Synthesized Real World Data: JSON and the Argonauts

Anil Sekhari, Sekharico Informatics Ltd, Edinburgh, UK
Dale Armstrong, Sekharico Informatics Ltd, Edinburgh, UK

ABSTRACT

When it comes to developing products and tools for life sciences organisations, it is difficult for tech-based companies to acquire clinical data.

An alternative is to use synthesized real world data (RWD). This paper demonstrates how to create CDISC SDTM domains from synthesized RWD. The journey starts with individual unstructured data-JSON files (in HL7 FHIR data exchange format) and how this data is first transformed to the OHDSI OMOP domains (common data model v5.4 data standard).

The final stage in the journey is a second transformation from OMOP domains into SDTM domains. All the transformations are performed using SAS® and this paper highlights the obstacles encountered and how to overcome them.

The hope is that this will serve as a useful paper to begin your own journey into using synthesized RWD in the development of products and tools.

INTRODUCTION

Life science companies are increasingly looking to utilize RWD to aid in the approval of their new treatments, being included in their submissions sent to regulatory agencies.

The project scope was to create CDISC SDTM datasets from Electronic Health Records (EHR) with the ultimate aim that the SDTM datasets could be used to create a control arm to use with an existing clinical trial (CDISCPILOT1) that is available on Github.

This project utilized synthesized EHR records from an application called Synthea, which is an open source synthetic patient generator. Synthea creates realistic patient health records in a variety of file formats (e.g. JSON, TXT and CSV) and a variety of data exchange standards. It includes demographic data to medical encounters/events to death.

The synthesized data is first transformed to OHDSI OMOP+ domains before the final SDTM datasets are produced.
OHDSI (Observational Health Data Sciences and Informatics) is an organization focused on developing open source tools, standards and expertise in discovering the value in large scale analytics within health data.
OMOP (Observational Medical Outcomes Partnership) common data model version 5.4 is an open community data standard. It was designed to specialize for analysis that varies in size, used standardized vocabularies (data dictionaries) and the re-use of existing vocabularies. The OMOP domains created by this project are classed as OMOP+ domains due to the extra variables that were added to each OMOP domain to provide traceability from the raw patient EHR JSON files.

All the conversions from individual patient EHR JSON files to OMOP+ domains and from the OMOP+ domains to SDTM were performed using PC SAS v9.4.

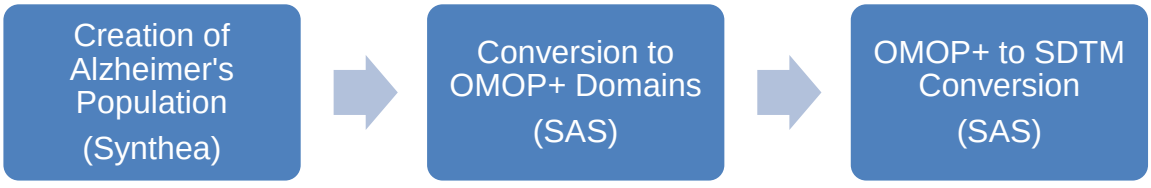

Fig 1. Summary of conversions

CREATION OF SYNTHESIZED PATIENT EHR RECORDS

The challenge was to create an Alzheimer's population and this was created using an open source synthetic patient generator called Synthea. The Synthea Customizer toolkit was used which allowed us to have a greater level of control over the synthesized patient EHR records.

The Synthea Customizer guided mode was all that was required, allowing the synthesis of HL7 FHIR R4 JSON patient EHR files. An input parameter of 150 patients was selected for the size of the population and clinical criteria of two Alzheimer's conditions using the setting "Any Of". Figure 2 lists all the settings applied and is a screenshot of the Synthea Customizer toolkit web interface.

The synthesis process is not quick, if population sizes are greater than 250 patients then be prepared to wait. Also it's worth doing a couple of test runs using very small numbers of patients to initially view the synthesized EHR patient files and scrutinize them to see if they are fit for purpose. Some patient EHR files had special characters in the filenames and due to time constraints we decided to not use those particular patient EHR files.

Selecting "Basic Setup" gives a download for the keep module, which should be downloaded to the same location as Synthea, as well as a command line for the terminal. Once the keep file is in position, run the command line to start creating patient EHR files.

There is also the properties file shown in figure 3, which once downloaded, can be further customized directly using a text editor. The properties file controls format of the patient EHR files e.g. CSV, JSON etc. The properties file location should be specified in the command line (see Figure 4). Figure 5 is an example of part of a section from one synthesized patient EHR file in the format of a JSON file.

Spend a bit of time installing the Synthea application from Github and exploring how best to use the application in order to meet project needs.

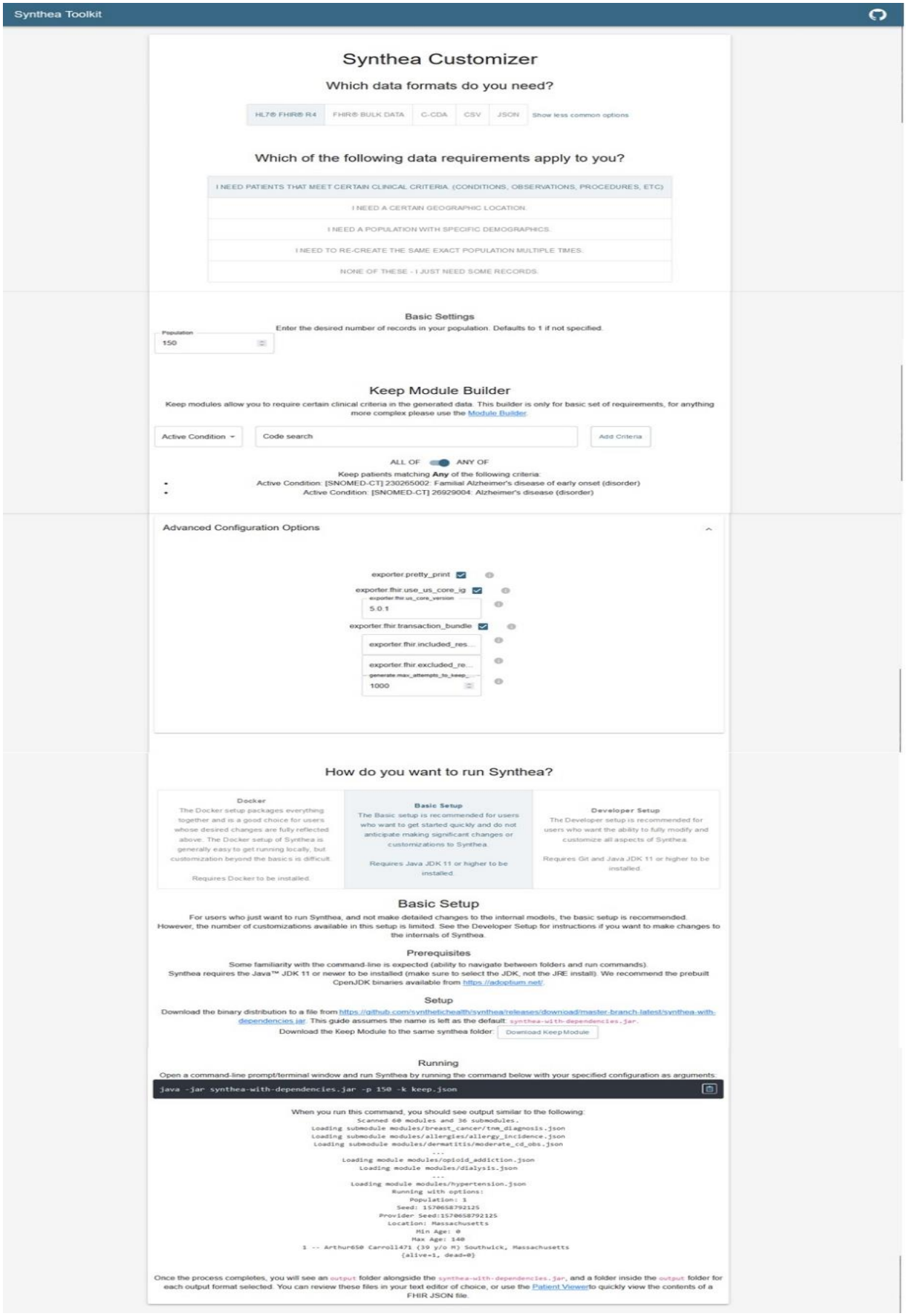

Fig 2: Synthea customizer

```
# Starting with a properties file because it requires no additional dependencies

exporter.baseDirectory = ./output/
exporter.use_uuid_filenames = false
exporter.subfolders_by_id_substring = false
# exporters that use XML or JSON can enable or disable 'pretty printing'
exporter.pretty_print = true
# number of years of history to keep in exported records, anything older than this may be filtered out
# set years_of_history = 0 to skip filtering altogether and keep the entire history
exporter.years_of_history = 10
# split records allows patients to have one record per provider organization
exporter.split_records = false
exporter.split_records.duplicate_data = false
exporter.metadata.export = true
exporter.ccda.export = false
exporter.fhir.export = true
exporter.fhir_stu3.export = false
exporter.fhir_dstu2.export = false
exporter.fhir.use_shr_extensions = false
exporter.fhir.use_us_core_ig = true
exporter.fhir.us_core_version = 5.0.1
exporter.fhir.transaction_bundle = true
# using bulk_data=true will ignore exporter.pretty_print
exporter.fhir.bulk_data = true
# included_ and excluded_resources list out the resource types to include/exclude in the csv exporters.
# only one of these may be set at a time, if both are set then both will be ignored.
# if neither is set, then all resource types will be included.
# note the Patient and Encounter resources will always be included, even if specifically listed as excluded here
exporter.fhir.included_resources =
exporter.fhir.excluded_resources =
exporter.groups.fhir.export = false
exporter.hospital.fhir.export = true
exporter.hospital.fhir_stu3.export = false
exporter.hospital.fhir_dstu2.export = false
exporter.practitioner.fhir.export = true
exporter.practitioner.fhir_stu3.export = false
exporter.practitioner.fhir_dstu2.export = false
exporter.encoding = UTF-8
exporter.json.export = false
exporter.json.include_module_history = false
exporter.csv.export = false
# if exporter.csv.append_mode = false, then each run will add new data to any existing CSVs. if false, each run will clear out the files and start fresh
exporter.csv.append_mode = false
# if exporter.csv.folder_per_run = false, then each run will have CSVs placed into a unique subfolder. if false, each run will only use the top-level csv folder
exporter.csv.folder_per_run = false
# included_files and excluded_files list out the files to include/exclude in the csv exporter
# only one of these may be set at a time, if both are set then both will be ignored
# if neither is set, then all files will be included
# see list of files at: https://github.com/synthetichealth/synthea/wiki/CSV-File-Data-Dictionary
# include filenames separated with a comma, ex: patients.csv,procedures.csv,medications.csv
# NOTE: the csv exporter does not actively delete files, so if Run 1 you included a file, then Run 2 you exclude that file, the version from Run 1 will still be present
```

Fig 3. Synthea properties file example

```
PS C:\Users\sekha> java -jar synthea-with-dependencies.jar -p 150 -c C:\Daleworkfiles\Contractwork\Prototype2\synthea.properties -k keep.json
```

Fig 4. Command line example

Fig 5. Section of a patient EHR Data-JSON file

OMOP+ DOMAINS PRODUCTION

The next stage of the project was to ingest the patient EHR JSON files into SAS and start applying a structure to the data. The JSON files are basically flat text files that are split into sections with indicators for the start and end of each section.

SAS has the ability to read JSON files but the engine applies its' own logic and split each patient EHR file into 100+ datasets. This was unexpected and required working out how the datasets were connected to each other before we could even start thinking about the OMOP+ conversion. Based on this we chose to handle the patients one at a time via a macro loop.

The first line of code is used to specify the location of the JSON files and then creates a SAS dataset DIRLIST which contains all the filenames for each JSON file. That dataset is then used to create macro parameters which are used in the final datastep that read in the patient EHR JSON files one by one.

```
filename DIRLIST pipe ' dir "C:\Daleworkfiles\Contractwork\Prototype2\output\fhir\ALZpopbox\*.json" /b ';

**creating list of all filenames present in directory**;
data dirlist ;
   infile dirlist lrecl=200 truncover;
   input file_name $100.;
run;

**placing each filename into a macro variable**;
data _null_;
   set dirlist end=end;
   count+1;
   call symputx('read'||put(count,4.-l),cats("C:\Daleworkfiles\Contractwork\Prototype2\output\fhir\ALZpopbox\",%str(file_name)));
   call symputx('dset'||put(count,4.-l),scan(file_name,1,'.'));
   if end then call symputx('max',count);
run;

*******************************************************************************;
***  pulling SOURCE values from each subject                                  ***;
*******************************************************************************;
%macro json;
  %do i=1 %to &max.;

    filename mydata "&&read&i.";
    libname myjson JSON fileref=mydata ;

    proc datasets lib=myjson noprint; quit;
%end;
%mend json;
%json;
```

Fig 6. SAS code for reading in JSON files

Using the JSON libname engine, SAS creates a large number of SQL tables (100+) for each JSON file that is read in. There is one SQL table that is created called ALLDATA, this contains all the data from the JSON file in one table. The table mirrors the different sections of data as in the source JSON file.

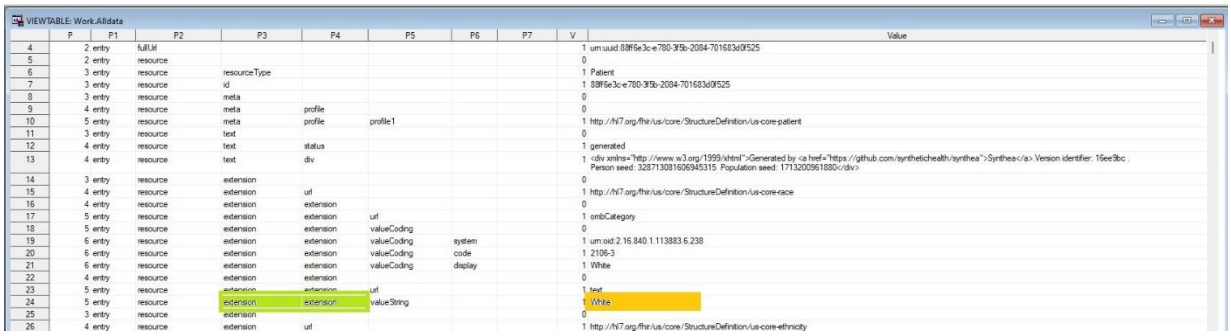

Fig 7. Table ALLDATA example

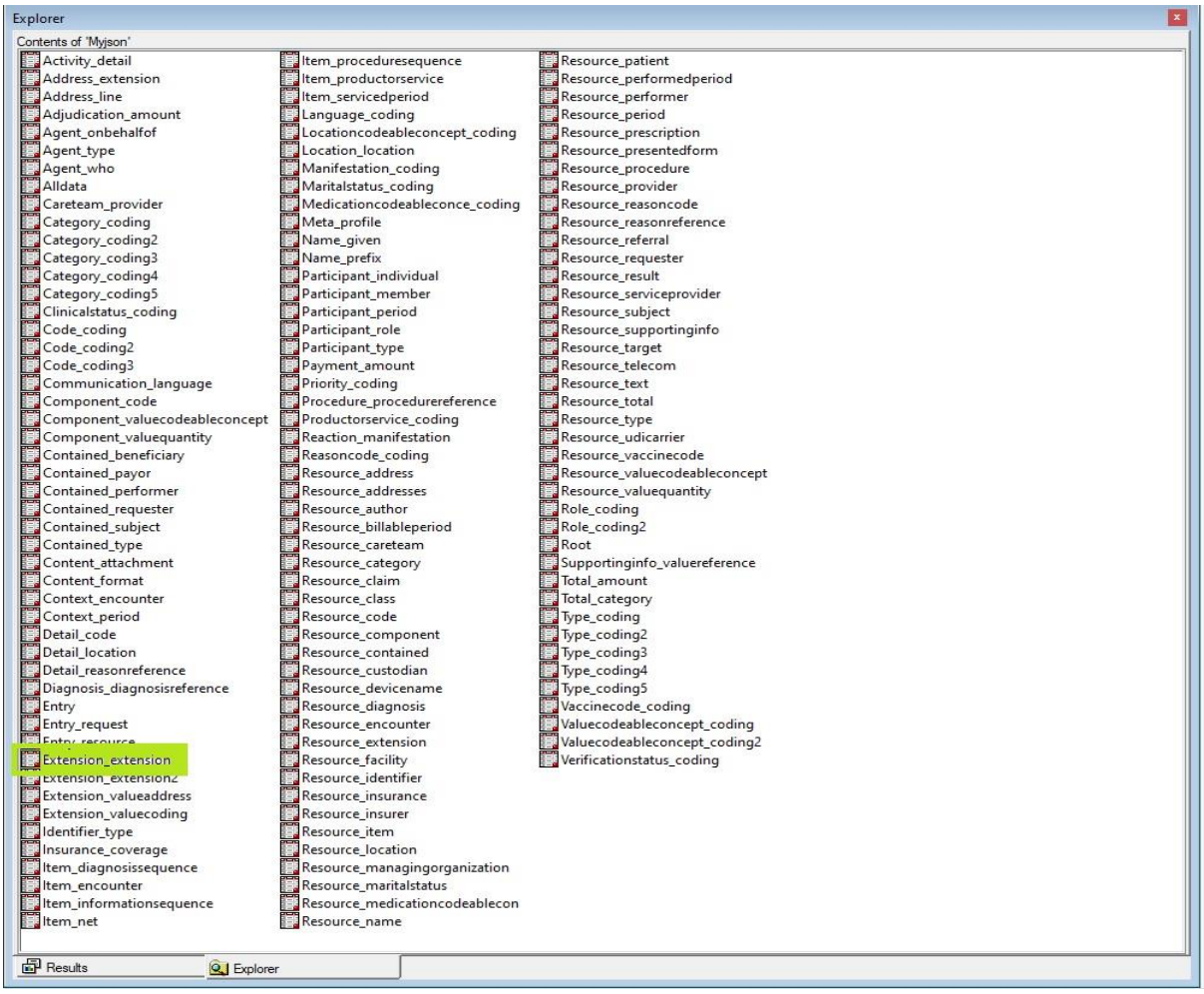

Fig 8. All SQL tables created from a single patient EHR JSON file

After spending a fair amount of time trying to figure out the connections between all the tables, the decision was made to just use the ALLDATA table and extract the data we needed from this table. ALLDATA still needed some extra information before it was fully useful and we created an updated version that merged information from table ENTRY_RESOURCE. The extra information was adding unique identifiers for each section of data to help when extracting the required data values for each OMOP+ domain. A utility macro (ORD_ENTRY.sas) was created to perform this for each patient EHR JSON file after it had been read into SAS.

```
%macro ord_entry / store;

  data __source1;
    set myjson.alldata (where=(v=1 and p1='entry' and p2^='fullUrl'));
    obsno=_n_;
  run;

  data __source2 (rename=(value2=value));
    set __source1 (where=(p3='resourceType'));
    startobs=lag(obsno);
    endobs=obsno-1;
    value2=lag(value);
    if value2='' then delete;
    drop p: value v obsno;
  run;
```

```
  * set up last row *;

  proc sql noprint;
    select nobs into: totalobs
    from sashelp.vtable
    where libname='WORK' and memname='__SOURCE1';
  quit;

  data __source3 (rename=(obsno=startobs));
    set __source1 (where=(p3='resourceType')) end=eof;
    if eof;
    endobs=&totalobs;
    drop p: v;
  run;

  data __source4;
    set __source2
        __source3;
    ordinal_entry=_n_;
  run;

  * total number of ordinal_entries *;

  data _null_;
    set __source4 end=eof;
    if eof then call symputx("ordlast",ordinal_entry);
  run;

  * create macro vars for start observation number and end observation number for each
ordinal *;

  %do y=1 %to &ordlast.;

    data _null_;
      set __source4 end=eof;
      if ordinal_entry=&y. then do;
        call symputx("s_&y",startobs);
        call symputx("e_&y",endobs);
     end;
     if eof then call symputx("ordlast",ordinal_entry);
    run;

  %end;

  * version of __alldata now updated with identifiers for ordinal_entry sections *;

  data __source5;
    set __source1;
    %do z=1 %to &ordlast;
      %if &z=1 %then %do;
        if &&s_&z. <= obsno <= &&e_&z. then ordinal_entry=&z. ;
      %end;
      %else %do;
        else if &&s_&z. <= obsno <= &&e_&z. then ordinal_entry=&z.;
      %end;
    %end;
  run;
```

```
  proc sort data=__source5;
    by ordinal_entry obsno;
  run;

  data alldata2;
    merge myjson.entry_resource(in=a keep=ord: resourcetype id)
          __source5(in=b);
    by ordinal_entry;
    if a;
  run;

%mend ord_entry;
```

OHDSI OMOP common data model v5.4 was used and only a limited number of OMOP+ domains were needed for the project, these included:
Condition_occurrence
Death
Drug_exposure
Fact_relationship
Measurement
Observation
Person_extension
Visit_occurrence

The domains are classed as OMOP+ domains because original values were retained for purposes of traceability from the patient EHR file to the OMOP+ domain. For each OMOP+ domain created, a corresponding specification file was set up to help with the coding of variables and record selection from the patient EHR file. The coding itself was a tricky task, this involved obtaining coding dictionaries from ATHENA, identifying which of those coding dictionaries are needed, extracting the correct records from dataset ALLDATA2 needed for each OMOP+ domain and then the further complications of dependencies between the OMOP+ datasets.

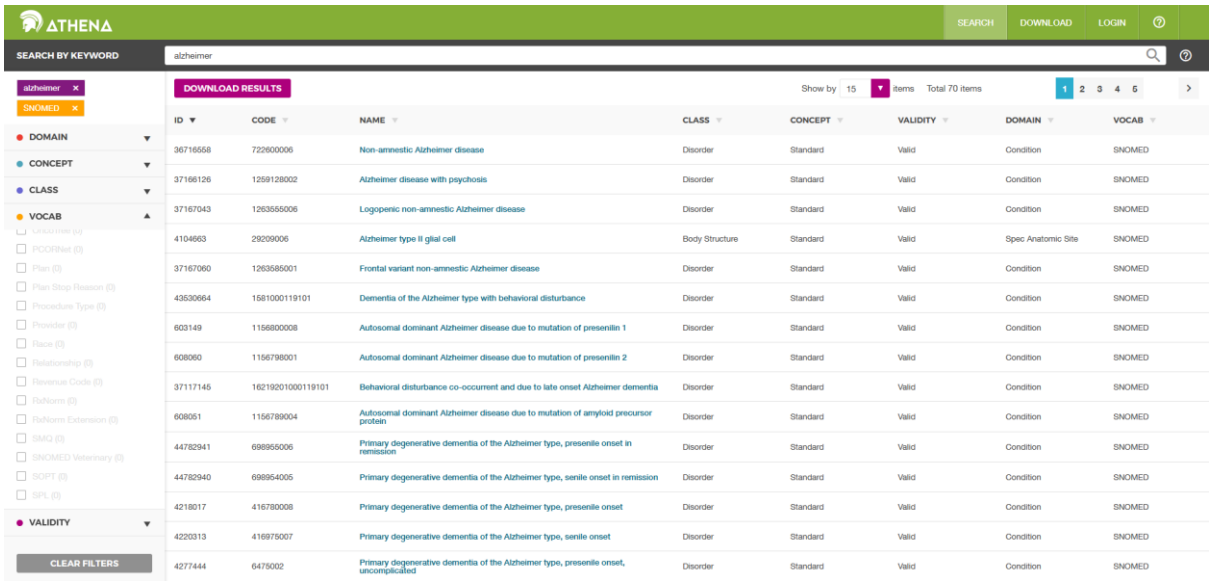

Fig 9. Partial screenshot of a coding dictionary available in ATHENA

The dictionaries were needed to convert HL7 FHIR codes to the OMOP equivalent. Access to ATHENA requires an account setup and takes approx. 3 working days to gain authorization. The main point to note is that the downloadable concept csv file is massive and contains all dictionaries. This file was converted this to a tab delimited text file, then imported into SAS and only kept the dictionaries (LOINC, SNOMED, etc.) needed for this project.

SDTM DOMAINS PRODUCTION

Figure 10 below is a flow chart of the connections between the OMOP+ domains used in the creation of each SDTM domain. The yellow circles represent the SAS programs and where coding dictionaries/mapping files have been used, these are displayed in the bottom section of the chart.

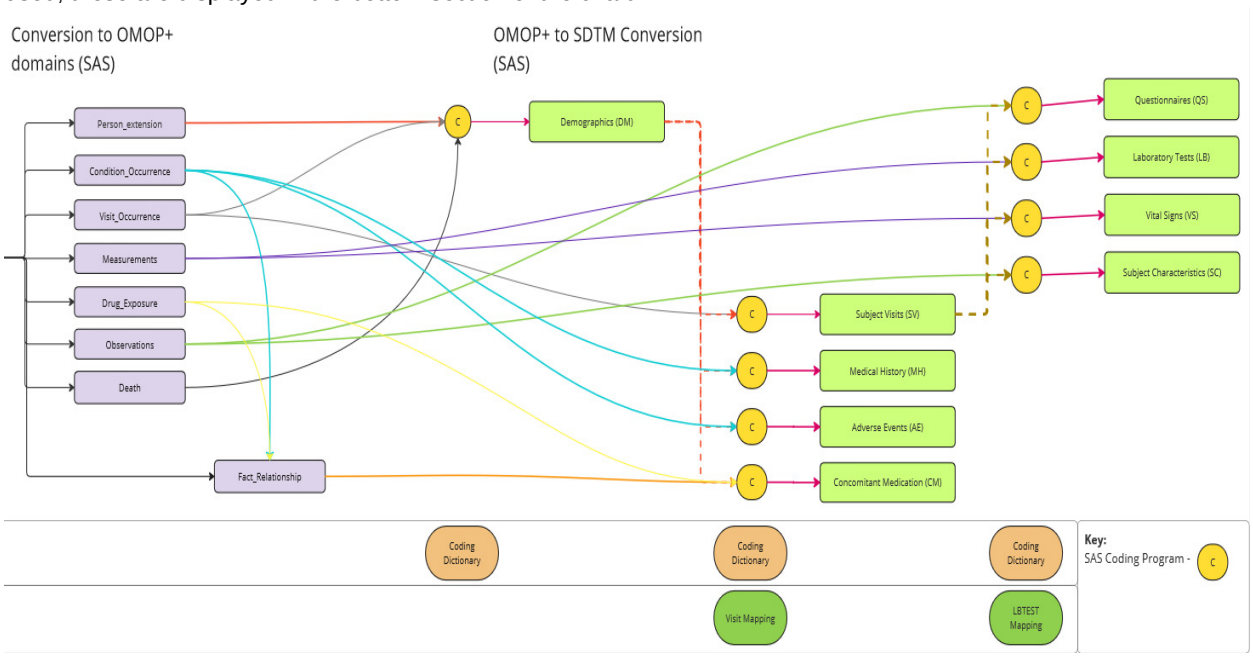

Fig 10. OMOP+ to SDTM conversion

Mapping files were needed to convert original values from the patient EHR JSON files in the OMOP+ domains to SDTM controlled terminology. Figure 11 below is an example of the conversion needed for the SDTM LB domain. This is a time consuming task in setting up the mapping files but ultimately makes the end processing easier.

	1	2	3	4	5
1	measurement_source_value	unit_source_value	LBTEST	LBTESTCD	LBSTRESU
2	Basophils [#/volume] in Blood by Automated count	10*3/uL	Absolute Basophil Count	BASO	10^3/uL
3	Neutrophils [#/volume] in Blood by Automated count	10*3/uL	Absolute Neutrophil Count	NEUT	10^3/uL
4	Activated clotting time (ACT) of Blood by Coagulation assay	s	Activated Coagulation Time Measurement	ACT	s
5	aPTT in Blood by Coagulation assay	s	Activated Partial Thromboplastin Time	APTT	s
6	ALT (Elevated)	[iU]/L	Alanine Aminotransferase Measurement	ALT	IU/L
7	Alanine aminotransferase [Enzymatic activity/volume] in Serum or Plasma	U/L	Alanine Aminotransferase Measurement	ALT	U/L
8	Albumin	g/dL	Albumin Measurement	ALB	g/dL
9	Albumin [Mass/volume] in Serum or Plasma	g/dL	Albumin Measurement	ALB	g/dL
10	Microalbumin Creatinine Ratio	mg/g	Albumin To Creatinine Protein Ratio	ALBCREAT	mg/g
11	Microalbumin/Creatinine [Mass Ratio] in Urine	mg/g	Albumin To Creatinine Protein Ratio	ALBCREAT	mg/g
12	Alkaline Phosphatase	[iU]/L	Alkaline Phosphatase Measurement	ALP	IU/L
13	Alkaline phosphatase [Enzymatic activity/volume] in Serum or Plasma	U/L	Alkaline Phosphatase Measurement	ALP	U/L
14	Anion Gap	mmol/L	Anion Gap Measurement	ANIONG	mmol/L
15	AST (Elevated)	[iU]/L	Aspartate Aminotransferase Measurement	AST	IU/L
16	Aspartate aminotransferase [Enzymatic activity/volume] in Serum or Plasma	U/L	Aspartate Aminotransferase Measurement	AST	U/L
17	Bacteria		Bacterial Count	BACT	
18	Bacteria identified in Urine by Culture		Bacterial Count	BACT	
19	Basophils/100 leukocytes in Blood by Automated count	%	Basophil to Leukocyte Ratio	BASOLE	%
20	Bicarbonate [Moles/volume] in Arterial blood	mmol/L	Bicarbonate Measurement	BICARB	mmol/L
21	Bicarbonate [Moles/volume] in Venous blood	mmol/L	Bicarbonate Measurement	BICARB	mmol/L
22	Calcium	mg/dL	Calcium Measurement	CA	mg/dL
23	Calcium [Mass/volume] in Blood	mg/dL	Calcium Measurement	CA	mg/dL
24	Calcium [Mass/volume] in Serum or Plasma	mg/dL	Calcium Measurement	CA	mg/dL
25	Carbon Dioxide	mmol/L	Carbon Dioxide Measurement	CO2	mmol/L
26	Carbon dioxide [Partial pressure] in Arterial blood	mm[Hg]	Carbon Dioxide Measurement	CO2	mmHg
27	Carbon dioxide [Partial pressure] in Venous blood	mm[Hg]	Carbon Dioxide Measurement	CO2	mmHg

Fig 11. Example mapping file

The visit mapping was performed to emulate the visit schedule in the CDISCPILOT01 study, with the aim to create SDTM domains that were similar to the CDISCPILOT01 study both in terms of content and structure. An example of this is only keeping laboratory tests that were present in the CDISCPILOT01 SDTM LB domain. Overall a lot of data from the source patient EHR JSON files was not selected as it was not needed for the SDTM domains created for this project.

CONCLUSION

The project aim of creating SDTM domains from the synthesized patient EHR JSON files was successfully achieved but the process was an intensive manual, highly complex, multi-stage and time-consuming process. Traceability was achieved and data points could be traced from the raw source file to the SDTM domain.

This paper does not discuss the additional steps of patient attrition and anonymization both of which were performed as part of this project. There are already applications available on Github that convert HL7 FHIR to OMOP but these lacked the traceability aspect which was a necessity for this project, these could be used as a starting point to try and add the traceability aspect but whichever route is chosen, be prepared for a lot of challenges and allow many, many hours for completion.

REFERENCES

This project formed part of a larger proof of concept prototype developed by EQTY Life Sciences and details are in another PhUSE paper **RE04: Ensuring Governance and Traceability of RWD from Source to Regulatory Submission: Use Case** (*ClinLine & EQTY Life Sciences).*

Synthea: https://github.com/synthetichealth/synthea

Synthea Customizer: https://synthetichealth.github.io/spt/

SAS Blog – Reading data with the SAS JSON libname: https://blogs.sas.com/content/sasdummy/2016/12/02/json-libname-engine-sas/

OMOP Common Data Model v5.4: https://ohdsi.github.io/CommonDataModel/cdm54.html#person

ATHENA: https://athena.ohdsi.org/auth/login

CDISCPILOT1 study: https://github.com/RConsortium/submissions-pilot3-adam/tree/main/submission/sdtm

ACKNOWLEDGMENTS

Thanks goes to Berber Snoeijer (Clinline) and Alistair Dootson (EQTY Life Sciences) for their support, guidance and flexibility for this project.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

Anil Sekhari
Sekharico Informatics Limited
Edinburgh, UK
anil.sekhari@sekharico.com