WITH STANDARDS – UNLOCK THE POWER OF DATA

cdisc

2022
US
INTERCHANGE
26-27 OCTOBER | AUSTIN

COSA Dataset-JSON Hackathon Results

Sam Hume, DSc
VP, Data Science CDISC
Session 6, Tack B: Business Optimization & Technical Topics
Oct. 27, 2022

Meet the Speaker

Sam Hume

Title: VP, Data Science

Organization: CDISC

Sam Hume leads the CDISC Data Science team, which collaborates with CDISC staff and stakeholders to develop tools and standards that support clinical and translational data science. Sam directs delivery of the CDISC Library metadata repository that houses all CDISC standards, co-leads the CDISC Data Exchange Standards team, co-leads CORE, and leads the technical CDISC RWD efforts. He has 25 years' experience in clinical research informatics and has held a number of senior technology positions in the biopharmaceutical industry. He holds a doctorate in information systems.

Agenda

1. COSA Dataset-JSON Hackathon Overview
2. Open-source Solutions
3. Commercial Solutions
4. Conclusion

COSA Dataset-JSON Hackathon

An overview of the COSA Dataset-JSON Hackathon

CDISC Open-Source Alliance (COSA)

COSA Mission: The CDISC Open-Source Alliance (COSA) supports, promotes, and sometimes sponsors open-source and free software development projects that create tools for implementing or developing CDISC standards to drive innovation in the CDISC community.

- Virtual hackathon
- Dataset-JSON Hackathon solutions may apply to be included in the COSA Repository Directory
- Requires an open-source license
- Requires a public repository
- Conference session at the US Interchange will highlight solutions
- COSA Webinar to demo solutions

https://cosa.cdisc.org/

Welcome to the COSA Dataset-JSON Hackathon

~ 150 registered participants

New draft data exchange standard for datasets

Create open-source solutions

Focused on data exchange

Convert Dataset-JSON to other common formats

Demonstrate and improve Dataset-JSON

cdisc

The Dataset-JSON Draft Data Exchange Standard

- Dataset-JSON is a draft standard for exchanging tabular datasets using JSON
- It is part of the ODM v2.0 draft standard
    - Planning to start Public Review on Nov. 8th
- It is based on the Dataset-XML v1.0 specification with enhancements, including
    - Much smaller file sizes
    - The addition of essential metadata to support data browsing
- Dataset-JSON links to a Define-XML file for the complete metadata
- Designed to meet the requirements of the regulatory submission use case
    - As well as other data exchange scenarios

Virtual Hackathon: Key Dates

1 Sept

Hackathon Kickoff TC

7 Sept – 19 Oct.

Team Bi-Weekly Check-in:
Wed 10am ET

- Sept. 7th
- Sept. 21st
- Oct. 5th
- Oct. 19th

17 Oct.

Share Project Results

Slides due Oct. 17th

19 Oct.

Project public repo available with code for an initial release

Development may continue

26–27 Oct.

US Interchange

Presentation of Results Oct. 27th

8 Dec.

Webinar Demonstrations

Solutions Created during the Hackathon

- Demonstrate conversion to and from different, language specific dataset formats
- Dataset browsers / viewers
- Methods for handling large datasets
- RESTful Web Services

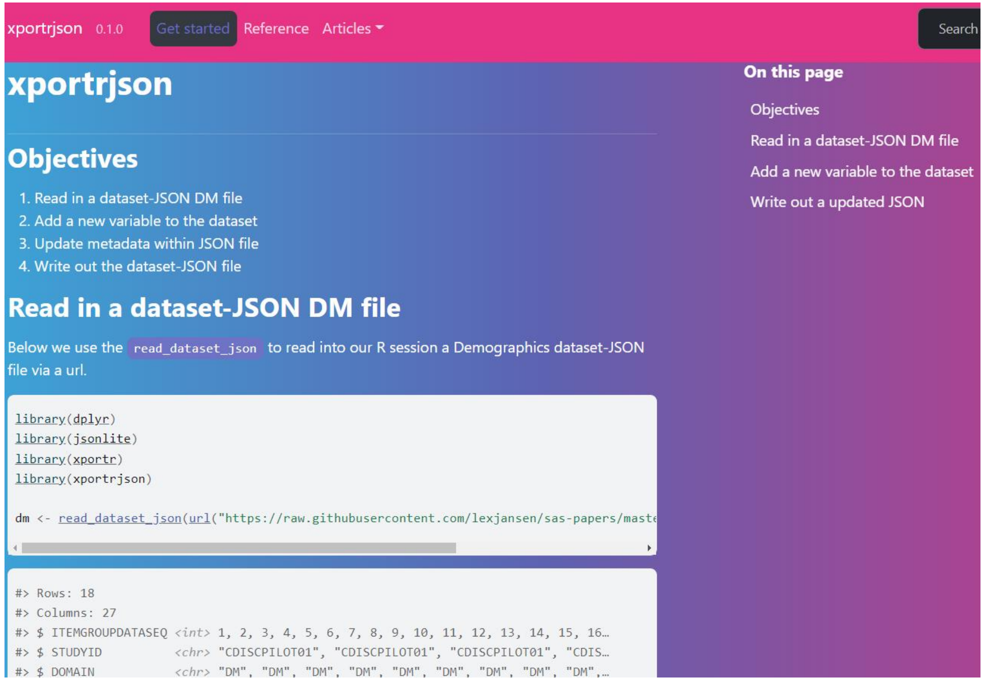
Language	# Solutions
R	5
SAS	4
Python	5
JavaScript	4
Java	1
Swift	1
XSLT	1

Open-Source Solutions

Summary of the Open-Source Solutions Developed During the Hackathon

Dataset-JSON – R package Implementation

- **Authors**: Mike Stackhouse (Atorus), Ben Straub(GSK), Eli Miller(Atorus), Eric Simms(GSK)
- **Repository**: https://github.com/atorus-research/dataset-json-hackathon
- **Website**: Read and Write JSON files specific to Clinical Trail Datasets • xportrjson (atorus-research.github.io)
- **Description**: Atorus and GSK built a simple R package {xportr} that writes out xpt files, we would like to extend this package to read and write out JSON files. This is a POC for that extension.
- **License**: MIT



xportrjson

Objectives

1. Read in a dataset-JSON DM file
2. Add a new variable to the dataset
3. Update metadata within JSON file
4. Write out the dataset-JSON file

Read in a dataset-JSON DM file

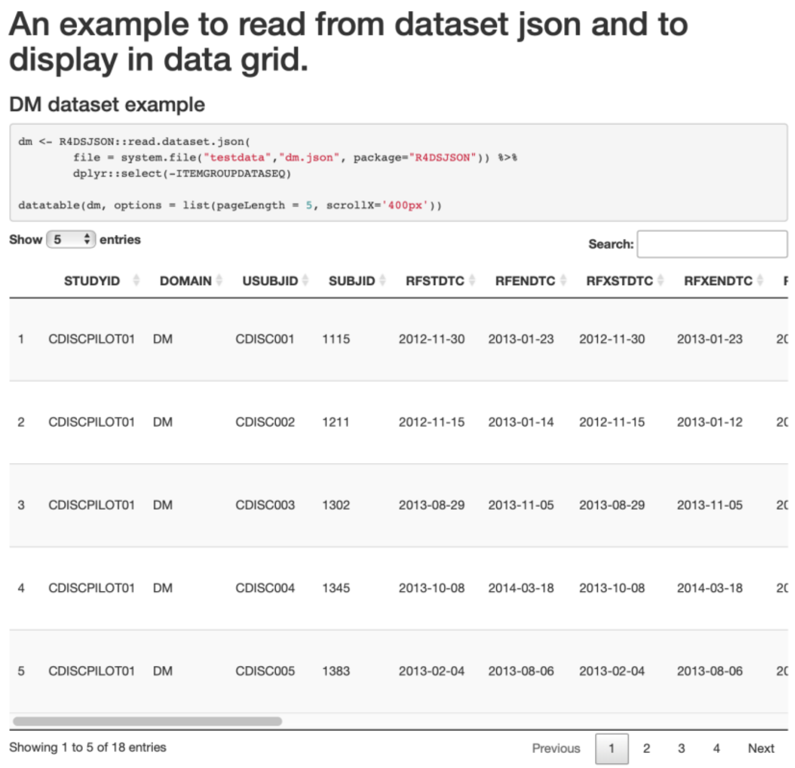
Below we use the `read_dataset_json` to read into our R session a Demographics dataset-JSON file via a url.

```
library(dplyr)
library(jsonlite)
library(xportr)
library(xportrjson)

dm <- read_dataset_json(url("https://raw.githubusercontent.com/lexjansen/sas-papers/maste
```

```
#> Rows: 18
#> Columns: 27
#> $ ITEMGROUPDATASEQ <int> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16…
#> $ STUDYID          <chr> "CDISCPILOT01", "CDISCPILOT01", "CDISCPILOT01", "CDIS…
#> $ DOMAIN           <chr> "DM", "DM", "DM", "DM", "DM", "DM", "DM", "DM", "DM",…
```

R4DSJSON – R Package for Dataset-JSON

- **Author:** Ippei Akiya
- **Repository:** https://github.com/i-akiya/R4DSJSON
- **Description:** R4DSJSON is to read CDISC Dataset-JSON files into R dataframe and to write it from R dataframe.
- **Purpose:** Make it easy to read and write Dataset-JSON in R.
- **License:** MIT



An example to read from dataset json and to display in data grid.

DM dataset example

```
dm <- R4DSJSON::read.dataset.json(
        file = system.file("testdata","dm.json", package="R4DSJSON")) %>%
        dplyr::select(-ITEMGROUPDATASEQ)

datatable(dm, options = list(pageLength = 5, scrollX='400px'))
```

Show 5 entries

Search:

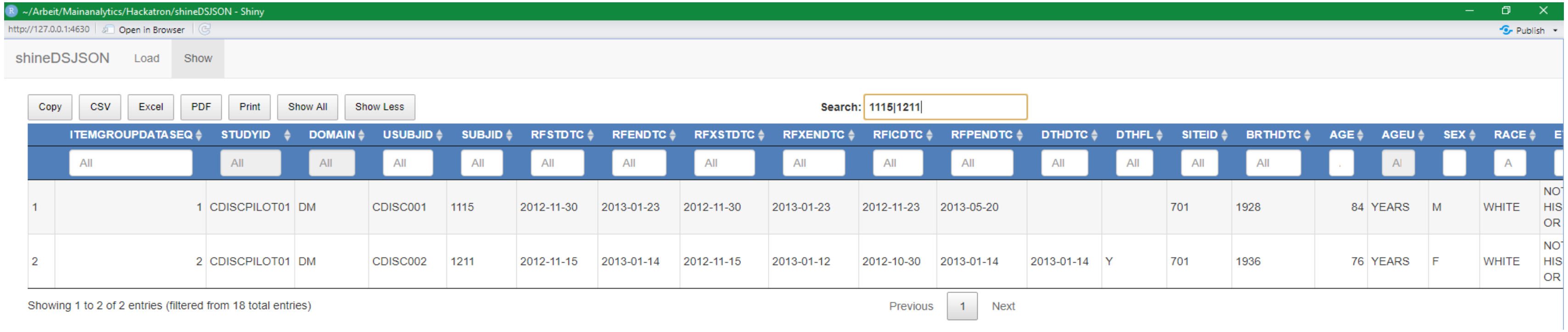
	STUDYID	DOMAIN	USUBJID	SUBJID	RFSTDTC	RFENDTC	RFXSTDTC	RFXENDTC	F
1	CDISCPILOT01	DM	CDISC001	1115	2012-11-30	2013-01-23	2012-11-30	2013-01-23	2(
2	CDISCPILOT01	DM	CDISC002	1211	2012-11-15	2013-01-14	2012-11-15	2013-01-12	2(
3	CDISCPILOT01	DM	CDISC003	1302	2013-08-29	2013-11-05	2013-08-29	2013-11-05	2(
4	CDISCPILOT01	DM	CDISC004	1345	2013-10-08	2014-03-18	2013-10-08	2014-03-18	2(
5	CDISCPILOT01	DM	CDISC005	1383	2013-02-04	2013-08-06	2013-02-04	2013-08-06	2(

Showing 1 to 5 of 18 entries

Previous 1 2 3 4 Next

shineDSJSON

- lightweight **Dataset-JSON viewer** (app / browser)



~/Arbeit/Mainanalytics/Hackatron/shineDSJSON - Shiny

http://127.0.0.1:4630 | Open in Browser | Publish

shineDSJSON Load Show

Copy CSV Excel PDF Print Show All Show Less

Search: 1115|1211

	ITEMGROUPDATASEQ	STUDYID	DOMAIN	USUBJID	SUBJID	RFSTDTC	RFENDTC	RFXSTDTC	RFXENDTC	RFICDTC	RFPENDTC	DTHDTC	DTHFL	SITEID	BRTHDTC	AGE	AGEU	SEX	RACE	E
	All	All	All	All	All	All	All	All	All	All	All	All	All	All	All	.	Al		A	
1	1	CDISCPILOT01	DM	CDISC001	1115	2012-11-30	2013-01-23	2012-11-30	2013-01-23	2012-11-23	2013-05-20			701	1928	84	YEARS	M	WHITE	NO HIS OR
2	2	CDISCPILOT01	DM	CDISC002	1211	2012-11-15	2013-01-14	2012-11-15	2013-01-12	2012-10-30	2013-01-14	2013-01-14	Y	701	1936	76	YEARS	F	WHITE	NO HIS OR

Showing 1 to 2 of 2 entries (filtered from 18 total entries)

Previous 1 Next

- read in JSON files remote and local
- independent column and table wide regex search
- export table/search results as: CSV, Excel, Print

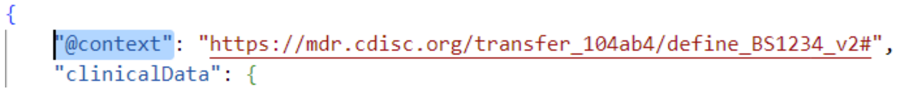
simple usage:

```
install_github("MichelLutz1994/shineDSJSON")
library(shineDSJSON)
shineDSJSON::runViewer()
```

Author: **Michel Lutz**
Feel free to checkout: https://github.com/MichelLutz1994/shineDSJSON

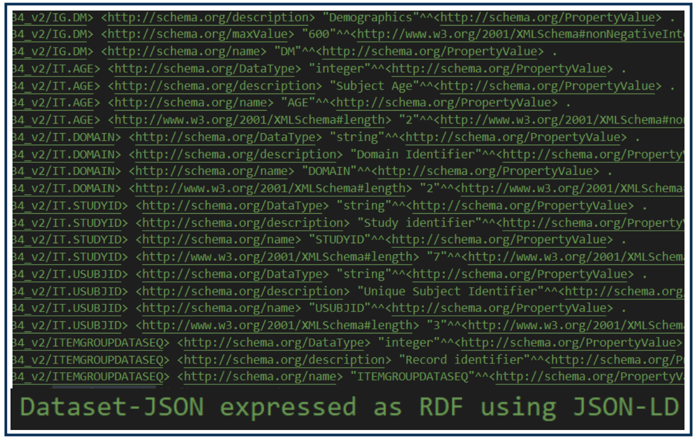
CDISC Over Linked Data (COLD counterpart to FHIR)

1. Apply 1-line **JSON-LD** context URL to Dataset-JSON



```
{
    "@context": "https://mdr.cdisc.org/transfer_104ab4/define_BS1234_v2#",
    "clinicalData": {
```

2. Point it to an explicit **JSON-LD** manifest/Define API
   - Common data contract
     - No need to include Define-XML files
     - Single *a-priori* source of metadata truth for all parties in a 'neutral zone'
     - Acts as standard machine-readable Data Transfer Agreement (DTA)
   - Transforms Dataset-JSON into a graph
     - Applies universally-unique ID to Dataset-JSON content
     - Express Dataset and Define as Linked Open Data on the semantic web
     - Express Dataset and Define as RDF triples / n-quads (to load into a common metamodel e.g. ODM + Biomedical Concepts)



```
34_v2/IG.DM> <http://schema.org/description> "Demographics"^^<http://schema.org/PropertyValue> .
34_v2/IG.DM> <http://schema.org/maxValue> "600"^^<http://www.w3.org/2001/XMLSchema#nonNegativeInt
34_v2/IG.DM> <http://schema.org/name> "DM"^^<http://schema.org/PropertyValue> .
34_v2/IT.AGE> <http://schema.org/DataType> "integer"^^<http://schema.org/PropertyValue> .
34_v2/IT.AGE> <http://schema.org/description> "Subject Age"^^<http://schema.org/PropertyValue> .
34_v2/IT.AGE> <http://schema.org/name> "AGE"^^<http://schema.org/PropertyValue> .
34_v2/IT.AGE> <http://www.w3.org/2001/XMLSchema#length> "2"^^<http://www.w3.org/2001/XMLSchema#no
34_v2/IT.DOMAIN> <http://schema.org/DataType> "string"^^<http://schema.org/PropertyValue> .
34_v2/IT.DOMAIN> <http://schema.org/description> "Domain Identifier"^^<http://schema.org/Property
34_v2/IT.DOMAIN> <http://schema.org/name> "DOMAIN"^^<http://schema.org/PropertyValue> .
34_v2/IT.DOMAIN> <http://www.w3.org/2001/XMLSchema#length> "2"^^<http://www.w3.org/2001/XMLSchema
34_v2/IT.STUDYID> <http://schema.org/DataType> "string"^^<http://schema.org/PropertyValue> .
34_v2/IT.STUDYID> <http://schema.org/description> "Study identifier"^^<http://schema.org/Property
34_v2/IT.STUDYID> <http://schema.org/name> "STUDYID"^^<http://schema.org/PropertyValue> .
34_v2/IT.STUDYID> <http://www.w3.org/2001/XMLSchema#length> "7"^^<http://www.w3.org/2001/XMLSchem
34_v2/IT.USUBJID> <http://schema.org/DataType> "string"^^<http://schema.org/PropertyValue> .
34_v2/IT.USUBJID> <http://schema.org/description> "Unique Subject Identifier"^^<http://schema.org
34_v2/IT.USUBJID> <http://schema.org/name> "USUBJID"^^<http://schema.org/PropertyValue> .
34_v2/IT.USUBJID> <http://www.w3.org/2001/XMLSchema#length> "3"^^<http://www.w3.org/2001/XMLSchem
34_v2/ITEMGROUPDATASEQ> <http://schema.org/DataType> "integer"^^<http://schema.org/PropertyValue>
34_v2/ITEMGROUPDATASEQ> <http://schema.org/description> "Record identifier"^^<http://schema.org/P
34_v2/ITEMGROUPDATASEQ> <http://schema.org/name> "ITEMGROUPDATASEQ"^^<http://schema.org/PropertyV
```

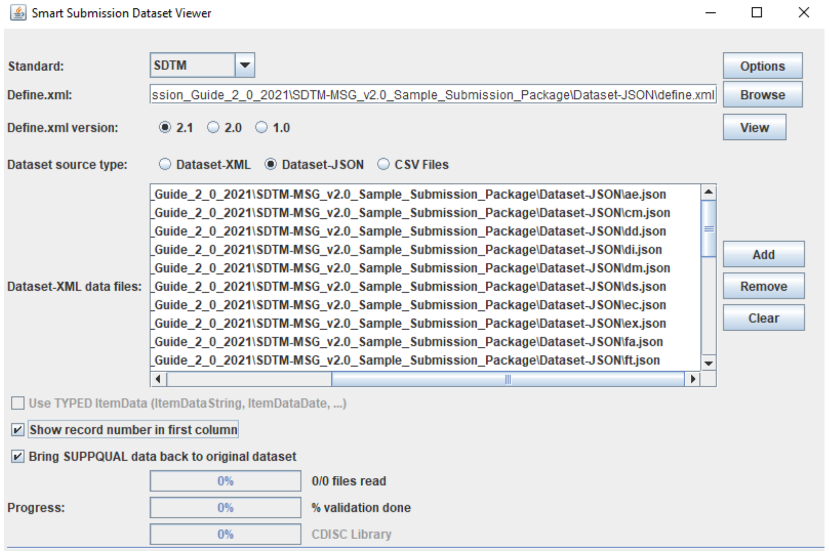
Dataset-JSON expressed as RDF using JSON-LD

3. Profit
   - Guarantee consistency when generating/interpreting datasets
   - Stream / preview large datasets without needing the entire JSON file
   - Explicit, native, human-readable format for expressing CDISC as linked data

Author: Jeremy Teoh **Repo (exploratory)**: https://github.com/TeMeta/Dataset-JSON_hackathon

Smart Submission Dataset Viewer

The Open Source "Smart Submission Dataset Viewer",
a "smart" dedicated viewer for SDTM, SEND and ADaM datasets,
accepts submission datasets in Dataset-JSON, Dataset-XML and CSV format


Smart Submission Dataset Viewer

Standard: SDTM

Define.xml: ssion_Guide_2_0_2021\SDTM-MSG_v2.0_Sample_Submission_Package\Dataset-JSON\define.xml

Define.xml version: [✓ 2.1] 2.0 1.0

Dataset source type: Dataset-XML [✓ Dataset-JSON] CSV Files

Dataset-XML data files:
_Guide_2_0_2021\SDTM-MSG_v2.0_Sample_Submission_Package\Dataset-JSON\ae.json
_Guide_2_0_2021\SDTM-MSG_v2.0_Sample_Submission_Package\Dataset-JSON\cm.json
_Guide_2_0_2021\SDTM-MSG_v2.0_Sample_Submission_Package\Dataset-JSON\dd.json
_Guide_2_0_2021\SDTM-MSG_v2.0_Sample_Submission_Package\Dataset-JSON\di.json
_Guide_2_0_2021\SDTM-MSG_v2.0_Sample_Submission_Package\Dataset-JSON\dm.json
_Guide_2_0_2021\SDTM-MSG_v2.0_Sample_Submission_Package\Dataset-JSON\ds.json
_Guide_2_0_2021\SDTM-MSG_v2.0_Sample_Submission_Package\Dataset-JSON\ec.json
_Guide_2_0_2021\SDTM-MSG_v2.0_Sample_Submission_Package\Dataset-JSON\ex.json
_Guide_2_0_2021\SDTM-MSG_v2.0_Sample_Submission_Package\Dataset-JSON\fa.json
_Guide_2_0_2021\SDTM-MSG_v2.0_Sample_Submission_Package\Dataset-JSON\ft.json

☐ Use TYPED ItemData (ItemDataString, ItemDataDate, ...)
☑ Show record number in first column
☑ Bring SUPPQUAL data back to original dataset

Progress:
0% 0/0 files read
0% % validation done
0% CDISC Library

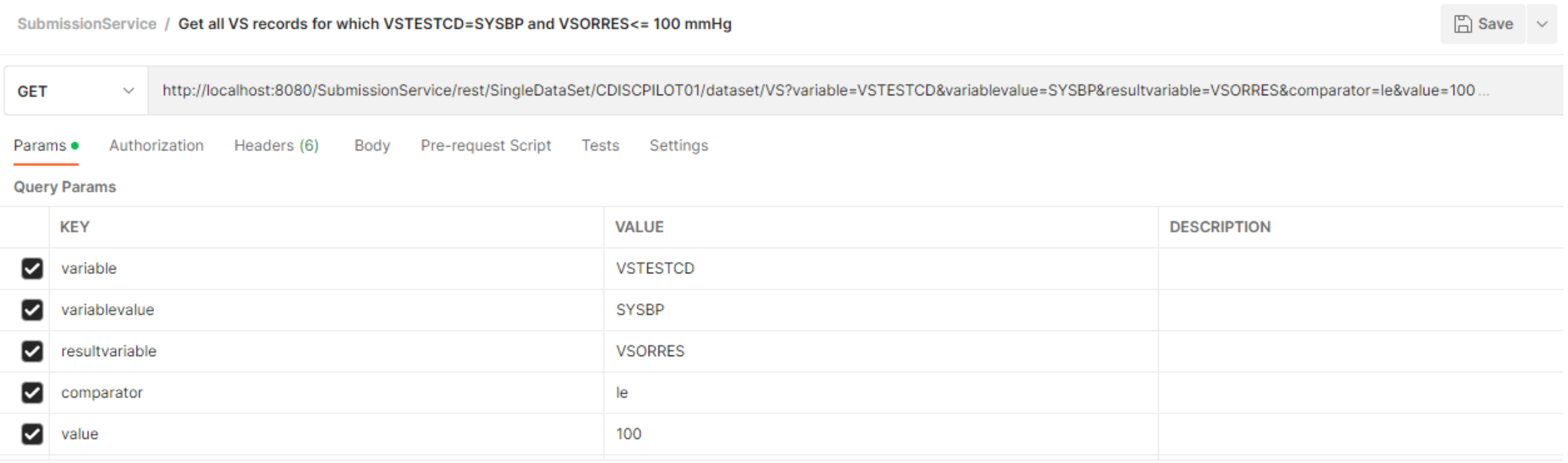
Included:
XPT datasets
can be
transformed to
Dataset-JSON
or Dataset-XML

Author: Jozef Aerts

https://sourceforge.net/projects/smart-submission-dataset-viewe/

RESTful Web Service using Dataset-JSON

- A simple prototype RESTful Web Service for querying submissions from a repository, using Dataset-JSON for the response, has been implemented
- Try it out at: http://xml4pharmaserver.com/WebServices/Submission_Services_Dataset-JSON/


SubmissionService / Get all VS records for which VSTESTCD=SYSBP and VSORRES<= 100 mmHg

GET http://localhost:8080/SubmissionService/rest/SingleDataSet/CDISCPILOT01/dataset/VS?variable=VSTESTCD&variablevalue=SYSBP&resultvariable=VSORRES&comparator=le&value=100 ...

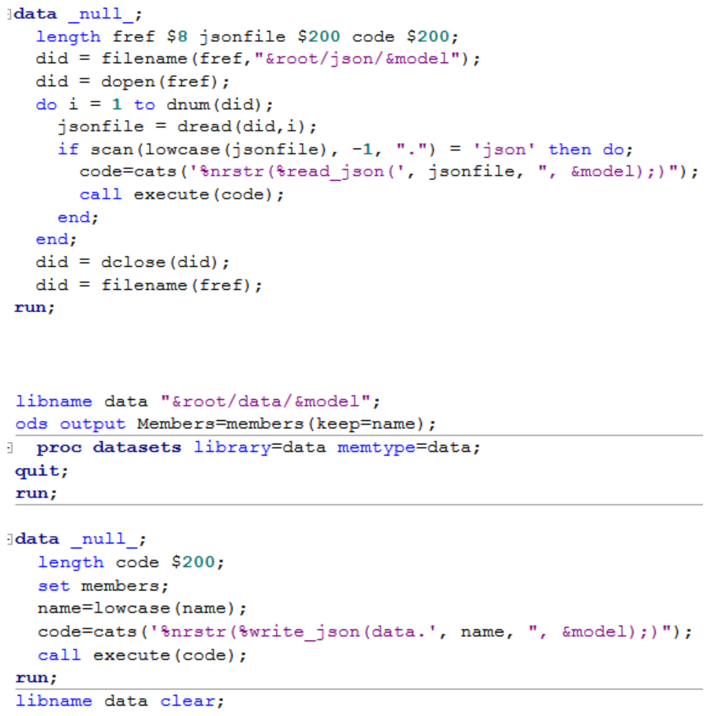
Query Params

	KEY	VALUE	DESCRIPTION
✓	variable	VSTESTCD	
✓	variablevalue	SYSBP	
✓	resultvariable	VSORRES	
✓	comparator	le	
✓	value	100	

Author: Jozef Aerts

Dataset-JSON - SAS Implementation

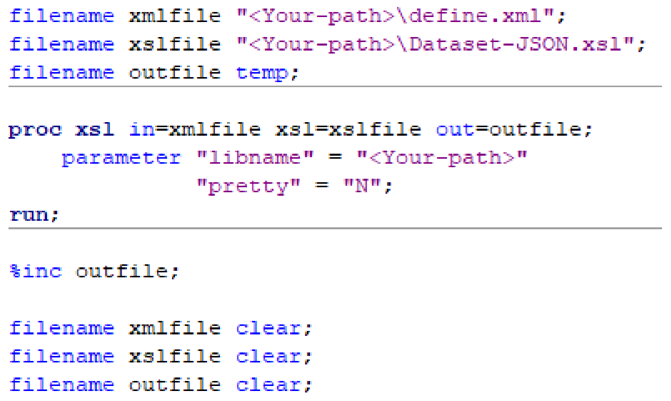
- **Author:** Lex Jansen
- **Repository:** https://github.com/lexjansen/dataset-json-sas
- **Description:** SAS macros and example programs used to create SAS datasets from Dataset-JSON as well as creating Dataset-JSON from SAS datasets.
  Metadata from Define-XML is read with Lua and used to for validation purposes and as pre-specified metadata.
- **Purpose:** Demonstrate Dataset-JSON's utility as a data exchange format.
- **License:** MIT



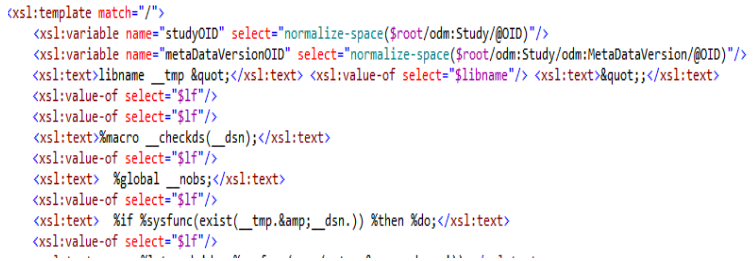
```
data _null_;
  length fref $8 jsonfile $200 code $200;
  did = filename(fref,"&root/json/&model");
  did = dopen(fref);
  do i = 1 to dnum(did);
    jsonfile = dread(did,i);
    if scan(lowcase(jsonfile), -1, ".") = 'json' then do;
      code=cats('%nrstr(%read_json(', jsonfile, ", &model);)");
      call execute(code);
    end;
  end;
  did = dclose(did);
  did = filename(fref);
run;


libname data "&root/data/&model";
ods output Members=members(keep=name);
  proc datasets library=data memtype=data;
quit;
run;

data _null_;
  length code $200;
  set members;
  name=lowcase(name);
  code=cats('%nrstr(%write_json(data.', name, ", &model);)");
  call execute(code);
run;
libname data clear;
```

SAS PROC XSL–Dataset-JSON from Define.xml

- **Author:** Pierre Dostie
- **Repository:** https://github.com/dostiep/Dataset-JSON
- **Description:** Create Dataset-JSON files using SAS Procedure XSL. SAS code is generated from a Define.xml using a XSL stylesheet (Dataset-JSON.xsl).
- **Purpose:** Demonstrate Dataset-JSON's utility as a data exchange format.
- **License:** MIT



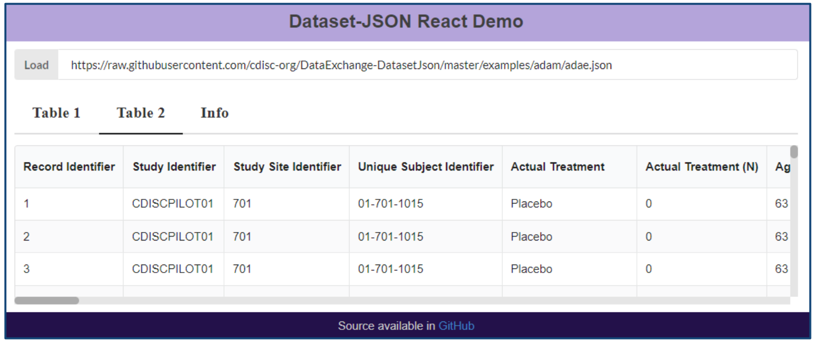
```
filename xmlfile "<Your-path>\define.xml";
filename xslfile "<Your-path>\Dataset-JSON.xsl";
filename outfile temp;

proc xsl in=xmlfile xsl=xslfile out=outfile;
    parameter "libname" = "<Your-path>"
              "pretty" = "N";
run;

%inc outfile;

filename xmlfile clear;
filename xslfile clear;
filename outfile clear;
```



```
<xsl:template match="/">
   <xsl:variable name="studyOID" select="normalize-space($root/odm:Study/@OID)"/>
   <xsl:variable name="metaDataVersionOID" select="normalize-space($root/odm:Study/odm:MetaDataVersion/@OID)"/>
   <xsl:text>libname __tmp &quot;</xsl:text> <xsl:value-of select="$libname"/> <xsl:text>&quot;;</xsl:text>
   <xsl:value-of select="$lf"/>
   <xsl:value-of select="$lf"/>
   <xsl:text>%macro __checkds(__dsn);</xsl:text>
   <xsl:value-of select="$lf"/>
   <xsl:text>  %global __nobs;</xsl:text>
   <xsl:value-of select="$lf"/>
   <xsl:text>  %if %sysfunc(exist(__tmp.&amp;__dsn.)) %then %do;</xsl:text>
   <xsl:value-of select="$lf"/>
```

Dataset-JSON React Demo

- **Author:** Katja Glaß
- **Repository:** https://github.com/KatjaGlassConsulting/dataset-json-react-demo
- **Description:** This is an example web application created by using the React framework to display the dataset content which is read in from an URL
- **Purpose:** demonstrate React usage and different table display options
- **License:** MIT



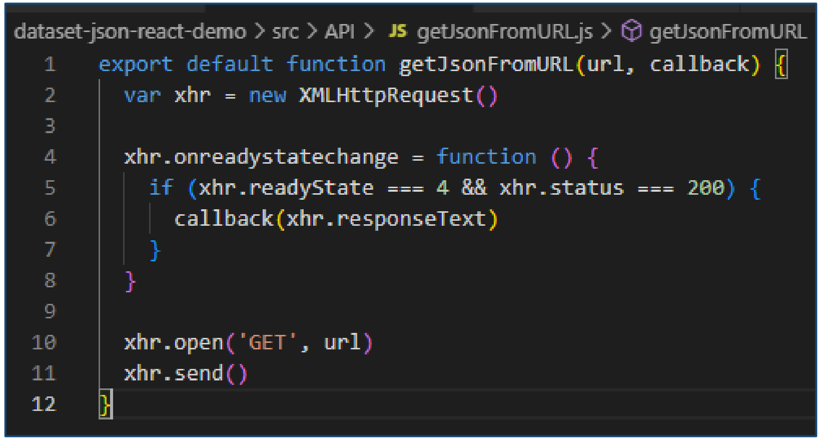
Dataset-JSON React Demo

Load https://raw.githubusercontent.com/cdisc-org/DataExchange-DatasetJson/master/examples/adam/adae.json

Table 1 Table 2 Info

Record Identifier	Study Identifier	Study Site Identifier	Unique Subject Identifier	Actual Treatment	Actual Treatment (N)	Ag
1	CDISCPILOT01	701	01-701-1015	Placebo	0	63
2	CDISCPILOT01	701	01-701-1015	Placebo	0	63
3	CDISCPILOT01	701	01-701-1015	Placebo	0	63

Source available in GitHub



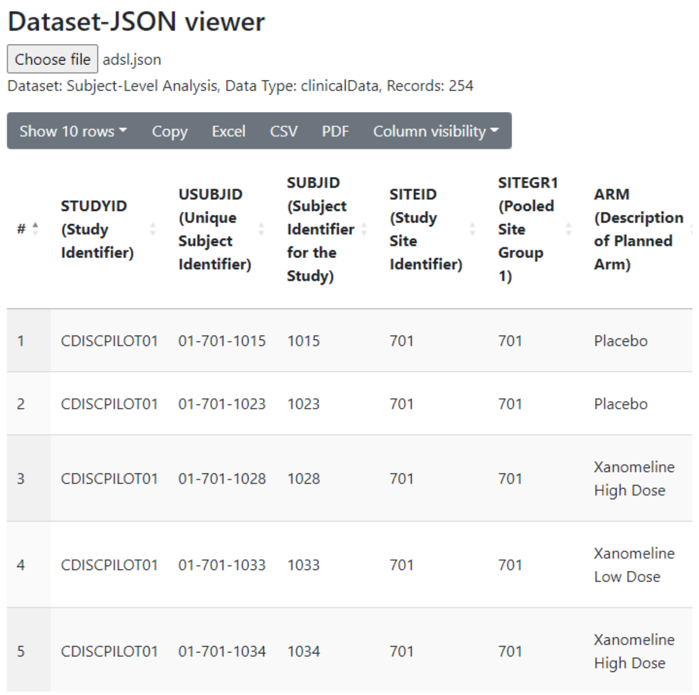
dataset-json-react-demo > src > API > getJsonFromURL.js > getJsonFromURL

```
export default function getJsonFromURL(url, callback) {
  var xhr = new XMLHttpRequest()

  xhr.onreadystatechange = function () {
    if (xhr.readyState === 4 && xhr.status === 200) {
      callback(xhr.responseText)
    }
  }

  xhr.open('GET', url)
  xhr.send()
}
```

Dataset-JSON-viewer

Statskom CLINICAL PROGRAMMING

- **Author: Andrew Ndikom**
- **Repository:** GitHUB Repo
- **Description:** Built on the Datatables JS library the tool renders Dataset-JSON files in a tabular format and allows users to:
  - Filter rows,
  - Control pagination,
  - Show/ hide columns,
  - Export data to a number of common file formats,
  - Copy data.
- **Purpose:** Offer users an intuitive, modern, browser based, Excel™ like experience for viewing and interacting with Dataset-JSON files.
- **Licence: MIT**



Dataset-JSON viewer

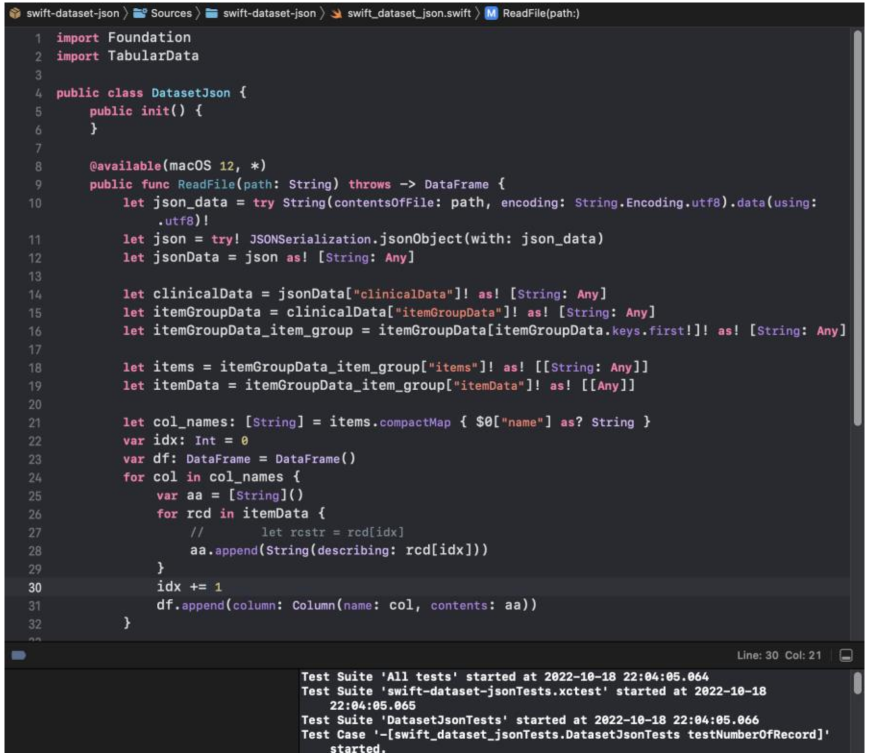
Choose file adsl.json

Dataset: Subject-Level Analysis, Data Type: clinicalData, Records: 254

#	STUDYID (Study Identifier)	USUBJID (Unique Subject Identifier)	SUBJID (Subject Identifier for the Study)	SITEID (Study Site Identifier)	SITEGR1 (Pooled Site Group 1)	ARM (Description of Planned Arm)
1	CDISCPILOT01	01-701-1015	1015	701	701	Placebo
2	CDISCPILOT01	01-701-1023	1023	701	701	Placebo
3	CDISCPILOT01	01-701-1028	1028	701	701	Xanomeline High Dose
4	CDISCPILOT01	01-701-1033	1033	701	701	Xanomeline Low Dose
5	CDISCPILOT01	01-701-1034	1034	701	701	Xanomeline High Dose

swift-dataset-json: Swift Package for Dataset-JSON

- **Author:** Ippei Akiya
- **Repository:** https://github.com/i-akiya/swift-dataset-json
- **Description:** CDISC Dataset-JSON file reader in swift that is useful to develop a data review application on iPhone and iPad.
- **Purpose:** Make it easy to read Dataset-JSON in Swift.
- **License:** MIT

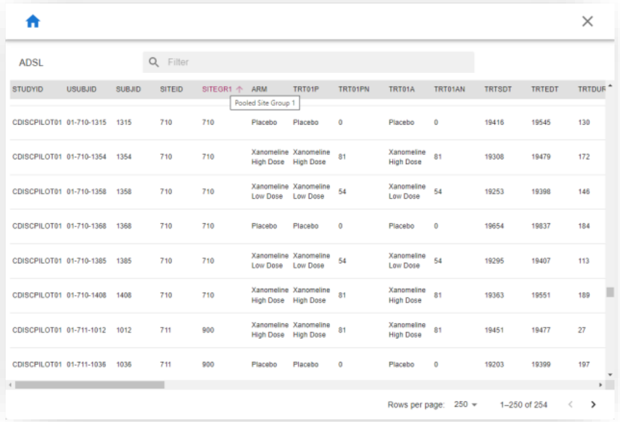

swift-dataset-json 〉 Sources 〉 swift-dataset-json 〉 swift_dataset_json.swift 〉 M ReadFile(path:)
Line: 30 Col: 21

```swift
import Foundation
import TabularData

public class DatasetJson {
    public init() {
    }

    @available(macOS 12, *)
    public func ReadFile(path: String) throws -> DataFrame {
        let json_data = try String(contentsOfFile: path, encoding: String.Encoding.utf8).data(using:
            .utf8)!
        let json = try! JSONSerialization.jsonObject(with: json_data)
        let jsonData = json as! [String: Any]

        let clinicalData = jsonData["clinicalData"]! as! [String: Any]
        let itemGroupData = clinicalData["itemGroupData"]! as! [String: Any]
        let itemGroupData_item_group = itemGroupData[itemGroupData.keys.first!]! as! [String: Any]

        let items = itemGroupData_item_group["items"]! as! [[String: Any]]
        let itemData = itemGroupData_item_group["itemData"]! as! [[Any]]

        let col_names: [String] = items.compactMap { $0["name"] as? String }
        var idx: Int = 0
        var df: DataFrame = DataFrame()
        for col in col_names {
            var aa = [String]()
            for rcd in itemData {
                //       let rcstr = rcd[idx]
                aa.append(String(describing: rcd[idx]))
            }
            idx += 1
            df.append(column: Column(name: col, contents: aa))
        }
```

```
Test Suite 'All tests' started at 2022-10-18 22:04:05.064
Test Suite 'swift-dataset-jsonTests.xctest' started at 2022-10-18
    22:04:05.065
Test Suite 'DatasetJsonTests' started at 2022-10-18 22:04:05.066
Test Case '-[swift_dataset_jsonTests.DatasetJsonTests testNumberOfRecord]'
    started.
```

stream/serve/view-dataset-json

- **Authors:** Parexel (Juan Abdon, Ivan Osipov, Mauro Bringas, Dmitry Kolosov)
- **Repository:** stream/serve/view-dataset-json
- **Description:** This solution includes 3 subprojects
  - stream-dataset-json - Python library to read Dataset-JSON files as a stream
  - serve-dataset-json - Python library to serve Dataset-JSON files via API
  - view-dataset-json - TypeScript project implementing a viewer for Dataset-JSON files



ADSL

Filter

Pooled Site Group 1

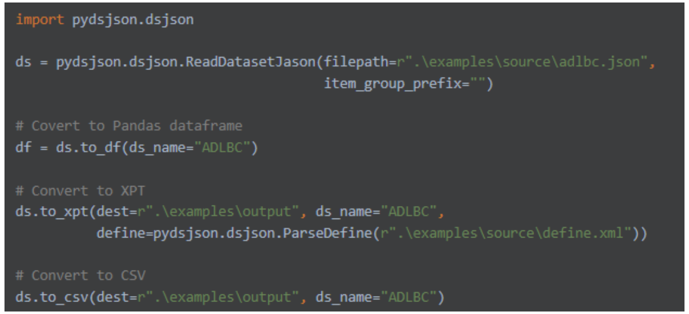
STUDYID	USUBJID	SUBJID	SITEID	SITEGR1 ↑	ARM	TRT01P	TRT01PN	TRT01A	TRT01AN	TRTSDT	TRTEDT	TRTDUR
CDISCPILOT01	01-710-1315	1315	710	710	Placebo	Placebo	0	Placebo	0	19416	19545	130
CDISCPILOT01	01-710-1354	1354	710	710	Xanomeline High Dose	Xanomeline High Dose	81	Xanomeline High Dose	81	19308	19479	172
CDISCPILOT01	01-710-1358	1358	710	710	Xanomeline Low Dose	Xanomeline Low Dose	54	Xanomeline Low Dose	54	19253	19398	146
CDISCPILOT01	01-710-1368	1368	710	710	Placebo	Placebo	0	Placebo	0	19654	19837	184
CDISCPILOT01	01-710-1385	1385	710	710	Xanomeline Low Dose	Xanomeline Low Dose	54	Xanomeline Low Dose	54	19295	19407	113
CDISCPILOT01	01-710-1408	1408	710	710	Xanomeline High Dose	Xanomeline High Dose	81	Xanomeline High Dose	81	19363	19551	189
CDISCPILOT01	01-711-1012	1012	711	900	Xanomeline High Dose	Xanomeline High Dose	81	Xanomeline High Dose	81	19451	19477	27
CDISCPILOT01	01-711-1036	1036	711	900	Placebo	Placebo	0	Placebo	0	19203	19399	197

Rows per page: 250 ▾ 1–250 of 254

- **Purpose:** The goal of the project is to write a library which allows to efficiently read Dataset-JSON files (including huge file sizes) and show to how it can be utilized for different purposes.
- **License: MIT**

Dataset-JSON - Python Implementation

- **Author:** Satish Ghadigaonkar
- **Repository:** https://github.com/satish-ghadigaonkar/pydsjson
- **Description:**
  - Python module to convert Dataset-JSON to Pandas dataframe, XPT and CSV as well as to convert XPT to Dataset-JSON.
  - Command-line interface is also available.
- **Purpose:** Demonstrate Dataset-JSON's utility as a data exchange format.
- **License:** MIT



```python
import pydsjson.dsjson

ds = pydsjson.dsjson.ReadDatasetJason(filepath=r".\examples\source\adlbc.json",
                                      item_group_prefix="")

# Covert to Pandas dataframe
df = ds.to_df(ds_name="ADLBC")

# Convert to XPT
ds.to_xpt(dest=r".\examples\output", ds_name="ADLBC",
          define=pydsjson.dsjson.ParseDefine(r".\examples\source\define.xml"))

# Convert to CSV
ds.to_csv(dest=r".\examples\output", ds_name="ADLBC")
```

django-sdtm-export (repository link)

Problem: When collecting clinical data in a Django app, implementing exports for each domain that adhere to SDTM standards can be laborious and repetitive.

Solution: django-sdtm-export: django-sdtm-export is an open source package that can be easily added to any django application. It allows for simple declarative export specifications, meaning new domains can be exported with as little as 75 lines of code. It supports exporting to both CSV and Dataset-JSON.

Contributors:

- Lindus Health: Madeleine De Forest-Brown, Zaid Al-Jubouri, Amiel Kollek
- Jeremy Teoh

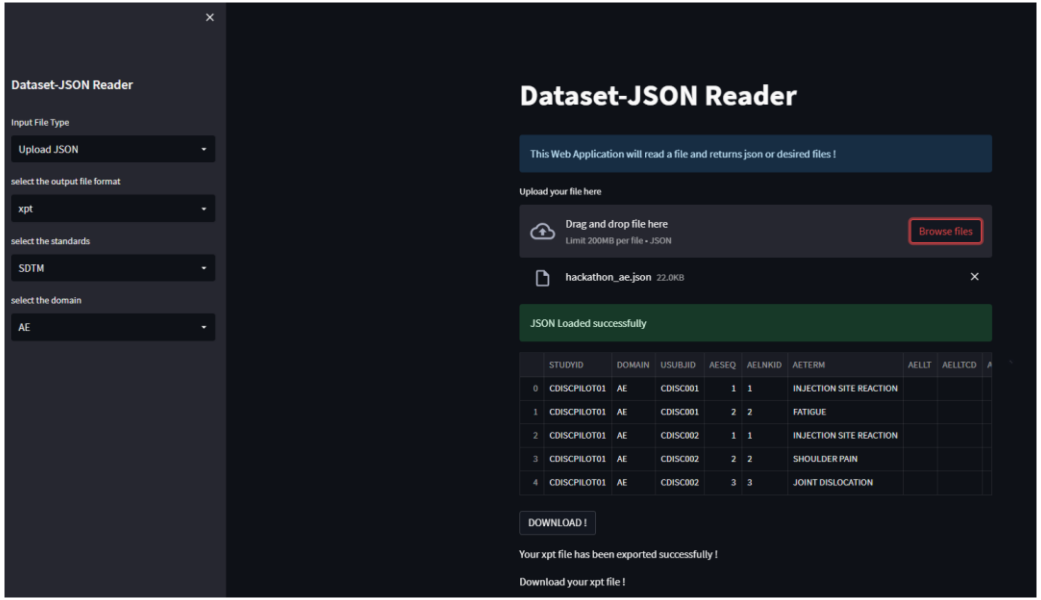
Technologies: Python, Django, Pandas

License: MIT

Dataset-JSON Hackathon

- A open-source tool that search throughout the Dataset-JSON with user selections and convert it into various formats which make the conversion easier and more effective.

- **Description** : Created an open-source tool to work with Dataset-JSON, which takes JSON as input and allows the user to convert the desired output file format and vice versa. This tool is established to work with formats such as: SAS v5 XPORT, R data frame, xml, CSV.



Dataset-JSON Reader

Input File Type: Upload JSON

select the output file format: xpt

select the standards: SDTM

select the domain: AE

Dataset-JSON Reader

This Web Application will read a file and returns json or desired files !

Upload your file here

Drag and drop file here
Limit 200MB per file • JSON

hackathon_ae.json 22.0KB

JSON Loaded successfully

	STUDYID	DOMAIN	USUBJID	AESEQ	AELNKID	AETERM	AELLT	AELLTCD
0	CDISCPILOT01	AE	CDISC001	1	1	INJECTION SITE REACTION		
1	CDISCPILOT01	AE	CDISC001	2	2	FATIGUE		
2	CDISCPILOT01	AE	CDISC002	1	1	INJECTION SITE REACTION		
3	CDISCPILOT01	AE	CDISC002	2	2	SHOULDER PAIN		
4	CDISCPILOT01	AE	CDISC002	3	3	JOINT DISLOCATION		

Your xpt file has been exported successfully !

Download your xpt file !

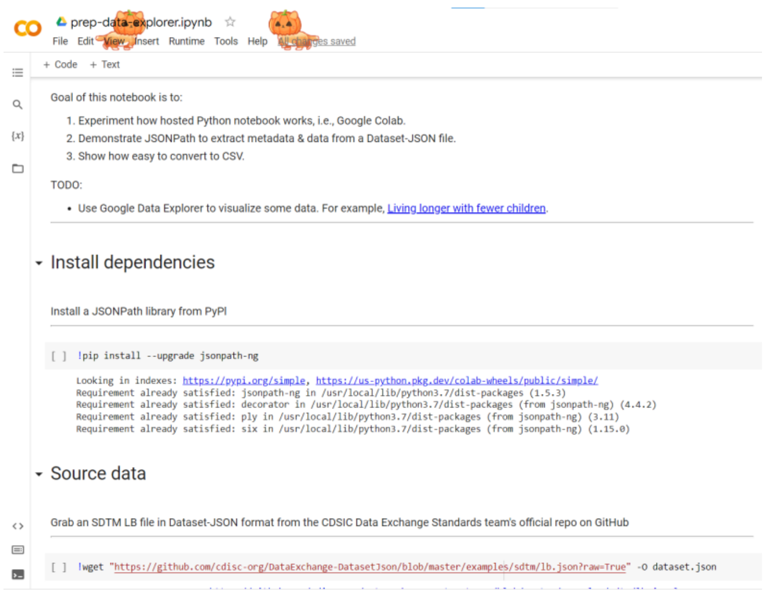
- **Authors** : Renswick.D and Deepika.S from Pfizer (SPA) India

- **Software** : Python

- **Solution link**: renswick-pfizer/Dataset-JSON-Hackathon (github.com)

Hosted Python Notebook, JSONPath, CSV

- **Author**: Anthony Chow
- **Repo**: https://colab.research.google.com/drive/1myHpenokUEb4DXeghxbGTJr2rajwP0I8
- **Description**: A small data wrangling exercise using Google Colab, a hosted Python notebook.
- **Purpose**: Demonstration of downloading a Dataset-JSON file and verifying metadata using jasonpath-ng library. Investigate using open data visualization tool such as Google Public Data Explorer.
- **License**: MIT



Goal of this notebook is to:

1. Experiment how hosted Python notebook works, i.e., Google Colab.
2. Demonstrate JSONPath to extract metadata & data from a Dataset-JSON file.
3. Show how easy to convert to CSV.

TODO:

- Use Google Data Explorer to visualize some data. For example, Living longer with fewer children.

Install dependencies

Install a JSONPath library from PyPI

```
[ ] !pip install --upgrade jsonpath-ng

    Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
    Requirement already satisfied: jsonpath-ng in /usr/local/lib/python3.7/dist-packages (1.5.3)
    Requirement already satisfied: decorator in /usr/local/lib/python3.7/dist-packages (from jsonpath-ng) (4.4.2)
    Requirement already satisfied: ply in /usr/local/lib/python3.7/dist-packages (from jsonpath-ng) (3.11)
    Requirement already satisfied: six in /usr/local/lib/python3.7/dist-packages (from jsonpath-ng) (1.15.0)
```

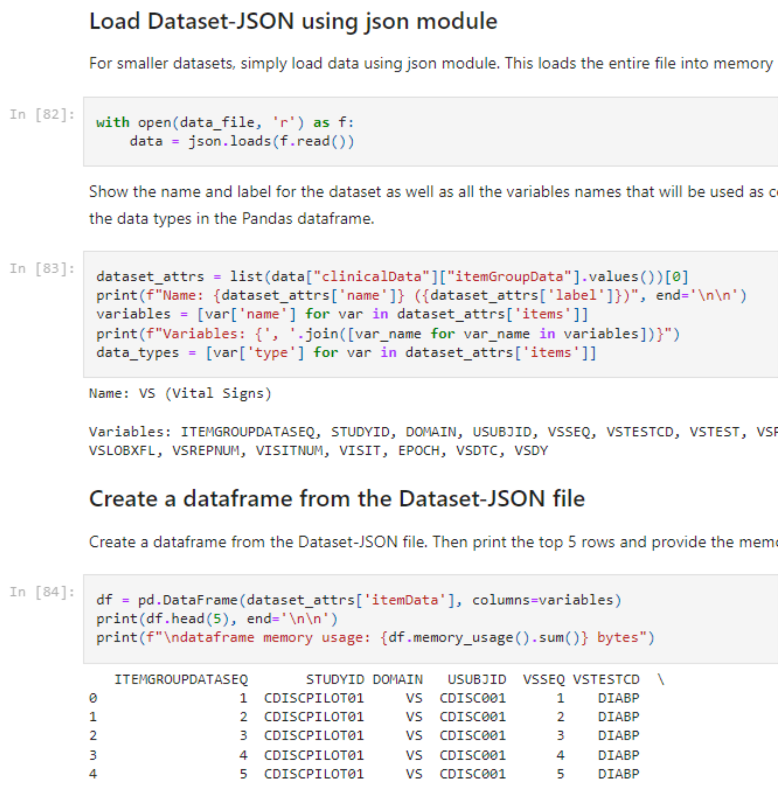
Source data

Grab an SDTM LB file in Dataset-JSON format from the CDSIC Data Exchange Standards team's official repo on GitHub

```
[ ] !wget "https://github.com/cdisc-org/DataExchange-DatasetJson/blob/master/examples/sdtm/lb.json?raw=True" -O dataset.json
```

Jupyter Notebook – Experimenting with Dataset-JSON

- **Author:** Sam Hume
- **Repository:** https://github.com/swhume/dataset-json-hackathon
- **Description:** Jupyter Notebook used to explore generating Python dataframes using Dataset-JSON as well as creating Dataset-JSON from CSV files. Also explores techniques for processing large datasets.
- **Purpose:** Demonstrate Dataset-JSON's utility as a data exchange format.
- **License:** MIT



Load Dataset-JSON using json module

For smaller datasets, simply load data using json module. This loads the entire file into memory

```
In [82]:
with open(data_file, 'r') as f:
    data = json.loads(f.read())
```

Show the name and label for the dataset as well as all the variables names that will be used as c
the data types in the Pandas dataframe.

```
In [83]:
dataset_attrs = list(data["clinicalData"]["itemGroupData"].values())[0]
print(f"Name: {dataset_attrs['name']} ({dataset_attrs['label']})", end='\n\n')
variables = [var['name'] for var in dataset_attrs['items']]
print(f"Variables: {', '.join([var_name for var_name in variables])}")
data_types = [var['type'] for var in dataset_attrs['items']]
```

```
Name: VS (Vital Signs)

Variables: ITEMGROUPDATASEQ, STUDYID, DOMAIN, USUBJID, VSSEQ, VSTESTCD, VSTEST, VSF
VSLOBXFL, VSREPNUM, VISITNUM, VISIT, EPOCH, VSDTC, VSDY
```

Create a dataframe from the Dataset-JSON file

Create a dataframe from the Dataset-JSON file. Then print the top 5 rows and provide the mem

```
In [84]:
df = pd.DataFrame(dataset_attrs['itemData'], columns=variables)
print(df.head(5), end='\n\n')
print(f"\ndataframe memory usage: {df.memory_usage().sum()} bytes")
```

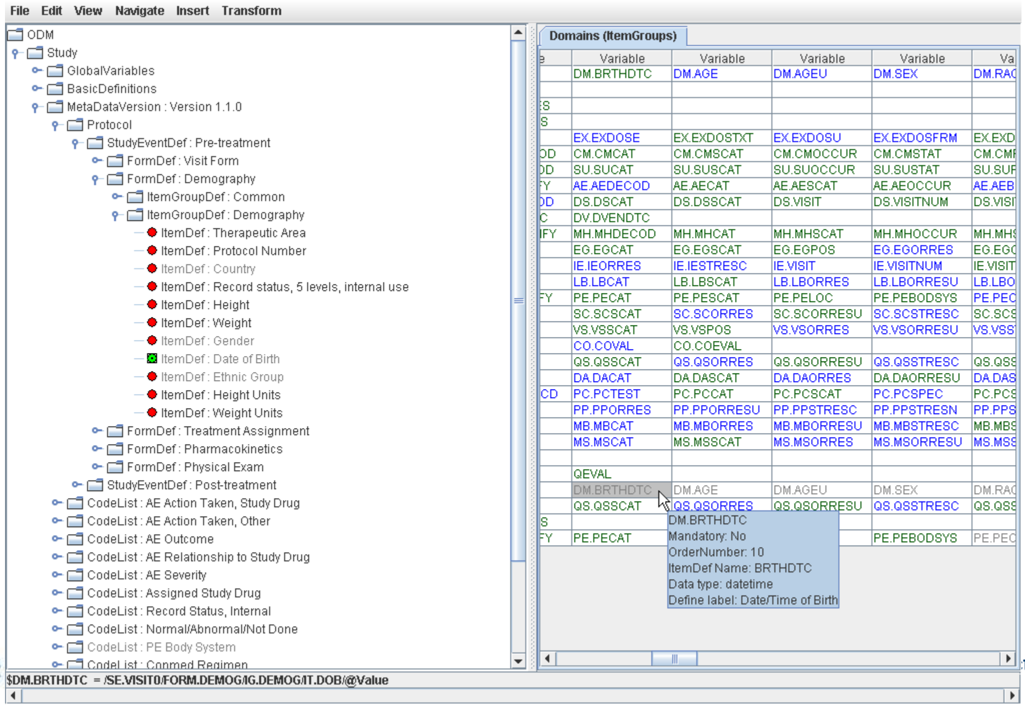
```
   ITEMGROUPDATASEQ       STUDYID DOMAIN   USUBJID  VSSEQ VSTESTCD  \
0                 1   CDISCPILOT01     VS   CDISC001      1    DIABP
1                 2   CDISCPILOT01     VS   CDISC001      2    DIABP
2                 3   CDISCPILOT01     VS   CDISC001      3    DIABP
3                 4   CDISCPILOT01     VS   CDISC001      4    DIABP
4                 5   CDISCPILOT01     VS   CDISC001      5    DIABP
```

Commercial Solutions

Summary of the Commercial Solutions Developed During the Hackathon

SDTM-ETL

The commercial SDTM-ETL mapping software (for generation of SDTM and SEND datasets including the corresponding define.xml) allows to generate the submission datasets in XPT, Dataset-JSON, Dataset-XML and CSV format.



- ODM
  - Study
    - GlobalVariables
    - BasicDefinitions
    - MetaDataVersion : Version 1.1.0
      - Protocol
      - StudyEventDef : Pre-treatment
        - FormDef : Visit Form
        - FormDef : Demography
          - ItemGroupDef : Common
          - ItemGroupDef : Demography
            - ItemDef : Therapeutic Area
            - ItemDef : Protocol Number
            - ItemDef : Country
            - ItemDef : Record status, 5 levels, internal use
            - ItemDef : Height
            - ItemDef : Weight
            - ItemDef : Gender
            - ItemDef : Date of Birth
            - ItemDef : Ethnic Group
            - ItemDef : Height Units
            - ItemDef : Weight Units
        - FormDef : Treatment Assignment
        - FormDef : Pharmacokinetics
        - FormDef : Physical Exam
      - StudyEventDef : Post-treatment
      - CodeList : AE Action Taken, Study Drug
      - CodeList : AE Action Taken, Other
      - CodeList : AE Outcome
      - CodeList : AE Relationship to Study Drug
      - CodeList : AE Severity
      - CodeList : Assigned Study Drug
      - CodeList : Record Status, Internal
      - CodeList : Normal/Abnormal/Not Done
      - CodeList : PE Body System
      - CodeList : Conmed Regimen

e	Variable	Variable	Variable	Variable	Va
	DM.BRTHDTC	DM.AGE	DM.AGEU	DM.SEX	DM.RAC
ES					
S					
	EX.EXDOSE	EX.EXDOSTXT	EX.EXDOSU	EX.EXDOSFRM	EX.EXD
OD	CM.CMCAT	CM.CMSCAT	CM.CMOCCUR	CM.CMSTAT	CM.CMF
DD	SU.SUCAT	SU.SUSCAT	SU.SUOCCUR	SU.SUSTAT	SU.SUF
Y	AE.AEDECOD	AE.AECAT	AE.AESCAT	AE.AEOCCUR	AE.AEB
DD	DS.DSCAT	DS.DSSCAT	DS.VISIT	DS.VISITNUM	DS.VISI
C	DV.DVENDTC				
IFY	MH.MHDECOD	MH.MHCAT	MH.MHSCAT	MH.MHOCCUR	MH.MHS
	EG.EGCAT	EG.EGSCAT	EG.EGPOS	EG.EGORRES	EG.EGC
	IE.IEORRES	IE.IESTRESC	IE.VISIT	IE.VISITNUM	IE.VISIT
	LB.LBCAT	LB.LBSCAT	LB.LBORRES	LB.LBORRESU	LB.LBO
FY	PE.PECAT	PE.PESCAT	PE.PELOC	PE.PEBODSYS	PE.PEC
	SC.SCSCAT	SC.SCORRES	SC.SCORRESU	SC.SCSTRESC	SC.SCS
	VS.VSSCAT	VS.VSPOS	VS.VSORRES	VS.VSORRESU	VS.VSS
	CO.COVAL	CO.COEVAL			
	QS.QSSCAT	QS.QSORRES	QS.QSORRESU	QS.QSSTRESC	QS.QSS
	DA.DACAT	DA.DASCAT	DA.DAORRES	DA.DAORRESU	DA.DAS
CD	PC.PCTEST	PC.PCCAT	PC.PCSCAT	PC.PCSPEC	PC.PCS
	PP.PPORRES	PP.PPORRESU	PP.PPSTRESC	PP.PPSTRESN	PP.PPS
	MB.MBCAT	MB.MBORRES	MB.MBORRESU	MB.MBSTRESC	MB.MBS
	MS.MSCAT	MS.MSSCAT	MS.MSORRES	MS.MSORRESU	MS.MSS
	QEVAL				
	DM.BRTHDTC	DM.AGE	DM.AGEU	DM.SEX	DM.RAC
	QS.QSSCAT	QS.QSORRES	QS.QSORRESU	QS.QSSTRESC	QS.QSS
S					
FY	PE.PECAT			PE.PEBODSYS	PE.PEC

DM.BRTHDTC
Mandatory: No
OrderNumber: 10
ItemDef Name: BRTHDTC
Data type: datetime
Define label: Date/Time of Birth

$DM.BRTHDTC = /SE.VISIT0/FORM.DEMOG/IG.DEMOG/IT.DOB/@Value

Author: Jozef Aerts

Website: www.xml4pharma.com/SDTM-ETL

Conclusion

Summary of Hackathon and Future Events

Overall Impressions of Dataset-JSON

- Dataset-JSON works:
  - As a general data exchange
  - As a general dataset format
  - For use with web-based APIs
  - Works with a wide-range of programming languages and technology stack
  - Simple to process
  - Easy to transform into SAS datasets, R or Python dataframes, and CSV
- Dataset-JSON file sizes are smaller than SAS XPORT v5 and Dataset-XML
- Dataset-JSON is row-based - typically transformed into datasets for analysis
- JSON is a language, platform independent data exchange format

Updates to Dataset-JSON

- Add useful ODM attributes:
  - ODMVersion, FileOID, PriorFileOID, CreationDateTime, AsOfDateTime…
- API reference implementation:
  - A common API achieves this better than individually-stored copies of a .XML document
  - Currently planned as a supplement to ODM v2.0

Next Steps

- Dec 8th COSA Spotlight Webinar demonstrating the Dataset-JSON Hackathon solutions
- Add Dataset-JSON Hackathon solutions to the COSA Repository Directory
- ODM v2.0 Public Review begins Nov. 8th
    - Dataset-JSON is part of ODM v2.0
    - Chance to comment
- Next COSA Hackathon: **admiral**
    - An open source, modularized toolbox that enables the collaborative development of ADaM datasets in R
    - Training: Tuesday, 17 January, 10AM - 1PM ET
    - Hackathon Kickoff: Thursday, 26 January, 10AM - 12PM ET
    - Hackathon: 1-28 February

Thank You!

Dataset JSON Hackathon 2022

cdisc