Transfer SDTM dataset in XPT format to RDF using D2RQ

mja@statgroup.dk

2017-11-02

Contents

Introduction

This script creates RDF version of a selected SDTM transport file using D2RQ (www.d2rq.org).

The approach takes the following steps: - import .xpt file into R as a data.frame - determine field types, e.g. date, for fields - write data.frame to a table in a database - define primary keys for the table - use D2RQ to transfer the table to RDF

The main guiding principle for this script was to use existing tools and their default settings. So, much of the of code can be improved.

Notes on other approaches for creating RDF from SDTM transport files - Importing the .xpt file into SAS will give the proper formatting for dates automatically. - SAS can export to a database, or expose SAS dataset with ODBC (tried it) or JDBC (did not get it to work). - D2RQ can read from a JDBC connection. - There are also systems supporting accessing a relational database using SPARQL. For example: Virtuoso (https://virtuoso.openlinksw.com/), Oracle Spatial Graph (http://www.oracle.com/technetwork/database/options/spatialandgraph/overview/rdfsemantic-graph-1902016.html).

Dependencies

R-packages

The following R-packages are need:

```
install.packages("SASxport")
install.packages("RMySQL")
```

D2RQ

D2RQ allows "Accessing Relational Databases as Virtual RDF Graphs", see more and download from (www.d2rq.org).

MariaDB

MariaDB is a open source database solution, see more on (https://mariadb.com/). There are installation packages for most operating systems (Windows, MacOS, Linux).

Another database solutions can be used. MariaDB is used here as example as it works easily with R using the RMySQL packages.

Setup

```
library("SASxport")
library("RMySQL")
```

```
targetDir<- "./res-ttl"
ttlDir<- "./res-ttl"
cdiscpilot01mapttlFn<- "cdiscpilot01-d2rq-mapping.ttl"
cdiscpilot01mapttlPath<- file.path(tempdir(), cdiscpilot01mapttlFn)
cdiscpilot01ttlFn<- "cdiscpilot01.ttl"
cdiscpilot01ttlPath<- file.path(tempdir(), cdiscpilot01ttlFn)
rqDir<- "./sparql-rq"
SDTMxptdir<-"../phuse-scripts/data/sdtm/cdiscpilot01"
# Does not work ... SDTMxptdir<-"http://github.com/phuse-org/phuse-scripts/master/data/sdtm/cdiscpilot0
```

Show parameters

```
param.df<- data.frame(filepathname=apply(array(c(targetDir, ttlDir, rqDir, cdiscpilot01mapttlPath, cdisc
rownames(param.df)<-c("Target diretory", ".ttl files for data cubes", "SPARQL scripts dir", "D2RQ mappi
knitr::kable(param.df)
```

	filepathname
Target diretory	/home/ma/projects/xpt2rdf/res-ttl
.ttl files for data cubes	/home/ma/projects/xpt2rdf/res-ttl
SPARQL scripts dir	./sparql-rq
D2RQ mapping file for cdiscpilot01 data	/tmp/RtmpSDq9XK/cdiscpilot01-d2rq-mapping.ttl

	filepathname
D2RQ file for cdiscpilot01 data	/tmp/RtmpSDq9XK/cdiscpilot01.ttl
Directory with SDTM xpt files	/home/ma/projects/phuse-scripts/data/sdtm/cdiscpilot01

Before executing the commands below, MariaDB must be started. For MariaDB on fedora see (https://fedoraproject.org/wiki/MariaDB):

```
systemctl start mariadb
```

Identifying datasets and columns used in RDF data cubes

The SDTM datasets to process is stored as follows:

```
xptfilesAndPrimaryKey<- data.frame(
    matrix(c(
                "dm", "USUBJID",
                "suppdm", "QNAM, USUBJID"
            ), ncol=2, byrow=TRUE )
  , stringsAsFactors=FALSE)
colnames(xptfilesAndPrimaryKey)<- c("dataset", "primaryKey")
parts<- xptfilesAndPrimaryKey
knitr::kable(parts)
```

dataset	primaryKey
dm	USUBJID
suppdm	QNAM, USUBJID

The the variables for the primary key is defined by directly. This could be automated by extracting the key fields from `define-xml`. However, manual tweaking or complex programming is needed, as some of the specified key variables have missing values, which is not possible for variables being part of a primary key in mariadb (https://mariadb.com/kb/en/mariadb/primary-keys-with-nullable-columns/).

For SUPPDM the primary key is given as "QNAM, USUBJID" as by default D2RQ creates the object using the reversed order of the fields in the primary key.

The following verifies existence of the .xpt file together with the structure defined in the transport file.

```
for (i in seq(nrow(xptfilesAndPrimaryKey))) {
        dsname<- xptfilesAndPrimaryKey$dataset[i]
        xpt.fn<- file.path( SDTMxptdir, paste0(tolower(dsname), ".xpt"))
        cat("Reading ", xpt.fn, "\n")
        adds.full<- SASxport::read.xport(xpt.fn, as.is=TRUE)
# knitr::kable(head(adds.full),caption=paste("Data in xpt file (first 10 rows):", xpt.fn))
    adds.def<- lookup.xport(xpt.fn)[[toupper(dsname)]]
    adds.contents<- data.frame(adds.def$name, adds.def$label, adds.def$type, adds.def$format)
# knitr::kable(adds.contents , caption=paste("Structure of xpt file:", xpt.fn) )
    }
```

```
## Reading  ../phuse-scripts/data/sdtm/cdiscpilot01/dm.xpt
## Reading  ../phuse-scripts/data/sdtm/cdiscpilot01/suppdm.xpt
```

Converting .xpt files to mariadb

The R to mariadb conversion needs to specify data types, specifically, the length of strings and date and data time objects. This is done variable by variable. The approach is easier to debug than doing it per dataset (using `apply`).

Please note: The following use of passwords in plain text is not recommeded. MariaCB and R documentation Doc suggest to connect to my-db as defined in ~/.my.cnf. See (https://github.com/rstats-db/RMySQL), (https://mariadb.com/kb/en/mariadb/configuring-mariadb-with-mycnf/), (http://stackoverflow.com/questions/2482234/how-to-know-mysql-my-cnf-location).

As first step create an empty database for the transfer.

```
con <- dbConnect(RMySQL::MySQL(), username="root", password="mariadb")
res <- dbSendQuery(con, paste( "DROP  DATABASE IF EXISTS cdiscpilot01;"))
res <- dbSendQuery(con, paste( "CREATE DATABASE cdiscpilot01;"))
dbDisconnect(con)
```

```
## Warning: Closing open result sets
```

```
## [1] TRUE
```

Now write the contents of the .xpt files to the database.

```
con <- dbConnect(RMySQL::MySQL(), username="root", password="mariadb", dbname="cdiscpilot01" )
dbListTables(con)
```

```
## character(0)
```

```
for (i in seq(nrow(xptfilesAndPrimaryKey))) {
    dsname<- xptfilesAndPrimaryKey$dataset[i]
    xpt.fn<- file.path( SDTMxptdir, paste0(tolower(dsname), ".xpt"))
    cat("Reading ", xpt.fn, "\n")
    adds.full<- NULL
    adds.xfer<- NULL
    adds.full<- SASxport::read.xport(xpt.fn, as.is=TRUE)
    adds.def<- lookup.xport(xpt.fn)[[toupper(dsname)]]
    field.type.def<- list(  )
    ## incase only transferring a subset of variables
    ## adds.xfer<- adds.full[,adds.def$name %in% adds.xfer.columns]
    adds.xfer<- adds.full
    for (j in seq(length(adds.def$name))) {
        vn<- adds.def$name[j]
        vtype<- adds.def$type[j]
        vformat<- adds.def$format[j]
        cat(vn, vtype, vformat, "\n")
        if (is.character(vtype) && is.character(vformat) ) {
            if ( vtype=="numeric" ) {
                if (vformat=="DATE") {
                    adds.xfer[,vn]<- as.Date(adds.xfer[,vn])
                    field.type.def[[vn]]<- "date"
                } else  { ## add more, consider using switch
                    field.type.def[[vn]]<- "double"
                }
            } else if ( vtype=="character" ) {
                vlen<- max(nchar(adds.xfer[,vn]))
                field.type.def[[vn]]<- paste0("varchar","(", vlen, ")")
            } else {
```

```
                cat("--> ", "Unhandled: ", vn, vtype, vformat, "\n")
            }
        }
    }
    str(field.type.def)
        dbWriteTable(con, dsname, adds.xfer, field.type=field.type.def, overwrite=TRUE, row.names=FALSE)
        sqlcmd<- paste0("ALTER TABLE ", dsname,
                        " ADD PRIMARY KEY", "(", xptfilesAndPrimaryKey$primaryKey[i], ")",
                        collapse=" " )
        cat("sql cmd: ", sqlcmd, "\n")
       res <- dbSendQuery(con, sqlcmd )
        cat("Done\n")
}
```

```
## Reading  ../phuse-scripts/data/sdtm/cdiscpilot01/dm.xpt
## STUDYID character
## DOMAIN character
## USUBJID character
## SUBJID character
## RFSTDTC character
## RFENDTC character
## RFXSTDTC character
## RFXENDTC character
## RFICDTC character
## RFPENDTC character
## DTHDTC character
## DTHFL character
## SITEID character
## AGE numeric
## AGEU character
## SEX character
## RACE character
## ETHNIC character
## ARMCD character
## ARM character
## ACTARMCD character
## ACTARM character
## COUNTRY character
## DMDTC character
## DMDY numeric
## List of 25
##  $ STUDYID : chr "varchar(12)"
##  $ DOMAIN  : chr "varchar(2)"
##  $ USUBJID : chr "varchar(11)"
##  $ SUBJID  : chr "varchar(4)"
##  $ RFSTDTC : chr "varchar(10)"
##  $ RFENDTC : chr "varchar(10)"
##  $ RFXSTDTC: chr "varchar(10)"
##  $ RFXENDTC: chr "varchar(10)"
##  $ RFICDTC : chr "varchar(0)"
##  $ RFPENDTC: chr "varchar(16)"
##  $ DTHDTC  : chr "varchar(10)"
##  $ DTHFL   : chr "varchar(1)"
##  $ SITEID  : chr "varchar(3)"
```

```
##  $ AGE     : chr "double"
##  $ AGEU    : chr "varchar(5)"
##  $ SEX     : chr "varchar(1)"
##  $ RACE    : chr "varchar(32)"
##  $ ETHNIC  : chr "varchar(22)"
##  $ ARMCD   : chr "varchar(8)"
##  $ ARM     : chr "varchar(20)"
##  $ ACTARMCD: chr "varchar(8)"
##  $ ACTARM  : chr "varchar(20)"
##  $ COUNTRY : chr "varchar(3)"
##  $ DMDTC   : chr "varchar(10)"
##  $ DMDY    : chr "double"
## sql cmd:  ALTER TABLE dm ADD PRIMARY KEY(USUBJID)
## Done
## Reading  ../phuse-scripts/data/sdtm/cdiscpilot01/suppdm.xpt
## STUDYID character
## RDOMAIN character
## USUBJID character
## IDVAR character
## IDVARVAL character
## QNAM character
## QLABEL character
## QVAL character
## QORIG character
## QEVAL character
## List of 10
##  $ STUDYID : chr "varchar(12)"
##  $ RDOMAIN : chr "varchar(2)"
##  $ USUBJID : chr "varchar(11)"
##  $ IDVAR   : chr "varchar(0)"
##  $ IDVARVAL: chr "varchar(0)"
##  $ QNAM    : chr "varchar(8)"
##  $ QLABEL  : chr "varchar(37)"
##  $ QVAL    : chr "varchar(1)"
##  $ QORIG   : chr "varchar(7)"
##  $ QEVAL   : chr "varchar(22)"
## sql cmd:  ALTER TABLE suppdm ADD PRIMARY KEY(QNAM, USUBJID)
## Done
```

```
dbDisconnect(con)
```

```
## Warning: Closing open result sets
```

```
## [1] TRUE
```

Use D2RQ to transfer to .ttl files

d2rq is used to convert the database. First a mapping file is generated and customized. The `d2rq:classDefinitionLabel` could also be changed to correspond to the SAS label.

Windows specific setup

If d2rq is located on a different drive, then the setting of D2RQ_ROOT in the scripts should be changed to

`set D2RQ_ROOT=%~dp0`

to include the drive name (the `d` in `%~dp0`).

Setup

```
d2rqbaseURL<- "http://www.example.org/cdiscpilot01/"
d2rqjdbcstring<- '"jdbc:mariadb://localhost:3306/cdiscpilot01?user=root&password=mariadb"'
if (.Platform$OS.type=="windows") {
    d2rq.generatemapping<- "c:/opt/d2rq-0.8.1/generate-mapping.bat"
    d2rq.dumprdf<- "c:/opt/d2rq-0.8.1/dump-rdf.bat"
} else {
    d2rq.generatemapping<- "/opt/d2rq-0.8.1/generate-mapping"
    d2rq.dumprdf<- "/opt/d2rq-0.8.1/dump-rdf"
    }
```

Generate mapping

```
## -b option does not work with -l
## /opt/d2rq-0.8.1/generate-mapping reports Unknown argument: -b
system(
    paste( d2rq.generatemapping,
          " -o ", cdiscpilot01mapttlPath,
          " --verbose",
             " ", d2rqjdbcstring, " "
          )
)

d2rqmap<- readLines(cdiscpilot01mapttlPath)
if (d2rqmap[1]== "@prefix map: <#> .") {
    d2rqmap[1]<- "@prefix map: <http://www.example.org/cdiscpilot01/db/map/> ."
}
if (d2rqmap[2]== "@prefix db: <> .") {
    d2rqmap[2]<- "@prefix db: <http://www.example.org/cdiscpilot01/db/> ."
}
if (d2rqmap[3]== "@prefix vocab: <vocab/> .") {
    d2rqmap[3]<- "@prefix vocab: <http://www.example.org/cdiscpilot01/datasets/vocab/> ."
}
fp<- file.path(targetDir, cdiscpilot01mapttlFn)
writeLines(d2rqmap, fp )
cat( "File ", fp, " created", "\n")
```

```
## File  ./res-ttl/cdiscpilot01-d2rq-mapping.ttl  created
```

Dump database as .ttl

```
system(paste( d2rq.dumprdf,
             " -b ", d2rqbaseURL,
             " --verbose",
             " -o ", cdiscpilot01ttlPath,
```

```
            " ", fp, " "
            ))
if (file.copy(cdiscpilot01ttlPath, targetDir, overwrite = TRUE)) {
    cat( "File ", cdiscpilot01ttlPath, " copied to directory ", targetDir, "\n")
}
```

```
## File  /tmp/RtmpSDq9XK/cdiscpilot01.ttl  copied to directory  ./res-ttl
```

The .ttl file is not stored at github.

Configuration of mariadb

If RMySQL gives an errormessage

After installation of mariadb

`install.packages("RMySQL")`

gives error

`libmariadb.so.2: cannot open shared object file: No such file or directory`

but the files exists as

`/usr/lib64/mariadb/libmariadb.so.2`

and the prerequisites in https://cran.r-project.org/web/packages/RMySQL/README.html are met.

My solution was to modify '/etc/ld.so.conf.d/mariadb-x86_64.conf' by adding a reference to

`/usr/lib64/mariadb`

Installing MariaDB on fedora

Follow the instructions at (https://fedoraproject.org/wiki/MariaDB).

For my setup I start mariadb for each session by

`systemctl start mariadb`

Alternatively, could be to have mariadb running as service.

D2RQ jdbc and mariadb

For d2rq jdbc - get mariadb-java-client from (). I used `mariadb-java-client-1.5.2.jar` and store it in `/opt/d2rq-0.8.1/lib/db-drivers/`.

Final stuff

Here is how to make the PDF file

```
rmarkdown::render('SDTM-ds-as-ttl-using-d2rq.Rmd', c("html_document", "pdf_document"), clean=TRUE)
```