openva_pipeline Documentation

Release 0.0.0.9000

Jason Thomas, Samuel J. Clark, and Martin Bratschi

Oct 25, 2018

CONTENTS:

The OpenVA Pipeline automates the processing of verbal autopsy (VA) data from an ODK Aggregate server, through the openVA library from the R statistical software, and ending with a DHIS2 server (with the VA program). A friendly graphical user interface to R is also provided via an RShiny app and vignette.

CHAPTER

ONE

SOFTWARE REQUIREMENTS

The following software is required by the openVA pipeline (note: installation instructions are found on a different page)

- **Python 3.5.0 (or higher)**
  - PIP (tool for installing Python packages)
- OpenJDK or Java JDK 7 or 8,
- R (OpenVA Pipeline was tested on Version 3.4.3)
- SQLite3
- SQLCipher
- ODK Briefcase (OpenVA Pipeline was tested on Version v1.10.1)

It is also useful to install DB Browser for SQLite. This optional tool is useful for configuring the SQLite Database.

CHAPTER

TWO

INSTALLATION GUIDE

The following instructions guide the installation of the OpenVA Pipeline on Ubuntu 16.04 operating system.

> **Note:** To make the installation process easier, all of the required software can be installed by downloading and running the bash script install_software.sh located in the main folder of the OpenVA_Pipeline repository.

1. Install Python, OpenJDK, **R**, SQLite3, SQLCipher, and Git by typing the following commands at a terminal prompt (indicated by $)

```
$ sudo apt update
$ sudo apt install python3-pip python3-dev openjdk-8-jre r-base r-
↪cran-rjava sqlite3 libsqlite3-dev sqlcipher libsqlcipher-dev git␣
↪-y
```

2. Download the ODK-Briefcase-v1.10.1.jar file to the same folder where *pipeline.py* is located.

3. In a terminal, start **R** by simply typing `R` at the prompt, or use `sudo R` for system-wide installation of the packages. The necessary packages can be installed (with an internet connection) using the following command:

```
> install.packages(c("openVA", "CrossVA"), dependencies=TRUE)
```

   Note that > in the previous command indicates the **R** prompt (not part of the actual command). This command will prompt the user to select a CRAN mirror (choose a mirror close to your geographic location). After the installation of the packages has been completed, you can exit **R** with the following command:

```
> q('no')
```

4. Python is pre-installed with Ubuntu 16.04, but additional packages and modules are needed, which can be installed with the following commands at a terminal:

```
$ pip3 install --upgrade pip --user
$ hash -d pip3
$ pip3 install --upgrade setuptools --user
$ pip3 install requests pysqlcipher3 --user
```

Note: the first command: `pip3 install --upgrade pip --user` will produce a warning message:

```
You are using pip version 8.1.1, however version 10.0.1 is␣
↪available.
You should consider upgrading via the 'pip install --upgrade pip'␣
↪command.
```

However, after running the command `hash -d pip3`, the command `pip3 --version` shows that version 10.0.1 is indeed installed.

5. Install DB Browser for SQLite with the commands

```
$ sudo apt install build-essential git-core cmake libsqlite3-dev␣
↪qt5-default qttools5-dev-tools libsqlcipher-dev -y
$ git clone https://github.com/sqlitebrowser/sqlitebrowser
$ cd sqlitebrowser
$ mkdir build
$ cd build
$ cmake -Dsqlcipher=1 -Wno-dev ..
$ make
$ sudo make install
```

Instructions for installing JDK 8 on Ubuntu 16.04 can be found here. After installing JDK 8, run the following command at the terminal to properly configure **R**

```
$ sudo R CMD javareconf
```

and then install the **R** packages (as described above).

CHAPTER

THREE

PIPELINE CONFIGURATION

1. **Create the SQLite database**: The openVA pipeline uses an SQLite database to access configuration settings for ODK Aggregate, openVA in R, and DHIS2. Error and log messages are also stored to this database, along with the VA records downloaded from ODK Aggregate and the assigned COD.

   (a) The necessary tables and schema are created in the SQL file pipelineDB.sql, which can be downloaded from the OpenVA_Pipeline GitHub webpage. Create the SQLite database in the same folder as the file *pipeline.py*.

   (b) Use SQLCipher to create the pipeline database, assign an encryption key, and populate the database using the following commands (note that the `$` is the terminal prompt and `sqlite>` is the SQLite prompt, i.e., not part of the commands).

```
$ sqlcipher
sqlite> .open Pipeline.db
sqlite> PRAGMA key=encryption_key;
sqlite> .read "pipelineDB.sql"
sqlite> .tables
sqlite> -- take a look --
sqlite> .schema ODK_Conf
sqlite> SELECT odkURL from ODK_Conf;
sqlite> .quit
```

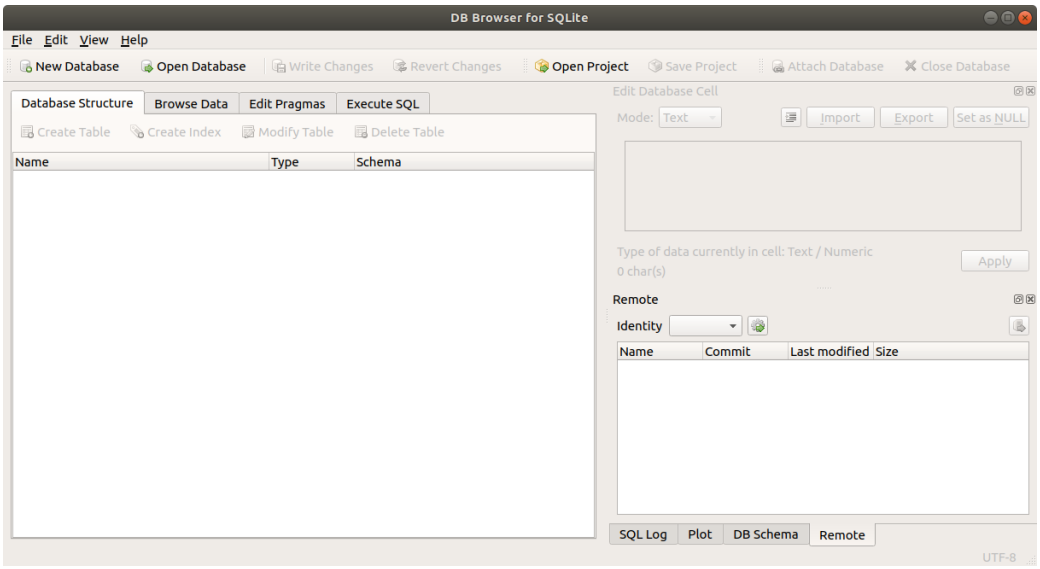
   Note how the pipeline database is encrypted, and can be accessed via with SQLite command: `PRAGMA key = "encryption_key;"`

```
$ sqlcipher
sqlite> .open Pipeline.db
sqlite> .tables

Error: file is encrypted or is not a database

sqlite> PRAGMA key = "encryption_key";
sqlite> .tables
sqlite> .quit
```

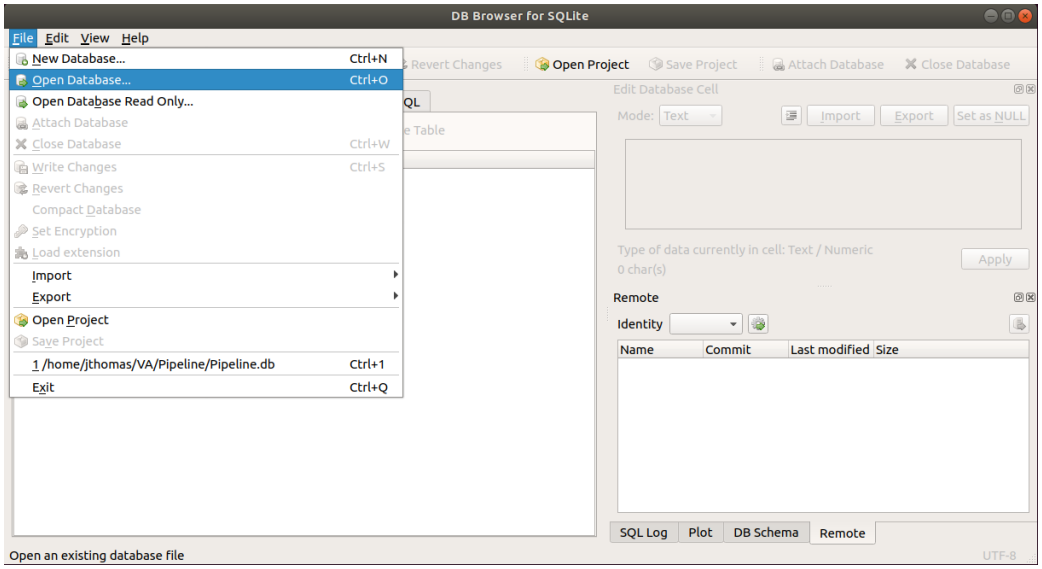
(c) Open the file *pipeline.py* and edit the first two lines of code by including the name of the pipeline SQLite database (the default is *Pipeline.db*) and the encryption password. The lines in *pipeline.py* are:

```
#--------------------------------------------#
# User Settings
sqlitePW = "enilepiP"
dbName   = "Pipeline.db"
#--------------------------------------------#
```

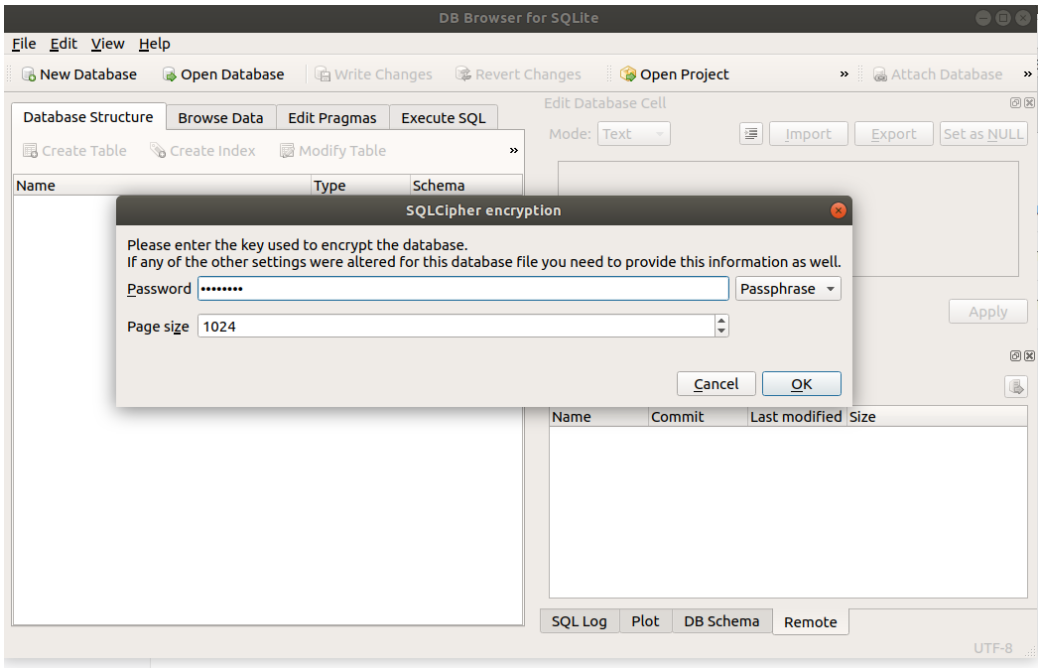
2. **Configure Pipeline**: The pipeline connects to ODK Aggregate and DHIS2 servers and thus requires usernames, passwords, and URLs. Arguments for openVA should also be supplied. We will use DB Browser for SQLite to configure these settings. Start by launching DB Browser from the terminal, which should open the window below `$ sqlitebrowser`



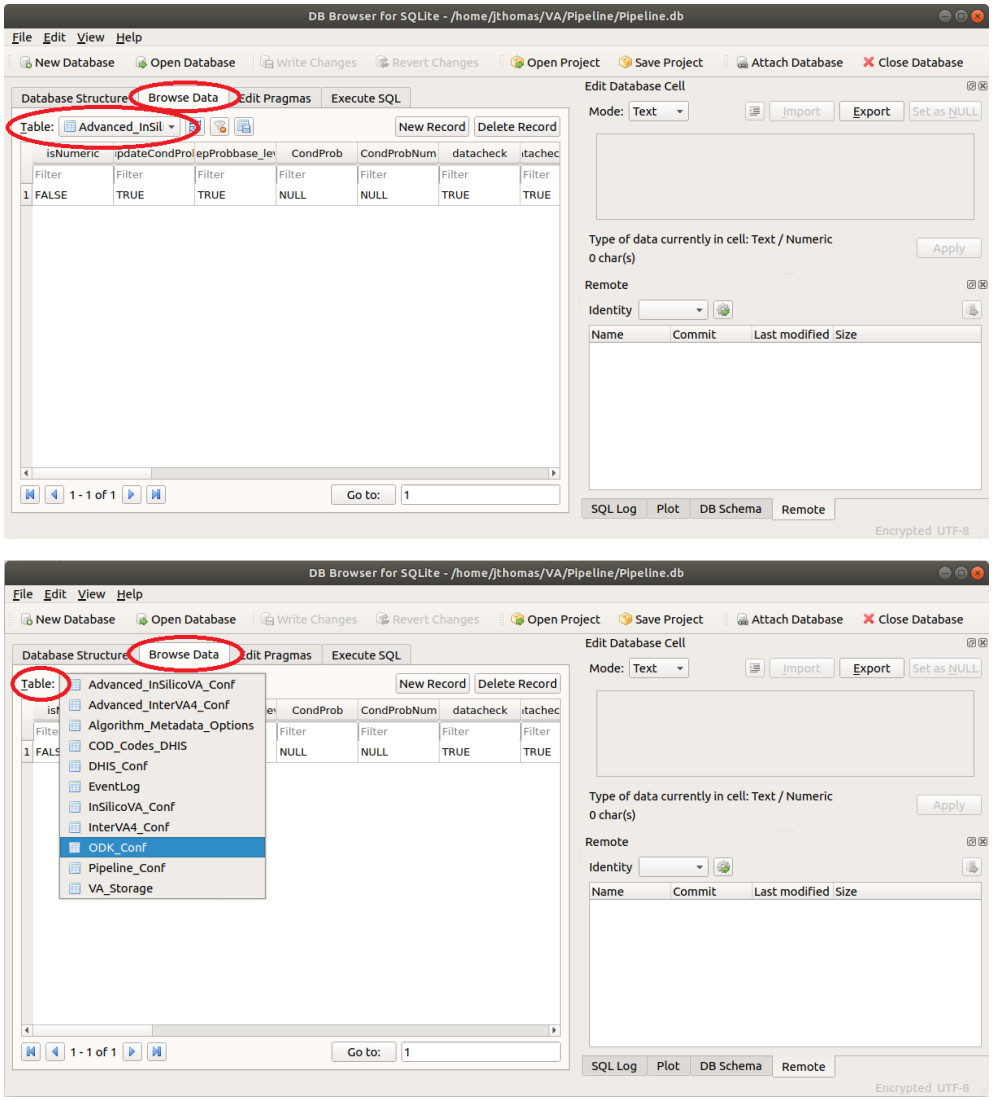
Next, open the database by selecting the menu options: *File -> Open Database...*



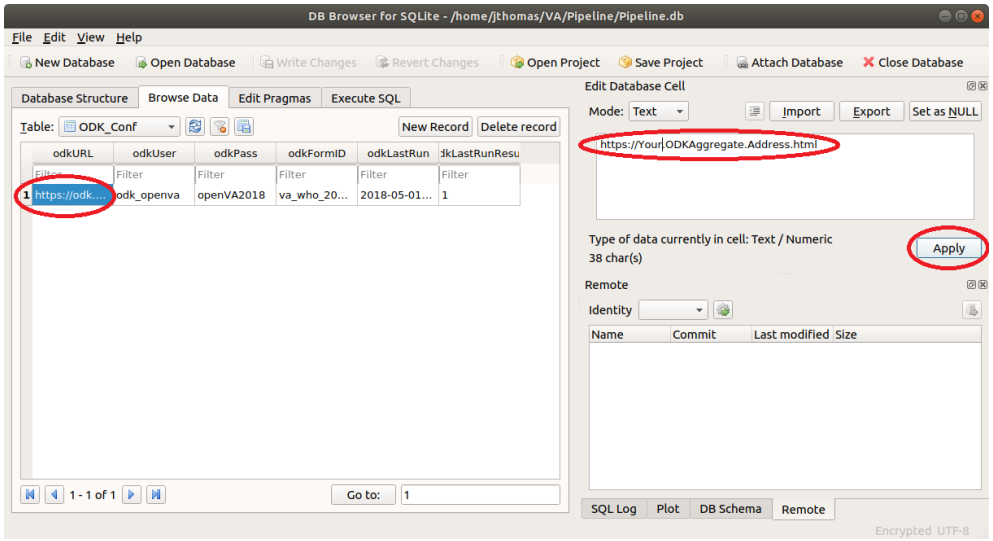
and navigate to the *Pipeline.db* SQLite database and click the *Open* button. This will prompt you to enter in encryption password.



(a) **ODK Configuration**: To configure the pipeline connection to ODK Aggregate, click on the *Browse Data* tab and select the ODK_Conf table as shown below.



Now, click on the *odkURL* column, enter the URL for your ODK Aggregate server, and click *Apply*.



Similarly, edit the *odkUser*, *odkPass*, and *odkFormID* columns so they contain a valid user name, password, and Form ID (see Form Management on ODK Aggregate server) of the VA questionnaire of your ODK Aggregate server.

(b) **openVA Configuration**: The pipeline configuration for openVA is stored in the *Pipeline_Conf* table. Follow the steps described above (in the ODK Aggregate Configuration section) and edit the following columns:

- *workingDirectory* – the directory where the pipeline files (i.e., *pipeline.py*, *Pipeline.db* and the ODK Briefcase application, *ODK-Briefcase-v1.10.1.jar*). Note that the pipeline will create new folders and files in this working directory, and must be run by a user with privileges for writing files to this location.
- *openVA_Algorithm* – currently, there are only two acceptable values for the alogrithm are `InterVA` or `Insilico`
- *algorithmMetadataCode* – this column captures the necessary inputs for producing a COD, namely the VA questionnaire, the algorithm, and the symptom-cause information (SCI) (see [below](#SCI) for more information on the SCI). Note that there are also different versions (e.g., InterVA 4.01 and InterVA 4.02, or WHO 2012 questionnare and the WHO 2016 instrument/questionnaire). It is important to keep track of these inputs in order to make the COD determination reproducible and to fully understand the assignment of the COD. A list of all algorith metadata codes is provided in the *dhisCode* column in the *Algorithm_Metadata_Options* table. The logic for each code is

  algorith|algorithm version|SCI|SCI version|instrument|instrument version

- *codSource* – both the InterVA and InSilicoVA algorithms return CODs from a list produced by the WHO, and thus this column should be left at the default value of `WHO`.

(c) **DHIS2 Configuration**: The pipeline configuration for DHIS2 is located in the

DHIS_Conf table, and the following columns should be edited with appropriate values for your DHIS2 server.

- *dhisURL* – the URL for your DHIS2 server
- *dhisUser* – the username for the DHIS2 account
- *dhisPass* – the password for the DHIS2 account
- *dhisOrgUnit* – the Organization Unit (e.g., districts) UID to which the verbal autopsies are associated. The organisation unit must be linked to the Verbal Autopsy program. For more details, see the DHIS2 Verbal Autopsy program installation guide

CHAPTER

FOUR

DOCUMENTATION FOR CLASSES, FUNCTIONS, AND METHODS

4.1 Main Interface

The OpenVA Pipeline is run using the following function

runPipeline.**runPipeline**(*database_file_name*, *database_directory*, *database_key*, *export_to_DHIS=True*)

Runs through all steps of the OpenVA Pipeline.

Parameters

- **database_file_name** – File name for the Transfer Database.
- **database_directory** – Path of the Transfer Database.
- **datatbase_key** – Encryption key for the Transfer Database
- **export_to_DHIS** (*(Boolean)*) – Indicator for posting VA records to a DHIS2 server.

4.2 API for Transfer Database

class transferDB.**TransferDB**(*dbFileName*, *dbDirectory*, *dbKey*, *plRunDate*)

This class handles interactions with the Transfer database.

The Pipeline accesses configuration information from the Transfer database, and also stores log messages and verbal autopsy records in the DB. The Transfer database is encrypted using sqlcipher3 (and the pysqlcipher3 module is imported to establish DB connection).

dbFileName [str] File name of the Tranfser database.

dbDirectory [str] Path of folder containing the Transfer database.

dbKey [str] Encryption key for the Transfer database.

plRunDate [date] Date when pipeline started latest run (YYYY-MM-DD_hh:mm:ss).

connectDB(self) Returns SQLite Connection object to Transfer database.

configPipeline(self, conn) Accepts SQLite Connection object and returns tuple with configuration settings for the Pipeline.

configODK(self, conn) Accepts SQLite Connection object and returns tuple with configuration settings for connecting to ODK Aggregate server.

configOpenVA(self, conn) Accepts SQLite Connection object and returns tuple with configuration settings for R package openVA.

checkDuplicates(self) Search for duplicate VA records in ODK Briefcase export file and the tranfser DB.

configDHIS(self, conn) Accepts SQLite Connection object and returns tuple with configuration settings for connecting to DHIS2 server.

storeVA(self) Deposits VA objects to the Transfer DB.

makePipelineDirs(self) Returns SQLite Connection object to Transfer database.

cleanODK(self) Remove ODK Briefcase Export files.

cleanOpenVA(self) Remove openVA files with COD results.

cleanDHIS(self) Remove DHIS2 blob files.

checkDuplicates(*conn*)
Search for duplicate VA records.

This method searches for duplicate VA records in ODK Briefcase export file and the Tranfser DB. If duplicates are found, a warning message is logged to the EventLog table in the Transfer database.

conn : sqlite3 Connection object

DatabaseConnectionError PipelineError

cleanDHIS()
Remove DHIS2 blob files.

cleanODK()
Remove ODK Briefcase Export files.

cleanOpenVA()
Remove openVA files with COD results.

configDHIS(*conn*, *algorithm*)
Query DHIS configuration settings from database.

This method is intended to be used in conjunction with (1) TransferDB.connectDB(), which establishes a connection to a database with the Pipeline configuration settings;

and (2) DHIS.connect(), which establishes a connection to a DHIS server. Thus, TransferDB.configDHIS() gets its input from TransferDB.connectDB() and the output from TransferDB.config() is a valid argument for DHIS.config().

conn [sqlite3 Connection object (e.g., the object returned from] TransferDB.connectDB())

algorithm : VA algorithm used by R package openVA

tuple Contains all parameters for DHIS.connect().

DHISConfigurationError

configODK(*conn*)

Query ODK configuration settings from database.

This method is intended to be used in conjunction with (1) TransferDB.connectDB(), which establishes a connection to a database with the Pipeline configuration settings; and (2) ODK.briefcase(), which establishes a connection to an ODK Aggregate server. Thus, TransferDB.configODK() gets its input from TransferDB.connectDB() and the output from TransferDB.configODK() is a valid argument for ODK.config().

conn : sqlite3 Connection object

tuple Contains all parameters for ODK.briefcase().

ODKConfigurationError

configOpenVA(*conn*, *algorithm*, *pipelineDir*)

Query OpenVA configuration settings from database.

This method is intended to receive its input (a Connection object) from TransferDB.connectDB(), which establishes a connection to a database with the Pipeline configuration settings. It sets up the configuration for all of the VA algorithms included in the R package openVA. The output from configOpenVA() serves as an input to the method OpenVA.setAlgorithmParameters(). This is a wrapper function that calls __configInterVA__, __configInSilicoVA__, and __configSmartVA__ to actually pull configuration settings from the database.

conn : sqlite3 Connection object algorithm : VA algorithm used by R package openVA pipelineDir : Working directory for the Pipeline

tuple Contains all parameters needed for OpenVA.setAlgorithmParameters().

OpenVAConfigurationError

configPipeline(*conn*)

Grabs Pipline configuration settings.

This method queries the Pipeline_Conf table in Transfer database and returns a tuple with attributes (1) algorithmMetadataCode; (2) codSource; (3) algorithm; and (4) workingDirectory.

tuple alogrithmMetadataCode - attribute describing VA data codSource - attribute detailing the source of the Cause of Death list algorithm - attribute indicating which VA algorithm to use workingDirectory - attribute indicating the working directory

PipelineConfigurationError

connectDB()
Connect to Transfer database.

Uses parameters supplied to the parent class, TransferDB, to connect to the (encrypted) Transfer database.

SQLite database connection object Used to query (encrypted) SQLite database.

DatabaseConnectionError

makePipelineDirs()
Create directories for storing files (if they don't exist).

The method creates the following folders in the working directory (as set in the Transfer database table Pipeline_Conf): (1) ODKFiles for files containing verbal autopsy records from the ODK Aggregate server; (2) OpenVAFiles containing R scripts and results from the cause assignment algorithms; and (3) DHIS for holding blobs that will be stored in a data repository (DHIS2 server and/or the local Transfer database).

PipelinError

storeVA(*conn*)
Store VA records in Transfer database.

This method is intended to be used in conjunction with the DHIS module, which prepares the records into the proper format for storage in the Transfer database.

conn : sqlite3 Connection object

PipelineError DatabaseConnectionError

updateODKLastRun(*conn*, *plRunDate*)
Update Transfer Database table ODK_Conf.odkLastRun

conn : sqlite3 Connection object plRunDate : date Date when pipeline started latest run (YYYY-MM-DD_hh:mm:ss)

4.3 API for ODK Briefcase

class odk.**ODK**(*odkSettings*, *workingDirectory*)
Manages Pipeline's interaction with ODK Aggregate.

This class handles the segment of the pipeline related to ODK. The ODK.connect() method calls ODK Briefcase to connect with an ODK Aggregate server and export VA records.

It also checks for previously exported files and updates them as needed. Finally, it logs messages and errors to the pipeline database.

odkSettings [(named) tuple with all of configuration settings as] attributes.

workingDirectory : Directory where openVA Pipeline should create files.

briefcase(self) Uses ODK Briefcase to export VA records from ODK Aggregate server.

mergePrevExport(self) Merge ODK Briefcase export files (if they exist).

ODKBriefcaseError

briefcase()
Export records from ODK Aggregate server using ODK Briefcase.

Longer description here.

Connection object SQL connection object for querying config settings

ODKError

mergeToPrevExport()
Merge previous ODK Briefcase export files.

4.4 API for OpenVA

class openVA.**OpenVA**(*vaArgs*, *pipelineArgs*, *odkID*, *runDate*)
Assign cause of death (COD) to verbal autopsies (VA) R package openVA.

This class creates and executes an R script that copies (and merges) ODK Briefcase exports, runs openVA to assign CODs, and creates outputs for depositing in the Transfers DB and to a DHIS server.

algorithm [str] Which VA algorithm should be used to assign COD.

copyVA(self) Create data file for openVA by merging ODK export files.

rScript(self) Wrapper for algorithm-specific methods that create an R script to use openVA to assign CODs.

__rScript_InSilicoVA(self)__ Write an R script for running InSilicoVA and assigning CODs.

__rScript_InterVA(self)__ Write an R script for running InterVA and assigning CODs.

getCOD(self) Run R as subprocess and run the script for assigning CODs.

OpenVAError

copyVA()
Create data file for openVA by merging ODK export files.

getCOD()
Create and execute R script to assign a COD with openVA; or call the SmartVA CLI to assign COD.

rScript()
Create an R script for running openVA and assigning CODs.

smartVA_to_csv()

Write two CSV files: (1) Entity Value Attribute blob pushed to DHIS2 (entityAttributeValue.csv

2. table for transfer database (recordStorage.csv)

Both CSV files are stored in the OpenVA folder.

4.5 API for DHIS2

class dhis.**DHIS**(*dhisArgs*, *workingDirectory*)
Class for transfering VA records (with assigned CODs) to the DHIf2S server.

This class includes methods for importing VA results (i.e. assigned causes of death from openVA or SmartVA) as CSV files, connecting to a DHIS2 server with the Verbal Autopsy Program, and posting the results to the DHIS2 server and/or the local Transfer database.

dhisArgs [(named) tuple] Contains parameter values for connected to DHIS2, as returned by transferDB.configDHIS().

connect(self) Wrapper for algorithm-specific methods that create an R script to use openVA to assign CODs.

postVA(self) Prepare and post VA objects to the DHIS2 server.

verifyPost(self) Verify that VA records were posted to DHIS2 server.

CheckDuplicates(self) Checks the DHIS2 server for duplicate records.

DHISError

connect()
Run get method to retrieve VA program ID and Org Unit.

postVA(*apiDHIS*)
Post VA records to DHIS.

verifyPost(*postLog*, *apiDHIS*)
Verify that VA records were posted to DHIS2 server.

class dhis.**API**(*dhisURL*, *dhisUser*, *dhisPass*)
This class provides methods for interacting with the DHIS2 API.

dhisURL [str] Web address for DHIS2 server (e.g., "play.dhis2.org/demo").

dhisUser [str] Username for DHIS2 account.

dhisPassword [str] Password for DHIS2 account.

get(self, endpoint, params) GET method for DHIS2 API.

post(self, endpoint, data) POST method for DIHS2 API.

post_blob(self, f) Post file to DHIS2 and return created UID for that file.

DHISError

get(*endpoint*, *params=None*)
GET method for DHIS2 API. :rtype: dict

post(*endpoint*, *data*)
POST method for DHIS2 API. :rtype: dict

post_blob(*f*)
Post file to DHIS2 and return created UID for that file :rtype: str

class dhis.**VerbalAutopsyEvent**(*va_id*, *program*, *dhis_orgunit*, *event_date*, *sex*, *dob*, *age*, *cod_code*, *algorithm_metadata*, *odk_id*, *file_id*)
DHIS2 event + a BLOB file resource

format_to_dhis2(*dhisUser*)
Format object to DHIS2 compatible event for DHIS2 API :rtype: dict

dhis.**create_db**(*fName*, *evaList*)
Create a SQLite database with VA data + COD :rtype: None

dhis.**getCODCode**(*myDict*, *searchFor*)
Return COD label expected by DHIS2. :rtype: str

dhis.**findKeyValue**(*key*, *d*)
Return a key's value in a nested dictionary.

4.6 Excetptions

exception transferDB.**PipelineError**
Base class for exceptions in the openva_pipeline module.

exception transferDB.**DatabaseConnectionError**

exception transferDB.**ODKConfigurationError**

exception transferDB.**OpenVAConfigurationError**

exception transferDB.**DHISConfigurationError**

exception odk.**ODKError**

exception openVA.**OpenVAError**

exception openVA.**SmartVAError**

CHAPTER

FIVE

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

INDEX

S

T

U

V