Worked Examples using Python openVA

Python openVA Workshop

Dar es Salaam, Tanzania

Day 2 (August 9th): morning session

Agenda – Workflow with Python openVA

- Loading data
- Data conversion (ODK -> openVA) with pyCrossVA
- Data consistency checks
- Running algorithms
  - InSilicoVA
  - InterVA
- Viewing and saving results

Loading Data

- Data are loaded as a comma-separated value (CSV) file
  - Expected format: 2016 WHO VA Instrument (versions 1.5.1 - 1.5.3)
  - (app will be updated when algorithms are ready for 2022 instrument)
- Changing the WHO format
  - It is fine to ADD new questions/items to the form
  - REMOVING questions/items may cause problems
  - Similarly with changing the names (e.g., Id10004)

Data Conversion with pyCrossVA

- ODK exports of the 2016 WHO VA instrument contains over 500 columns
  - Only a subset of these are used to create new variables (y/n/. or yes/no/missing) used by the algorithms
- pyCrossVA is the tool used to convert the original VA data to the format expected by openVA
  - It uses ID names defined in the 2016 WHO VA instrument to find the variables it needs
  - If a necessary column cannot be found, pyCrossVA will assume it is missing and will not be able to create the openVA variables that depend on that column
- Python openVA GUI will provide you with the messages from pyCrossVA
  - In Customizable mode, you can also save the converted data set as a CSV file

Data Consistency Checks

- With the development of the InterVA algorithm, several data consistency checks were designed to ensure that indicators and symptoms do not indicate conflicting information (e.g., male and pregnant).
- Example inconsistency
  - ageInDays: 10 days
  - How long did (s)he have a cough: 4 weeks
- The data consistency checks try to correct these issues
  - (does not make changes to you original data file; make a copy and only changes this copy)

Data Consistency Checks

- The data checks are defined in InterVA's probbase.xls
    - the symptom-cause-information matrix with the conditional probabilities of each symptom given a cause
- Each type of consistency check is performed for each VA record, and then the process is repeated a second time.
- Before the consistency checks are run, **any VA record with missing information on age or sex are removed** -- these indicators are necessary for running the InterVA and InSilicoVA algorithms.

Data Consistency Checks

- Let’s take a look at probbase.xls
- Relevant columns (that we will refer to when describing the checks)
  - indic (column A)
  - subst (column F)
  - dontask1 - dontask8 (columns H - O)
  - doaskif (column P)
  - nnonly (column Q)

Data Consistency Checks

There are 3 types of data consistency checks:

(1) Don't Ask – if the answer to Question A is YES, then do not ask Question B

(2) Ask If – if the answer to Question A is YES

(3) Neonate Only – certain questions are only asked if the decedent was a neonate

Data Consistency Checks: Don't Ask

- **Necessary Conditions** (for an inconsistency to exist)
  - both symptoms have non-missing values
  - index symptom == (Y or N) value in `subst` probbase column
  - symptom in the `dontaskX` probbase column == last character (Y or N) in that cell (`dontaskX` ranges from `dontask1` to `dontask8`
- **Action**: index symptom is set to missing
- **Log Message**
  - "(don't ask symptom) cleared in working information"

Data Consistency Checks: Don't Ask Example

- **Log message**:
  - "4 W610059-o (married) value inconsistent with W610022-a (65+) - cleared in working information"
- VA record ID is 4
- `i059o` - Was she married at the time of death? (with `subst == Y`)
- `i022a` - Was s(he) aged 65 years or more at death?
- `dontask6 - i022aY` (don't ask item for index symptom `i059o` if `i022a == Y`)
- **Action**:
  - `i059a` is changed (in the working copy of the data) from Y to missing

Data Consistency Checks: Ask If

- **Necessary Conditions**
  - the index symptom == (Y or N) value in the `subst` probbase column (and thus not a missing value)
  - the symptom listed in the `doaskif` probbase column *does not* equal the last character in that cell
- **Action**: assign the symptom listed in the `doaskif` to the last character (Y or N) in that cell (concatenated to the symptom label, e.g., `i022cY`)
- **LogMessage**
  - "(ask if symptom label) updated in working information"

Data Consistency Checks: Ask if Example

- **Log Message**
  - "7 W610152-o (fev nsw) not flagged in category W610147-o (fever) - updated in working information"
- VA record ID is 7
- `i152o` - Did (s)he have night sweats? (with `subst == Y`)
- `i147o` - During the illness that led to death, did (s)he have a fever?
- `doaskif` - do ask `i152o` if `i147o == Y`
- **Action**
  - `i147o` is changed from missing to Y

Data Consistency Checks: Neonate Only

- **Necessary Conditions**
  - index symptom == (Y or N) value in the `subst` probbase column (and thus index symptom does not have a missing value)
  - the decedent was NOT a neonate
- **Action**: assign the index symptom to missing
- **Log Message**:
  - "(index symptom) only required for neonates - cleared in working information"

Data Consistency Checks: Neonate only Example

- **Log Message**
  - "103075 W610394-a (born 1st pr) only required for neonates - cleared in working information"
- VA record ID is 103075
- `i394a` - Was this baby born from the mother's first pregnancy?
- VA record was not a neonatal death
- **Action**
  - `i394a` is set to missing

Running Algorithms & Accessing Results

Let’s go to the app…