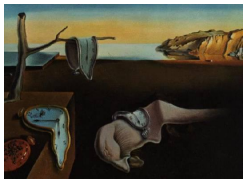

Journal of Statistical Software

MMMMMM YYYY, Volume VV, Issue II. doi: 10.18637/jss.v000.i00

The openVA toolkit for Verbal Autopsies

Zehang Richard Li
University of Washington

Jason Thomas
The Ohio State University

Tyler H. McCormick
University of Washington

Samuel J. Clark
The Ohio State University

Abstract

Verbal autopsy (VA) is a survey-based tool widely used to infer cause of death (COD) in regions without complete-coverage civil registration and vital statistics systems. In such settings, many deaths happen outside of medical facilities and are not officially documented by a medical professional. VA surveys, consisting of signs and symptoms reported by a person close to the decedent, are used to infer the cause of death for an individual, and to estimate and monitor the cause of death distribution in the population. There are three components required for analyzing VAs: (i) VA survey data, (ii) inputs that give information about the association between symptoms and causes, and (iii) a statistical or algorithmic method for assigning a likely cause. For each of these pieces, there are several variants produced independently by research and policy organizations. Incompatibility between components means that there is no systematic way of applying and comparing different methods without laborious, idiosyncratic data conversion. The **openVA** package aims to simplify comparison across existing VA tools through a standardized pipeline that is compatible with all openly available methods and data structure. It provides an open-sourced, R implementation of four most widely used VA methods: InterVA, InSilicoVA, Naive Bayes Classifier, and Tariff. It supports different data input and output formats, and customizable information about the associations between causes and symptoms. The paper discusses the relevant algorithms, the implementations in R and some additional information on advanced model fitting with InSilicoVA.

Keywords: verbal autopsy, cause of death, demography, reproducible research, R.

1. Introduction

Verbal autopsy (VA) is a well established approach to ascertain cause-of-death when medical certification and full autopsies are not feasible or practical (Garenne 2014; Taylor, Parker, Reinke, Faruquee *et al.* 1983). After a death is identified, a specially-trained fieldworker interviews the caregivers (usually family members) of the decedent. A typical VA interview includes a set of structured questions with categorical or quantitative responses and a narrative section that records the 'story' of the death from the respondent's point of view (World Health Orga-

nization and others 2012). Currently, there are multiple commonly used questionnaires with overlapping, but not identical questions.

After VA data are collected, inferring a cause has two additional components. First, there must be some external information about the relationship between causes and symptoms. One means of obtaining this information is directly through expert opinion. A common practice is to ask groups of physicians to rate the likelihood of a symptoms occurring given a particular cause of death, which can be converted into a set of probabilities of observing a symptom given a particular cause. Alternatively, external information can take the form of a small training set of cases with cause of death assigned by in person autopsy. This process is extremely time and resource-intensive, however, and requires assumptions about the generalizability of deaths in the training set to the population of interest. A more common practice is to obtain training data using clinically trained, experienced physicians read a fraction of the interviews and determine causes. To address the fact that physicians frequently do not agree on causes, VA interviews are often read by two physicians, and sometimes three, and the final causes are determined through a consensus mechanism (e.g. Kahn, Collinson, Gómez-Olivé, Mokoena, Twine, Mee, Afolabi, Clark, Kabudula, Khosa *et al.* 2012). Second, actually assigning a cause of death requires an algorithmic or statistical method that combines the external information on the cause-symptom relationship with the symptoms observed in the VA interviews.

In all, therefore, there are three components required to analyze VA data: (i) VA survey data, (ii) inputs that give information about the association between symptoms and causes, and (iii) a statistical or algorithmic method for assigning a likely cause. The current state of VA literature does not distinguish between these three, in part because existing software for algorithmic and statistical methods require a specific set of inputs and survey format. This restriction prevents robust comparison between methods and contexts. A health agency in one region may, for example, want to analyze VA data using the same VA algorithm as a neighboring region to ensure estimates are comparable. Unless the agencies used the same survey format, however, this is not possible with existing tools. The **openVA** package (Li, McCormick, and Clark 2019) addresses this issue through an open-source toolkit that (i) performs data processing and conversion between existing survey formats, (ii) implements multiple currently available algorithmic and statistical methods, and (iii) provides visualization and tools for interpreting results.

1.1. The openVA package

openVA comprises a suite of R (R Core Team 2014) packages for the analysis of verbal autopsy data. The goal of this package is to provide researchers with an open-sourced tool that supports flexible data input format and allows easier data manipulation and model fitting. The core packages in the **openVA** family are **InterVA4**, **InterVA5**, **nbc4va**, **InSilicoVA**, and **Tariff**, each of which implements one coding algorithm. Three survey formats are considered in this paper: the WHO 2012 and WHO 2016 instrument after standard dichotomization procedures into binary variables (World Health Organization and others 2012; Nichols, Byass, Chandramohan, Clark, Flaxman, Jakob, Leitao, Maire, Rao, Riley, Setel, and Grp 2018), and the IHME questionnaire in the format of the Population Health Medical Research Consortium (PHMRC) dataset (Murray, Lopez, Black, Ahuja, Ali, Baqui, Dandona, Dantzer, Das, Dhingra *et al.* 2011a). We demonstrate the analysis with all four algorithms using datasets in both formats.

The main focus of this paper is to provide a general introduction to the implementation details of the included algorithms both in terms of the underlying methodology, and through a series of case studies. For each of the four algorithms discussed, there is a standalone R package available on CRAN, and three of them, InterVA-4, InterVA-5 and Tariff, are also available in

compiled software program distributed by the original authors. The **openVA** package has four major contributions:

1. It provides a clean, standard, and easy interface for analysts to fit and evaluate each method on different types of data input using the same syntax. Previously, most of the methods are designed to be used specifically with their own input formats and are usually incompatible with others. The **openVA** closes this gap and allows easier and fair model comparison of multiple algorithms on the same data.

2. It provides a series of functionalities to summarize and visualize results from multiple algorithms, which is helpful for analysts not familiar with data manipulation in R.

3. It does not directly implement any algorithms for coding VA data[1], so that it is possible for a research group to maintain their own algorithm implementations callable from the **openVA** package, while also make it available to the general users as a standalone piece of software. For example, the **nbc4va** was developed and maintained independently by researchers at the Center for Global Health Research in Toronto, but is designed so that it can be seamlessly integrated into the **openVA** package.

4. It is fully open-sourced, and can be run on multiple platforms. Compared to the alternative implementations, InterVA-4 and InterVA-5 software provide the source codes as an additional file and they are difficult to modify and re-compile. Tariff, as implemented through the SmartVA-analyze (Serina, Riley, Stewart, James, Flaxman, Lozano, Hernandez, Mooney, Luning, Black *et al.* 2015b) is closed-sourced. All of these software can only be run on Windows systems[2]. The open-sourced nature of **openVA** significantly expands its potential for methodological research and its suitability for integration within a larger data analysis pipeline.

The rest of this paper is organized as follows: In Section 2 we briefly introduce the main component packages and the underlying algorithms. We demonstrate model fitting with the **openVA** package for different input data format in Section 3 to 5. We first show in Section 3 how to prepare and transform VA data in different formats. We then present the basic model fitting and plotting syntax in Section 4, and syntax for summarizing results in Section 5. Then in Section 6 we discuss the implementation of InSilicoVA algorithm in more detail and provide several examples for using InSilicoVA to address more complex research questions using VA data. We end in Section 7 with a discussion of remaining issues and limitations of the existing automated VA algorithms and proposes new functionalities to be included in **openVA** package.

2. Structure of the openVA package

The **openVA** suite currently consist of five standalone packages that are available on the CRAN for fitting different methods. In this section, we first provide a brief introduction to these four packages, and we discuss the mathematical details behind each algorithm in the next subsection. The versions of these packages can be checked in R using

```
R> library(openVA)
R> library(nbc4va)
R> openVA_status()
```

[1]A special case is when we extend the InterVA-4 algorithm to the scenario where symptoms and causes are not in the pre-defined set provided by the original InterVA-4 software, the extension is included in the **openVA** package instead of the **InterVA4** package for a faithful replication of the InterVA methods themselves.

[2]In April, 2018, a compiled library of SmartVA-analyze on Linux system was made available on Github as part of a related project, but the source codes remained unavailable and no instruction was provided as how researchers can use the library as an alternative to the Windows-based software.

- **InterVA4** (Li, McCormick, and Clark 2014; Li, McCormick, Clark, and Byass 2018c) is an R package that implements the InterVA-4 model (Byass, Chandramohan, Clark, D'Ambruoso, Fottrell, Graham, Herbst, Hodgson, Hounton, Kahn *et al.* 2012). It provides replication of both InterVA software version 4.02 and the later release of version 4.03 update (Byass 2015). The standard input of **InterVA4** is in the form of a pre-defined set of indicators, based on the 2012 WHO VA instrument (see `http://www.who.int/healthinfo/statistics/verbalautopsystandards/`). The default InterVA-4 algorithm cannot be applied to other data input format because its internal built-in prior information is specific to a fixed set of indicators and causes. The same restriction is also maintained in **InterVA4** package. However, the mathematical formulation of InterVA-4 model is completely generalizable to other binary input format. The generalized algorithm is described in Section 2.1, and also implemented in the **openVA** package.

- **InterVA5** (Thomas, Li, McCormick, Clark, and Byass 2018) is an R package that implements the InterVA-5 model developed by Peter Byass (Byass 2018, , not yet publicly released at the time of writing). The InterVA-5 model updates the previous version in several ways. First, the input data must adhere to the format of the 2016 WHO VA instrument (Nichols *et al.* 2018). Second, changes have been made to the data processing steps, which are described in Section 6.2. It is also worth noting that the model outputs have expanded by the inclusion of the most likely Circumstances Of Mortality CATegory, or COMCAT, among the results – the categories include: culture, emergency, health systems, inevitable, knowledge, resources, or an indeterminant combination of multiple factors (for more details, see D'Ambruoso, Boerma, Byass, Fottrell, Herbst, Kallander, and Mullan 2017). Despite these changes, the mathematical formulation of InterVA-5 is identical to that of InterVA-4.

- **nbc4va** (Miasnikof, Giannakeas, Gomes, Aleksandrowicz, Shestopaloff, Alam, Tollman, Samarikhalaj, and Jha 2015; Wen, Miasnikof, Giannakeas, and Gomes 2018) is an R package that implements the Naive Bayes Classifier for VA encoding. The mathematical formulation is similar to that of InterVA-4, but accounts for absence of symptoms while it is ignored in InterVA-4. It also calculates the conditional probabilities of symptoms given causes of death from training dataset instead of using physician provided values. Unlike the other three algorithms, **nbc4va** is developed and maintained by Prabat Jha's research group in Toronto but is designed so that it can be seamlessly integrated into **openVA**.

- **InSilicoVA** (Li, McCormick, and Clark 2018a) is an R package that implements the InSilicoVA algorithm, a Bayesian hierarchical framework for cause-of-death assignment and cause-specific mortality fraction estimation proposed in McCormick, Li, Calvert, Crampin, Kahn, and Clark (2016). It is originally designed to work with the 2012 WHO VA instrument, i.e., the same input data as in InterVA-4 software, but is also generalizable to other data input format. It is a fully probabilistic algorithm and could incorporate multiple sources of information, such as known sub-population in the dataset, and partial or complete physician coding. The internal MCMC sampler is implemented in Java for improved speed.

- **Tariff** (Li, McCormick, and Clark 2018b) is an R package that implements the Tariff algorithm (James, Flaxman, Murray, and Consortium Population Health Metrics Research 2011). It most closely reflects the description of Tariff 2.0 method (Serina *et al.* 2015b). The Tariff algorithm is developed by the Institute for Health Metrics and Evaluation (IHME) and officially implemented in the SmartVA-Analyze software. However, as the developers of this R package are not affiliated with the authors of the original algorithm,

there are some discrepancies in implementation. The SmartVA-Analyze software uses data collected from the PHMRC Full or Shortened Questionnaire with the Open Data Kit (ODK) Collect system Serina, Riley, Stewart, Flaxman, Lozano, Mooney, Luning, Hernandez, Black, Ahuja *et al.* (2015a). The source code of both SmartVA-Analyze and the two versions of Tariff are not publicly available. Thus the **Tariff** package was developed based solely on the descriptions in the published work. Despite the difference in implementation, **Tariff** is able to achieve comparable results as the published work as demonstrated in McCormick *et al.* (2016). More detailed descriptions of the **Tariff** implementations are also discussed in the supplement of McCormick *et al.* (2016).

2.1. Overview of VA algorithmic and statistical methods

Typically, VA surveys are first converted into a series of binary responses to questions about each death, and then the two main goals of a typical VA analysis are to estimate

1. the population cause-specific mortality fractions (CSMF),
2. probability distribution or rankings of cause-of-death (COD) for each individual death.

In this section, we formally compare the modeling approaches utilized by each algorithm. We adopt the following notations: Let there be N deaths, each with S binary indicators of symptoms. s_{ij} denotes the indicator for presence of j-th symptom in the i-th death, which can take values from 0, 1, or NA (for missing data). The pre-defined set of causes is of size C. For the population, the CSMF of cause k is denoted as $\pi(C_k)$, and for the i-th death, the probability of dying from cause j is denoted as P_{ik}. The COD assignment for the i-th death is denoted as y_i.

- **InterVA4** (Byass *et al.* 2012) and **InterVA5** (Byass 2018) algorithm calculate the probability of each COD given the observed symptoms using the Bayes rule, so that

$$P_{ik} = \frac{p_0(C_k) \prod_{j=1}^{S} P(s_{ij} = 1 | y_i = k) \mathbf{1}_{s_{ij}=1}}{\sum_{k'=1}^{C} p_0(C_{k'}) \prod_{j=1}^{S} P(s_{ij} = 1 | y_i = k') \mathbf{1}_{s_{ij}=1}}$$

  where both the prior distribution of each causes, $p_0(C_k)$, and the conditional probability of each symptom given each cause, $P(s_{ij} = 1 | y_i = k)$, are provided internally in the algorithm for each of the default cause-symptom combinations. The standard InterVA software only supports the fixed set of symptoms and causes where such prior information is provided. For a different data input format, it is straightforward to generalize this formulation with training data. We extend the algorithm by calculating $\hat{P}(s_{ij} = 1 | y_i = k)$ from the empirical probability of observing symptom j in deaths labeled as from cause k. In practice, $P(s_{ij} = 1 | y_i = k)$ used in InterVA-4 algorithms are represented as rankings with letter grades instead of numerical value. For example, $P(s_{ij} = 1 | y_i = k) = A+$ is translated into $P(s_{ij} = 1 | y_i = k) = 0.8$, etc. Byass *et al.* (2012). Thus we also map $\hat{P}(s_{ij} = 1 | y_i = k)$ to a similar letter-value correspondence table in Byass *et al.* (2012) by truncating the raw empirical values. The details of such truncations can be found in McCormick *et al.* (2016).

  After the individual COD distributions are calculated, InterVA-4 utilizes a series of predefined rules to decide up to top three most likely COD assignments and truncates the probabilities for the rest of the CODs to 0 and adds an "undetermined" category so that the probabilities sum up to 1 (See the user guide of Byass (2015)). Then the population-level CSMFs are calculated as the aggregation of individual COD distribution, such that

$$\pi_k = \sum_{i=1}^{N} P_{ik}^*$$

where P^*_{ik} denotes the COD distribution after the modification by truncation. InterVA-4.03 fixes two major bugs introduced in InterVA-4.02, and changed some data checking rules that were previously ignored in InterVA-4.02.

- **Naive Bayes Classifier** (Miasnikof *et al.* 2015) is very similar to the InterVA algorithm with two major differences. First, instead of considering only symptoms that present, NBC algorithm also considers symptoms that are absent. Second, the conditional probabilities of symptoms given causes are calculated from training data instead of given by physicians, which is similar to our extension of InterVA-4 discussed above. Similar to InterVA-4, the NBC method can be written as

$$P_{ik} = \frac{p_0(C_k)\prod_{j=1}^{S}(P(s_{ij}=1|y_i=k)\mathbf{1}_{s_{ij}=1} + P(s_{ij}\neq 1|y_i=k)\mathbf{1}_{s_{ij}\neq 1})}{\sum_{k'=1}^{C} p_0(C_{k'})\prod_{j=1}^{S}(P(s_{ij}=1|y_i=k')\mathbf{1}_{s_{ij}=1} + P(s_{ij}\neq 1|y_i=k')\mathbf{1}_{s_{ij}\neq 1})}$$

- **InSilicoVA** algorithm (McCormick *et al.* 2016) assumes a generative model that characterizes both CSMF at the population level, and the COD distributions at the individual level. In short, the core generative process assumes

$$\begin{aligned} s_{ij}|y_i=k &\propto \text{Bernoulli}(P(s_{ij}|y_i=k)) \\ y_i|\pi_1,...,\pi_C &\propto \text{Multinomial}(\pi_1,...,\pi_C) \\ \pi_k &= \exp\theta_k / \sum_{k=1}^{C}\exp\theta_k \\ \theta_k &\propto \text{Normal}(\mu,\sigma^2) \end{aligned}$$

and additional hyper priors are also placed on $P(s_{ij}|y_i=k)$, μ, and σ^2. The priors for $P(s_{ij}|y_i=k)$ are set by the rankings used in InterVA-4 if the data is prepared into InterVA format, or learned from training data if otherwise. Parameter estimation is performed using Markov Chain Monte Carlo (MCMC), so that a sample of posterior distribution of π_k can be obtained after the sampler converges.

- **Tariff** algorithm (James *et al.* 2011) differs from all other three methods in that it does not calculate an explicit probability distribution of COD for each death. Instead, for each death i, a Tariff score is calculated for each COD k so that

$$Score_{ik} = \sum_{j=1}^{S} \text{Tariff}_{kj}\mathbf{1}_{s_{ij}=1}$$

where the symptom-specific Tariff score Tariff_{kj} is defined as

$$\text{Tariff}_{kj} = \frac{n_{kj} - median(n_{1j}, n_{2j}, ..., n_{Cj})}{IQR(n_{1j}, n_{2j}, ..., n_{Cj})}$$

where n_{kj} is the count of how many deaths from cause k contain symptom j in the training data. The scores calculated are then turned into rankings by comparing to a reference distribution of scores calculated from re-sampling the training dataset to have a uniform COD distribution. It is worth noting that the Tariff algorithm produces the COD distribution for each death in terms of their rankings instead of the probability distributions. And thus the CSMF for each cause k is calculated by the fraction of deaths with cause k being the highest ranked cause, i.e.,

$$\pi_k = \frac{\sum_{i=1}^{N}\mathbf{1}_{y_i=k}}{N}$$

In addition to the different model specifications underlying each algorithm, two major distinctions are most significant in understanding the four methods. First, the missing indicators are assumed to be equivalent to "absence" in InterVA, NBC, and Tariff, but strictly as "missing" in InSilicoVA. Second, the CSMFs are calculated in three different ways. Tariff calculates CSMF as the proportion of the most likely COD assignments in the dataset. InterVA-4 calculates CSMF as the aggregated distribution of up to the top three most likely CODs. And InSilicoVA estimates CSMF directly from the parametric model using MCMC. Some further feature comparisons are summarized in Table 1.

Feature	InterVA	Tariff	NBC	InSilicoVA
Exact replication in current openVA	Yes	No	Yes	Yes
Implementable without training dataset	Yes	No	No	Yes
Can produce instantaneous results for single death	Yes	Yes	Yes	No
Only significant symptoms are used at individual level	No	Yes	No	No
Accounts for absence of symptoms	No	No	Yes	Yes
Accounts for missing symptoms	No	No	No	Yes
Provides individual COD distribution	Yes	No	Yes	Yes
Direct estimation of CSMF and its uncertainty	No	No	No	Yes

Table 1: Comparison of the four VA methods in their features.

3. Data Preparation

In the **openVA** package, we consider two main forms of questionnaire: the WHO instrument and the IHME questionnaire for SmartVA-Analyze. For a complete data pipeline, we need to (1) process raw verbal autopsy records collected to input for different coding algorithms, (2) standardize the input format for each algorithm, and (3) convert the transformed data between different formats. The first task, translating the raw data collected from the Open Data Toolkit to the symptom lists used in InterVA-4 and Tariff 2.0 can be achieved with additional packages, such as **crossVA** (Choi, Clark, Li, Maire, and Thomas 2019). Thus in this section, we focus the discussion on the latter two tasks.

3.1. Standardizing format: WHO

For users familiar with InterVA-4 software and the associated data processing steps, the standard input format from the WHO 2012 instrument is usually well understood: the input data is in a matrix form where each row represents one death, and contains 246 columns, starting from the first item being the ID of the death. The 245 items following the ID each represent one binary variable of symptom/indicator, where "presence" is coded by "Y", and "absence" is coded by an empty cell. The details of this format, as well as the translation from WHO 2012 instrument, could be found at `http://www.who.int/healthinfo/statistics/verbalautopsystandards/en/index2.html`.

To accommodate updates for the WHO 2016 instrument (Nichols *et al.* 2018), InterVA-5 software accepts a data input matrix with 354 columns that include 353 columns of symptom/indicators followed by an additional column for the record ID. It should be noted that the R package **InterVA5** retains the format with the record ID residing in the first column. Another important update with InterVA-5 is that it acknowledge the difference between both "Yes" and "No" (or "Y/y" and "N/n", which is different from the coding scheme in InterVA-4) are processed as relevant responses, while all other responses are treated as missing values and

ignored. With respect to mode outputs, InterVA-5 utilizes the WHO 2016 COD categories, which is nearly identical to the WHO 2012 COD categories (used by InterVA-4) except that haemorrhagic fever and dengue fever are two separate categories in 2016.

The only difference we highlight is that InSilicoVA distinguishes "missing" from "absent" in the input. This is conceptually easy to understand: knowing that a symptom does not exist provides some information to the possible cause assignment, while a missing symptom does not. Missing data could arise from different stages of the data collection process. Although in theory, most of the VA algorithms could benefit from distinguishing "missing" from "absent", InSilicoVA is the only algorithm that has implemented with missing data. We highly recommend users to pre-process all the raw data so that a "missing" value in the data spreadsheet is coded by a "." (following the stata practice), and an "absent" is coded by a empty cell "", as in the standard InterVA-4 software. For WHO 2016 data, both "." and "-" (following the default coding scheme of InterVA-5 software) are interpreted as missing values. For methods other than InSilicoVA, the "missing" and "absent" will be considered the same internally and thus will not introduce compatibility problem.

3.2. Standardizing format: PHMRC

For studies that need to compare with existing work using PHMRC gold standard dataset (Murray *et al.* 2011a), a practical challenge is that although the data is publicly accessible, the pre-processing steps described from the relevant publications are not clear enough nor easy to implement. The **openVA** package provides functions to clean up the PHMRC gold standard data in the best way we could replicate from literature.

First, we allow users to download part or all of the PHMRC gold standard data directly from its on-line repository:

```
R> PHMRC_first1000 <- read.csv(getPHMRC_url("adult"), nrows = 1000)
```

The data can then be cleaned up into a set of dichotomized symptoms, as described in Section 2 of the supplement material of McCormick *et al.* (2016). This dichotomization process was originally described in Murray *et al.* (2011a). However, following the procedures described in the original paper, we are not able to reproduce the thresholds provided with the on-line supplement materials. Therefore we have included both the thresholds used in Murray *et al.* (2011a) and the thresholds calculated as in their description based on the data that user inputs. See the following codes for the summary of the two different transformation results.

```
R> convert.default <- ConvertData.phmrc(PHMRC_first1000, phmrc.type = "adult",
+                                       cutoff = "default", cause = "va34")

The first column is site, assign IDs to each row by default
1000 deaths in input. Format: adult
168 binary symptoms generated


Number of Yes        21023
Number of No         124644
Number of Not known  22333
```

```
R> convert.adapt <- ConvertData.phmrc(PHMRC_first1000, phmrc.type = "adult",
+                                     cutoff = "adapt", cause = "va34")

The first column is site, assign IDs to each row by default
1000 deaths in input. Format: adult
168 binary symptoms generated
```

```
Number of Yes        21711
Number of No         123956
Number of Not known  22333
```

Notice that the original PHMRC data are useful for comparing and validating new methods, as well as for using as training data, but the cleaning functions require only the columns to be exactly as the PHMRC gold standard dataset on-line, so they could also be used for new data that are pre-processed into the same format[3].

3.3. Standardizing format: Other

In practice, researchers might also be interested in using dichotomous data containing a customized set of symptoms. The **openVA** package also supports customized input as long as they are dichotomous. However, since for a customized set of symptoms, neither the built-in conditional probability matrix of InterVA nor the PHMRC gold standard dataset could be used to learn the relationship between training and testing data, some training data with known causes of death is necessary for all three algorithm.

To make data processing step easier for users, it is also possible to convert datasets in slightly different format to the default default format using the `ConvertData` function. For example, the small dataset below used a coding scheme different from the default format of the input. Instead of re-processing the data from its original source, what we need is only to let "Yes" recoded into "Y", "No" into "", and "Don't know" and "Refused to answer" into ".".

```
R> # make up a fake 2 by 3 dataset with 2 deaths and 3 symptoms
R> toyid <- c("d1", "d2")
R> toycause <- c("A", "B")
R> toyData <- matrix(c("Yes", "No", "Don't know",
+                   "Yes", "Refused to answer", "No"),
+                   byrow = TRUE, nrow = 2, ncol = 3)
R> toyData <- cbind(toyid, toycause, toyData)
R> colnames(toyData) <- c("ID", "Cause", "S1", "S2", "S3")

R> toyData

     ID   Cause S1    S2                 S3
[1,] "d1" "A"   "Yes" "No"               "Don't know"
[2,] "d2" "B"   "Yes" "Refused to answer" "No"

R> toyDataNew <- ConvertData(toyData, yesLabel = "Yes", noLabel = "No",
+                     missLabel = c("Don't know", "Refused to answer"))
R> toyDataNew

  ID Cause S1 S2 S3
1 d1     A  Y     .
2 d2     B  Y  .
```

3.4. Convert data between standardized formats

The **openVA** package supports customized set of symptoms to be used as input for all methods, thus it is usually less important to convert data from one format to another, since the conversion

[3]However, since calculating the thresholds adaptively requires the knowledge of underlying cause-of-deaths, the latter approach could not be applied to unlabeled datasets.

inevitably creates loss of information. We recognize the exact mapping of symptoms between different format can be useful for some applications. For example, a full mapping of PHMRC dataset into the InterVA format enables the use of physician provided conditional probabilities included in the InterVA software. This remains as an important feature to be added to the package in the future.

4. Basic Model Fitting

For the purpose of illustrating the model fitting process in the **openVA** package, we consider two datasets: (1) a random sample of 1,000 deaths from ALPHA network without gold standard labels collected with the WHO 2012 instrument, and (2) the adult VA records in the PHMRC gold standard data, with the 1,554 records from Andhra Pradesh, India used as testing set and the rest as training set. The former could be used for InterVA-4 and InSilicoVA and the latter can can be applied to all methods with some minor modification to InterVA-4 and InSilicoVA. We first show the model fitting syntax for each of the models, and then summarize and compare the results, and present some codes for visualization.

The randomly sampled VA records from ALPHA network sites are already included in the openVA package and can be loaded using

```
R> data(RandomVA1)
R> dim(RandomVA1)
```

```
[1] 1000  246
```

```
R> head(RandomVA1[, 1:10])
```

```
  ID elder midage adult child under5 infant neonate male female
1 d1     Y                                                Y
2 d2     Y                                                      Y
3 d3            Y                                         Y
4 d4                  Y                                        Y
5 d5                  Y                                  Y
6 d6                  Y                                        Y
```

For the PHMRC gold standard data, We first read the complete dataset from on-line, and then organize them into training and testing data.

```
R> PHMRC_all <- read.csv(getPHMRC_url("adult"))
R> AP <- which(PHMRC_all$site == "AP")
R> test <- PHMRC_all[AP, ]
R> train <- PHMRC_all[-AP, ]
R> dim(test)
```

```
[1] 1554  946
```

```
R> dim(train)
```

```
[1] 6287  946
```

4.1. ALPHA data: InterVA-4 and InSilicoVA

For the generic implementation of InterVA-4 algorithm, we turn to the 1,000 randomly sampled unlabeled data from ALPHA sites. The model fitting syntax is very similar to before:

```
R> interva1_4_02 <- codeVA(data = RandomVA1, data.type = "WHO2012",
+                          model = "InterVA", version = "4.02")

R> interva1_4_03 <- codeVA(data = RandomVA1, data.type = "WHO2012",
+                          model = "InterVA", version = "4.03")
```

Also similar to before, we can compare the fitted CSMF for the 60 causes for the two implementations.

```
R> csmf_v4_02 <- getCSMF(interva1_4_02)
R> csmf_v4_03 <- getCSMF(interva1_4_03)
R> plot(as.numeric(csmf_v4_02), as.numeric(csmf_v4_03),
+       xlab = "CSMF InterVA4.02",
+       ylab = "CSMF InterVA4.03")
R> abline(a=0,b=1,col = "red")
```

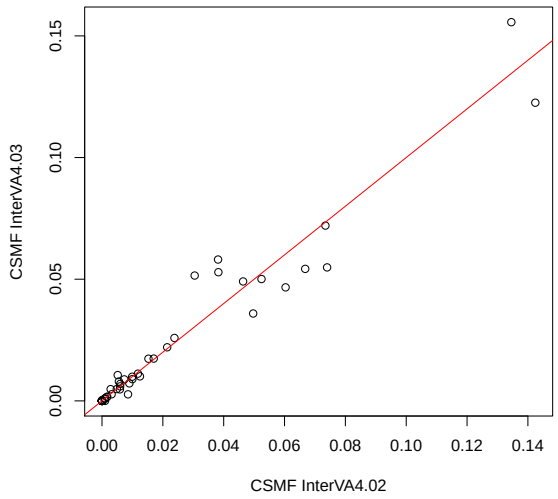

Figure 1: Comparing InterVA-4-4.02 and InterVA-4.03 fitted CSMF on the same dataset

Similarly, we can implement InSilicoVA with the default parameters, and run the MCMC for 10,000 iterations.

```
R> insilico1 <- codeVA(RandomVA1, data.type = "WHO2012", model = "InSilicoVA",
+                       Nsim=10000, auto.length = FALSE)
```

4.2. PHMRC data: all methods

First, to fit the InterVA algorithm with the PHMRC data, as explained in Section 2.1, we could learn the marginal conditional probabilities from the training data, and apply the generalized algorithm. In such case, the `version` is suppressed accordingly. To obtain marginal conditional probabilities, we can map the empirical $P_{s|c}$ matrix to the letter grade system so that the percentile of each rank stays the same as the original $P_{s|c}$ matrix in InterVA-4 (`type = "quantile"`). Alternatively we can also use the original fixed values of translation, and assign letter grades closest to each entry in $\hat{P}_{s|c}$ (`type = "fixed"`). With a training data of sample size n, the smallest non-zero entry in $P_{s|c}$ is $1/n$. This may lead to potential issue that many letter grades in the original InterVA $P_{s|c}$ may not exist if $1/n$ is not small enough. Finally, we

can also directly use the values in the $\hat{P}_{s|c}$ without converting them (`type = "empirical"`). In this paper, we assume the first type of conversion throughout for both InterVA-4 and InSilicoVA.

```
R> interva2 <- codeVA(data = test, data.type = "PHMRC", model = "InterVA",
+                     data.train = train, causes.train = "gs_text34",
+                     phmrc.type = "adult")

R> insilico2 <- codeVA(data = test, data.type = "PHMRC", model = "InSilicoVA",
+                      data.train = train, causes.train = "gs_text34",
+                      phmrc.type = "adult",
+                      jump.scale = 0.05, convert.type = "fixed",
+                      Nsim=10000, auto.length = FALSE)
```

The NBC and Tariff method, on the other hand, does not perform such conversion.

```
R> nbc2 <- codeVA(data = test, data.type = "PHMRC", model = "NBC",
+                     data.train = train, causes.train = "gs_text34",
+                     phmrc.type = "adult")

R> tariff2 <- codeVA(data = test, data.type = "PHMRC", model = "Tariff",
+                    data.train = train, causes.train = "gs_text34",
+                    phmrc.type = "adult")

The first column is site, assign IDs to each row by default
6287 deaths in input. Format: adult
1554 deaths in test input. Format: adult
168 binary symptoms generated

Number of Yes        162469
Number of No         977538
Number of Not known  177281

Start re-sampling for significant Tariff cells
Calculating ranks
```

Finally, we notice that we do not need to transform the data manually. Data transformations are performed automatically within the `codeVA` function. The arguments originally passed into `ConvertData.phmrc` can also be passed into `codeVA`. For example, if we wish to use the adaptive threshold instead of the ones provided int (Murray *et al.* 2011a),

```
R> tariff2_a <- codeVA(data = test, data.type = "PHMRC", model = "Tariff",
+                    data.train = train, causes.train = "gs_text34",
+                    phmrc.type = "adult", cutoff = "adapt")

The first column is site, assign IDs to each row by default
6287 deaths in input. Format: adult
1554 deaths in test input. Format: adult
168 binary symptoms generated

Number of Yes        162934
Number of No         977073
Number of Not known  177281

Start re-sampling for significant Tariff cells
Calculating ranks
```

We can then see the estimated CSMF of the two Tariff model differ slightly:

```
R> csmf_defaut <- getCSMF(tariff2)
R> csmf_adapt <- getCSMF(tariff2_a)
R> plot(as.numeric(csmf_defaut), as.numeric(csmf_adapt),
+       xlab = "CSMF using default cut-off from Murray et al., 2011",
+       ylab = "CSMF using reproduced cut-off")
R> abline(a=0,b=1,col = "red")
```

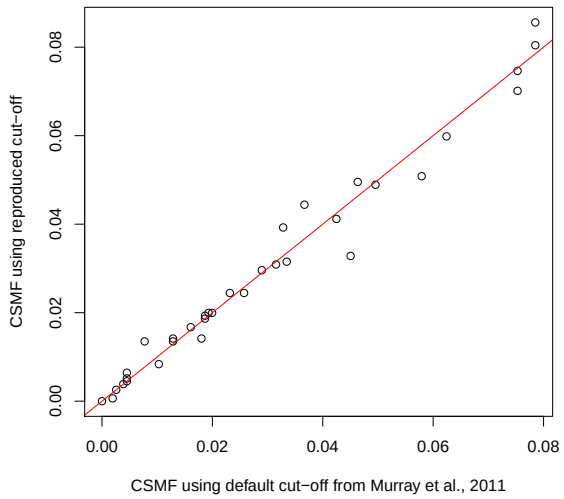

Figure 2: Comparing Tariff fitted CSMF under two different implementations

5. Summarizing results

We have shown the model fitting syntax for all the methods in the previous section. In this section we illustrate how to summarize results and compare them. First, we can see the quick summary of model fitting for each methods using the `summary()` function, e.g.,

```
R> summary(insilico2)
```

In additional to the overall summary of the fitted models. Several other metrics are typically used to evaluate and compare algorithms at both the population and individual levels. In the rest of this section, we show how to easily calculate and visualize some of these metrics with the **openVA** package.

5.1. CSMF Accuracy

Recall that CSMFs are calculated in different ways for different methods. Here we can extract the CSMFs directly using the `getCSMF()` function.

```
R> csmf.tariff2 <- getCSMF(tariff2)
R> csmf.interva2 <- getCSMF(interva2)
R> csmf.nbc2 <- getCSMF(nbc2)
R> csmf.insilico2 <- getCSMF(insilico2)
```

```
R> cbind(Tariff = csmf.tariff2, InterVA = csmf.interva2[1:34],
+        NBC = csmf.nbc2, InSilicoVA = csmf.insilico2[, "Mean"])
```

We can then formally compare the accuracy of the CSMF estimation defined as

$$CSMF_{acc} = 1 - \frac{\sum_i^C CSMF_i - CSMF_i^{(true)}}{2(1 - \min CSMF^{(true)})}$$

```
R> csmf.true <- table(test$gs_text34)
R> csmf.true <- csmf.true[names(csmf.tariff2)]
R> csmf.true <- as.numeric(csmf.true / sum(csmf.true))
R> cod_names <- names(csmf.tariff2)
R> getCSMF_accuracy(csmf.tariff2, csmf.true)

[1] 0.6755

R> getCSMF_accuracy(csmf.interva2, csmf.true, "Undetermined")

[1] 0.595

R> getCSMF_accuracy(csmf.nbc2, csmf.true)

[1] 0.7695

R> getCSMF_accuracy(csmf.insilico2[, "Mean"], csmf.true)

[1] 0.7301
```

In addition, for InSilicoVA, the posterior credible intervals can be obtained for CSMF. We can also get the distribution of the CSMF accuracy by calculating the metrics at each posterior samples:

```
R> csmf_accuarcy_insilico <- getCSMF_accuracy(insilico2, csmf.true)
R> hist(csmf_accuarcy_insilico)
```

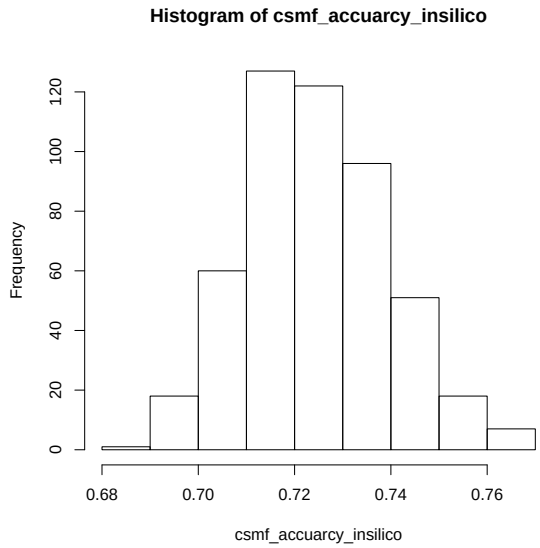

Figure 3: CSMF accuarcy histogram for posterior draws from InSilicoVA model

5.2. Most likely COD assignment

At each individual level, we can extract the most likely cause-of-death assignment from the fitted object using the `getTopCOD()` function.

```
R> cod.tariff2 <- getTopCOD(tariff2)
R> cod.interva2 <- getTopCOD(interva2)
R> cod.nbc2 <- getTopCOD(nbc2)
R> cod.insilico2 <- getTopCOD(insilico2)
```

Similarly we can compare these COD assignments to the true labels

```
R> # install.packages(c("lattice", "gplots"))
R> library(lattice)
R> library(gplots)
R> cod_names <- c(levels(test[, "gs_text34"]), "Undetermined")
R> cod.true <- factor(test[, "gs_text34"], levels = cod_names)
R> cod.all <- list(Tariff = cod.tariff2, InterVA = cod.interva2,
+                  NBC = cod.nbc2, InSilicoVA = cod.insilico2)
R> for(i in 1:length(cod.all)){
+    cod.fit <- factor(cod.all[[i]][,"cause"],
+                levels = cod_names)
+    tab <- table(cod.true, cod.fit)
+    acc <-  round(sum(diag(tab)) / sum(tab), 4) * 100
+    print(
+    levelplot(tab,
+          scales=list(tck=0, x=list(rot=90)),
+          col.regions=colorpanel(11, "white", "grey10"),
+          at=seq(0, 100, len = 11),
+          main=paste0(names(cod.all)[i], " - Accuracy: ", acc, "%"),
+          xlab="True Causes", ylab="Predicted Causes")
+    )
+  }
```

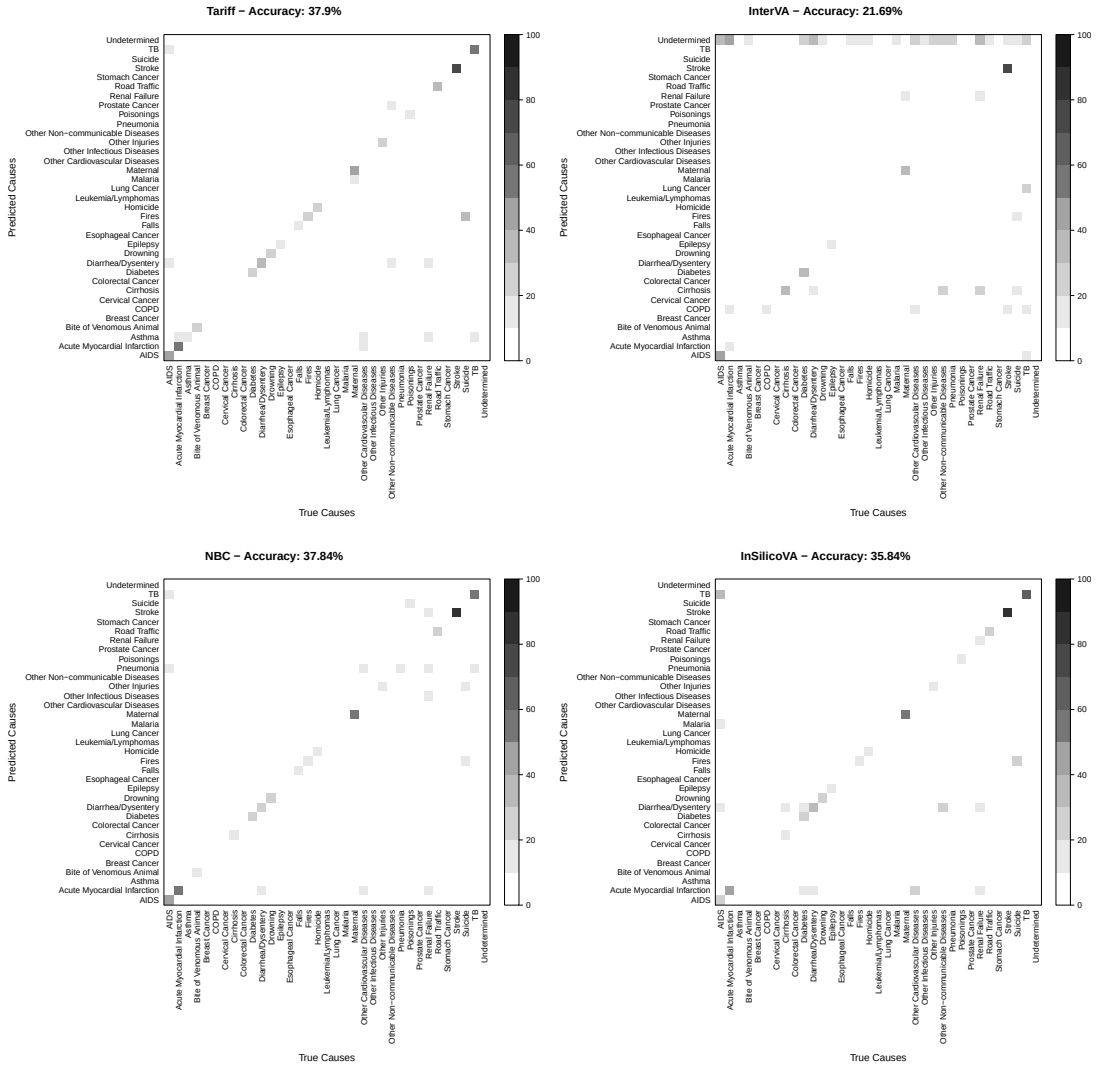

Figure 4: Comparing most likely COD assignments with the truth for the four methods

With the most likely COD assignment, other types of metrics based on individual COD assignment accuracy can be similarly constructed by users. One example is chance-corrected concordance (CCC), which is defined as the following for each COD:

$$CCC_j = \frac{\frac{\#\text{ correctly assigned to cause } j}{\#\text{ total number of death from cause } j} - \frac{1}{C}}{1 - \frac{1}{C}}.$$

The overall CCC is suggested to be the average of all the cause-specific CCC in Murray, Lozano, Flaxman, Vahdatpour, and Lopez (2011b). CCC can be easily calculated from the individual COD assignments, as illustrated in the following code snippet.

```
R> getCCC <- function(fitted, truth){
+     fitted <- as.character(fitted)
+     truth <- as.character(truth)
+     C <- length(unique(truth))
+     cccj <- rep(NA, C)
+     correct <- fitted[which(fitted == truth)]
+     N <- length(truth)
+
```

```
+    for(i in 1:length(unique(truth))){
+      c <- sort(unique(truth))[i]
+      if(length(which(truth == c)) == 0) next
+      cccj[i] <- length(which(correct == c))/length(which(truth == c))
+      cccj[i] <- (cccj[i] - 1/C) / (1 - 1/C)
+    }
+    ccc <- mean(cccj, na.rm = TRUE)
+    return(ccc)
+  }
R> cod.true <- test[, "gs_text34"]
R> ccc.tariff2 <- getCCC(cod.tariff2[,2], cod.true)
R> ccc.interva2 <- getCCC(cod.interva2[,2], cod.true)
R> ccc.nbc2 <- getCCC(cod.nbc2[,2], cod.true)
R> ccc.insilico2 <- getCCC(cod.insilico2[,2], cod.true)
R> c(ccc.tariff2, ccc.interva2, ccc.nbc2, ccc.insilico2)

[1] 0.3560 0.1513 0.3223 0.3059
```

5.3. Individual COD distribution

The **openVA** also provides summary methods for each death ID. For example, using Tariff method, we can extract the fitted rankings of causes for the death with ID 6288 by

```
R> summary(tariff2, id = "6288")

Tariff fitted top 5 causes for death ID: 6288


 Rank Cause
 1    Other Infectious Diseases
 2    Pneumonia
 3    COPD
 4    Asthma
 5    Epilepsy
```

Notice that the direct call of summary for InSilcoVA does not provide uncertainty measures,

```
R> summary(insilico2, id = "6288")

InSilicoVA fitted top  causes for death ID: 6288
Credible intervals shown: %
                               Mean Lower Median Upper
Pneumonia                 0.4693570    NA     NA    NA
Stroke                    0.4561976    NA     NA    NA
Other Infectious Diseases 0.0576625    NA     NA    NA
Epilepsy                  0.0090355    NA     NA    NA
COPD                      0.0052431    NA     NA    NA
Malaria                   0.0006562    NA     NA    NA
Diabetes                  0.0004767    NA     NA    NA
Acute Myocardial Infarction 0.0003786    NA     NA    NA
Falls                     0.0002990    NA     NA    NA
Renal Failure             0.0002927    NA     NA    NA
```

This is because the calculation of individual posterior probabilities of COD distribution is relatively time-consuming and memory-intense. To obtain individual-level uncertainty measure-

ment, we can either run the chain with the additional argument `indiv.CI = 0.95`,

```
R> insilico2 <- codeVA(data = test, data.type = "PHMRC", model = "InSilicoVA",
+                       data.train = train, causes.train = "gs_text34",
+                       phmrc.type = "adult",
+                       jump.scale = 0.05, convert.type = "fixed", indiv.CI = 0.95,
+                       Nsim=10000, auto.length = FALSE)
```

or often more practically, update the fitted object directly without re-running the chain,

```
R> insilico2 <- updateIndiv(insilico2, CI = 0.95)
```

And then the summary method will provide the credible intervals

```
R> summary(insilico2, id = "6288")
```

```
InSilicoVA fitted top  causes for death ID: 6288
Credible intervals shown: 95%
                                 Mean     Lower    Median     Upper
Pneumonia                   0.4693570 0.3488866 0.4645100 0.5883497
Stroke                      0.4561976 0.3448711 0.4595674 0.5563055
Other Infectious Diseases   0.0576625 0.0319657 0.0564092 0.0940048
Epilepsy                    0.0090355 0.0059884 0.0088742 0.0136374
COPD                        0.0052431 0.0033505 0.0052112 0.0081271
Malaria                     0.0006562 0.0004267 0.0006519 0.0009299
Diabetes                    0.0004767 0.0002977 0.0004612 0.0007221
Acute Myocardial Infarction 0.0003786 0.0002838 0.0003743 0.0005002
Falls                       0.0002990 0.0001571 0.0002635 0.0006124
Renal Failure               0.0002927 0.0001991 0.0002871 0.0004097
```

Notice for N deaths, C causes, the individual COD distribution with C.I can be represented by a $(N \times C \times 4)$-dimensional array, where the 4 dimensions are mean, median, lower bound, upper bound respectively. The function `get.indiv()` obtains this information in the form of a list of 4 matrices of dimension N by C, which can then be saved to other formats to facilitate further analysis. We can also change the desired width of C.I. when extracting the results, too.

```
R> insilico_prob <- get.indiv(insilico2, CI = 0.9)
R> head(insilico_prob$lower["6288", ])
R> head(insilico_prob$upper["6288", ])
```

We can extract the whole N by C matrix of individual COD distribution for other methods (except Tariff) too

```
R> allprobs <- getIndivProb(interva2)
R> dim(allprobs)
```

```
[1] 1554   34
```

```
R> # head(allprobs)
```

5.4. Visualization

The previous sections discuss how results could be extracted and examined in R. In this section, we show some visualization tools provided in the **openVA** package for presenting these results. The fitted CSMFs for the top causes can be easily visualized by

```
R> plotVA(tariff2,  title = "Tariff")
R> plotVA(interva2,  title = "InterVA")
R> plotVA(nbc2, title = "NBC")
R> plotVA(insilico2, title = "InSilicoVA", bw = TRUE)
```

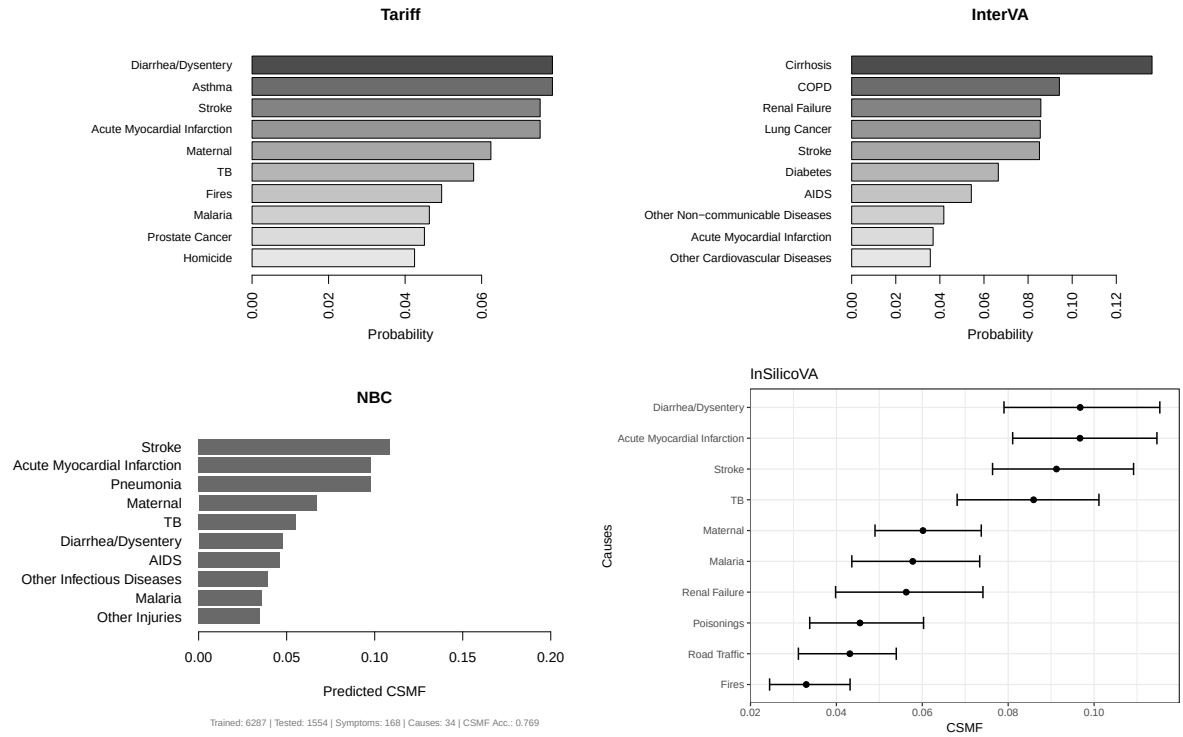

Figure 5: Plot of top 10 CSMFs for each method using plotVA() function.

The CSMFs can also be aggregated for easier visualization of groups of causes. For InterVA-4 cause list, we included an example grouping built into the package, so the aggregated CSMFs can be compared by

```
R> compare <- list(InterVA4_02 = interva1_4_02,
+                  InterVA4_03 = interva1_4_03,
+                  InSilicoVA = insilico1)
R> stackplotVA(compare, sample.size.print = TRUE,
+              xlab = "", angle = 0)
R> stackplotVA(compare, sample.size.print = TRUE,
+              xlab = "", angle = 0, type = "dodge")
```

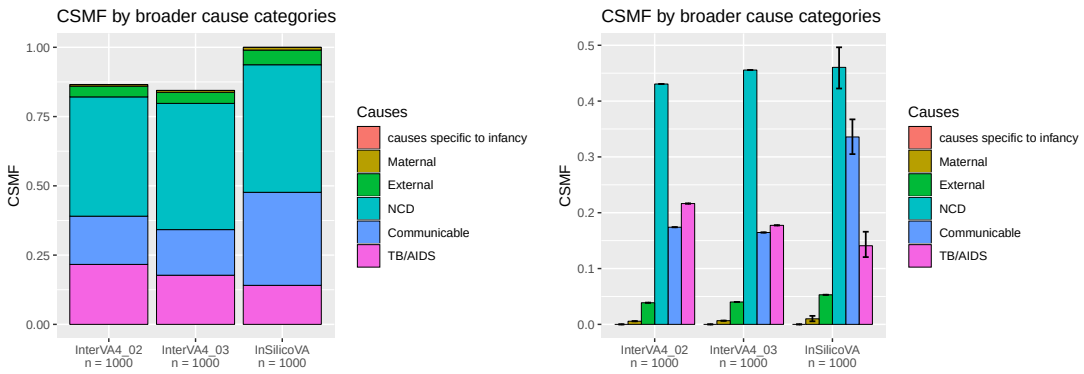

Figure 6: Comparing aggregated CSMF for InterVA-4 and InSilicoVA.

The missing proportion in the stack plot is due to the "undetermined" category, which is not in the example grouping variable. In practice, the grouping of causes of deaths often needs to be determined according to context and the research question of interest. Changing the grouping can be easily achieved by modifying the `grouping` argument in `stackplotVA()` function. For example, we may add another row to the matrix for the undetermined cause:

```
R> # also show changing group here
R> data(SampleCategory)
R> grouping <- SampleCategory
R> grouping[,1] <- as.character(grouping[,1])
R> grouping <- rbind(grouping, c("Undetermined", "Undetermined"))
R> stackplotVA(compare, sample.size.print = TRUE,
+              xlab = "", angle = 0,
+              grouping = grouping)
```

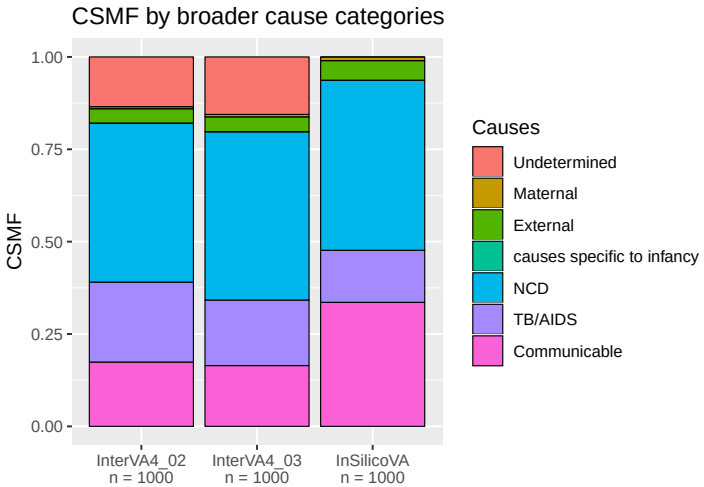

Figure 7: Comparing aggregated CSMF for InterVA-4 and InSilicoVA, adding undetermined category.

The ordering of the stacked bars can also be changed to reflect the structures within the aggregated causes.

```
R> order.group <- c("TB/AIDS",  "Communicable", "NCD",
+              "External", "Maternal",
```

```
+              "causes specific to infancy",
+              "Undetermined")
R> stackplotVA(compare, sample.size.print = TRUE,
+              xlab = "", angle = 0,
+              grouping = grouping, order.group = order.group)
```

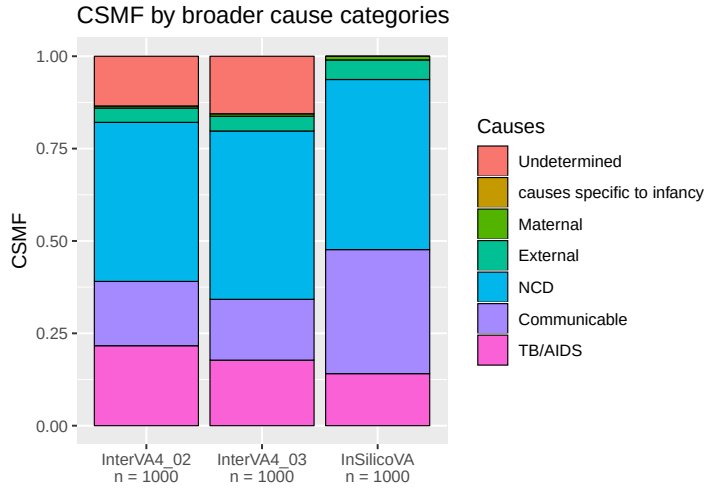

Figure 8: Comparing aggregated CSMF for InterVA-4 and InSilicoVA, with the causes re-ordered.

Finally, one can construct stacked plots for any list of causes by manually specifying a grouping rule. See below for an example with the PHMRC data.

```
R> cod.phmrc <- c(unique(as.character(train[, "gs_text34"])),
+                 "Undetermined")
R> group.phmrc <- c(rep("NCD", 3), "External", "NCD", "HIV/TB",
+                   "NCD", "Maternal", "External", "NCD", "External", "NCD",
+                   "External", rep("NCD", 2), "other infectious disease", "NCD",
+                   "External", "NCD", "HIV/TB", "other infectious disease",
+                   rep("NCD", 3), "External", rep("NCD", 2), rep("External", 2),
+                   rep("NCD", 4), "External", "Undetermined")
R> grouping2 <- cbind(cod.phmrc, group.phmrc)
R> compare2 <- list(Tariff = tariff2,
+                   InterVA = interva2,
+                   NBC = nbc2,
+                   InSilicoVA = insilico2)
R> stackplotVA(compare2, sample.size.print = TRUE,
+              xlab = "", angle = 0,
+              grouping = grouping2)
```

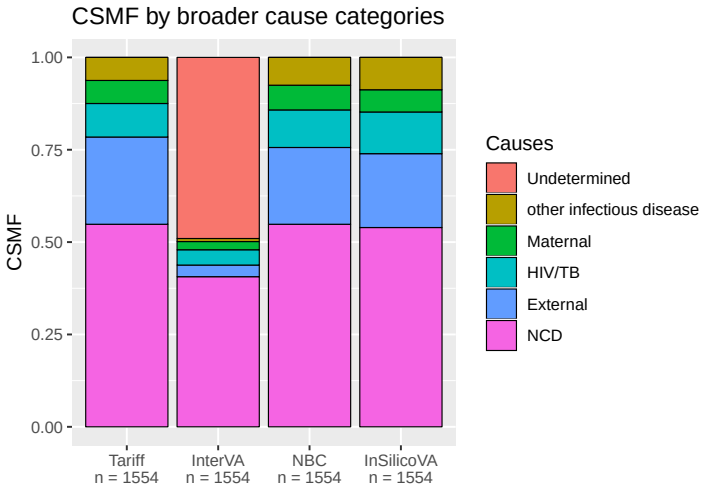

Figure 9: Comparing aggregated CSMF for InterVA-4 and InSilicoVA, with the causes re-ordered.

6. Additional functionality of InSilicoVA

In this subsection we first discuss two major changes to the InSilicoVA algorithm since the original publication (McCormick *et al.* 2016) in Section 6.1 and Section 6.2. We then illustrate some additional analysis using features unique to InSilicoVA method.

6.1. Removal of physically impossible causes

The originally proposed InSilicoVA assumes all causes of death are possible for each observation. The impact form such assumption is mild when data are abundant, but could be problematic when either sample size is small or the proportion of missing data is high. In both cases, physically impossible causes might get assigned with non-ignorable posterior mass (with wide posterior credible interval). Since the version 1.1.5 of **InSilicoVA**, the algorithm automatically checks and removes impossible causes before fitting the model. The k-th cause is defined as physically impossible for the i-th death if $P(s_{ij} = 1|y_i = k) = 0$ for any j representing either gender or age group that presents. We then consider a cause to be physically impossible to the underly population if it is impossible for all the observations of the input data. For example, with the new implementation, CSMF for pregnancy-related causes will not be estimated if the input data consist of only male deaths.

6.2. Structured symptom dependence

Another change in InSilicoVA algorithm is the implementation of the data consistency check. Inherited from the InterVA-4 algorithm, InSilicoVA performs a data consistency check before fitting the algorithm as described in Algorithm 1. This consistency check consists of two parts. First, it removes indicators whose presence contradicts the existence of a lower-level indicator. For example, if the question *pregnant at the time of death* is answered with an "Yes", then the algorithm makes sure that the item *child* is set to "No". The second part of the algorithm checks the existence of each higher-level indicator when lower-level indicators that implied by it exists. Again, if the question *pregnant at the time of death* is answered with an "Yes", then the algorithm makes sure that the item *female* is set to "Yes" as well.

Algorithm 1 Data Check

Require:
$\mathbf{s}_j$ = vector of $j = 1$ to 245 indicators for one death
notask(s): higher-level indicator(s) whose presence ensures indicator s does not exist
anc(s): higher-level indicator(s) that must exist if indicator s exists

Ensure: $\mathbf{s}^*$ = modified vector of $j = 1$ to 245 indicators for one death

1: **for** each indicator $j = 1$ to 245 **do**
2: Set $\mathbf{s}_j^* \leftarrow \mathbf{s}_j$ ▷ Initialize
3: **end for**
4: **for** each indicator $j = 1$ to 245 **do**
5: **if** there exists $\mathbf{s}_\ell^*$ in notask($\mathbf{s}_j^*$) and $\mathbf{s}_\ell^*$ is 'yes' **then**
6: $\mathbf{s}_j^* \leftarrow$ 'no' ▷ This indicator is nonsensical, set to 'No'
7: **end if**
8: **if** there exists $\mathbf{s}_\ell^*$ in anc($\mathbf{s}_j$) and $\mathbf{s}_j^*$ is 'yes' **then**
9: $\mathbf{s}_\ell^* \leftarrow$ 'yes' ▷ Set more general version of this indicator to 'yes'
10: **end if**
11: **end for**
12: Repeat loop in lines 4 to line 11 again ▷ Processes 2-level hierarchies

This data check procedure ensures the collected data are internal consistent, yet it might introduce additional complications under the common assumption that symptoms are conditionally independent given any cause of death. This can be illustrated with a simple example below. In calculating the conditional probability of observing two symptoms, e.g., "infant", and "recent abortion", we can decompose

$$\begin{aligned}\Pr(\text{infant} = Y \;\&\; \text{recent abortion} = N \mid \text{some cause}) &= \Pr(\text{infant} = Y \mid \text{some cause}) \\ &* \Pr(\text{recent abortion} = N \mid \text{some cause}).\end{aligned}$$

However, such independence does not hold since the two symptoms are mutually exclusive, i.e., the presence of symptom "infant" implies no recent abortion. In fact, the structure of the WHO instrument assures that the lower-level question will not be asked at all if the interview if about an infant death. Thus in practice, the above probability calculation yields an underestimate of the target value and it should be calculated simply with

$$\Pr(\text{infant} = Y \;\&\; \text{recent abortion} = N \mid \text{some cause}) = \Pr(\text{infant} = Y \mid \text{some cause})$$

To reflect the different joint probability decompositions when the known symptom hierarchy is available, it turns out we can easily modify the data check algorithm by imputing missing instead of absence to the lower level symptoms in line 6. We have found that such a change usually yield more reasonable CSMF estimates, especially for child and neonate deaths, in practice. Also it should be noticed that this problem stems from the conditional independence assumption that has been adopted in all other methods besides InSilicoVA. The computational trick here only eliminates the bias from mutually exclusive symptoms known from the hierarchical structure of VA questionnaires. More systematic ways to deal with symptom correlations are needed to further improve the estimation.

With the update of the WHO VA questionnaire to the 2016 format, three major changes were made to the data check procedure implemented by InterVA-5. First, similar to InSilicoVA,

InterVA-5 also imputes missing instead of absence to the lower level symptoms in line 6. Second, the hierarchical structure of the 2016 VA questionnaire makes use of both "Yes" and "No" (or "Y" and "N") as substantive values that trigger the asking of lower-level questions. For example, if the indicator/question "Did the baby ever cry?" has the response "No", then the following indicator/question, "Did the baby cry immediately after birth, even if only a little bit?" should not be asked – as implemented with the notask(s) function in Algorithm 1. Thus, additional steps are needed in the InterVA-5 data check which test the value of each indicator with the appropriate substantive response that implies subsequent action (i.e., asking or not asking a different indicator/question) in line 5 and 8. The last major change is that an additional check is made for indicators/questions that only apply to neonates. If a record for a non-neonate contains a value for an indicator/question that only applies to neonates, then that value is cleared in the working copy of the data file. The same check rules are used by InSilicoVA for WHO 2016 data as well[4].

6.3. Sub-population specific CSMFs

The Bayesian framework adopted by InSilicoVA allows the specification of sub-population in analyzing VA data. For example, when researchers want to estimate different CSMFs for multiple regions within the same country, or for different groups in the population, running separate models on subsets of data can be inefficient and does not allow each sub-population to borrow information from others. Instead, we can estimate different CSMF within each sub-population, but allowing them to share information from the jointly estimated conditional probability matrix, $P_{s|c}$. In this subsection, we show how to estimate different CSMFs for sub-populations specified by gender and age groups, using the randomly sampled ALPHA dataset.

First, in the dataset, we can include additional columns stating the sub-population each death belongs to.

```
R> data(RandomVA2)
R> head(RandomVA2[, 244:248])
```

```
  stradm smobph scosts   sex age
1      .      .       .   Men 60+
2      .      .       . Women 60-
3      .      .       . Women 60-
4      .      .       . Women 60+
5      .      .       . Women 60-
6      .      .       . Women 60-
```

Then we can fit the model with one or multiple additional columns specifying sub-population membership for each observation.

```
R> fit_ins1<- codeVA(RandomVA2, model = "InSilicoVA",
+                subpop = list("sex"),  indiv.CI = 0.95,
+                Nsim = 10000, auto.length = FALSE)
R> fit_ins2<- codeVA(RandomVA2, model = "InSilicoVA",
+                subpop = list("sex", "age"),  indiv.CI = 0.95,
+                Nsim = 10000, auto.length = FALSE)
```

The CSMFs for each of the sub-populations will be shown in the `summary` method.

[4]The implementation is slightly different between InSilicoVA and InterVA-5. InterVA-5 updates s_j^* to be missing in line 6 only when it is recorded to be the corresponding substantive value, since the non-substantive value is considered the same as missing values in the calculation. InSilicoVA, on the other hand, make the update regardless of the value of s_j^*, since the non-substantive value and missing are treated differently in the algorithm.

```
R> summary(fit_ins1, top=5)
```

```
InSilicoVA Call:
1000 death processed
10000 iterations performed, with first 5000 iterations discarded
 500 iterations saved after thinning
Fitted with re-estimated conditional probability level table
Data consistency check performed as in InterVA4
Sub population frequencies:
  Men Women
  460   540

Men - Top 5 CSMFs:
                                  Mean Std.Error  Lower Median
Other and unspecified infect dis 0.2531    0.0228 0.2131 0.2524
Renal failure                    0.0876    0.0145 0.0627 0.0881
HIV/AIDS related death           0.0858    0.0145 0.0604 0.0851
Digestive neoplasms              0.0550    0.0118 0.0361 0.0541
Other and unspecified neoplasms  0.0526    0.0121 0.0311 0.0524
                                  Upper
Other and unspecified infect dis 0.2998
Renal failure                    0.1198
HIV/AIDS related death           0.1159
Digestive neoplasms              0.0794
Other and unspecified neoplasms  0.0750

Women - Top 5 CSMFs:
                                   Mean Std.Error  Lower Median
Other and unspecified infect dis  0.2759    0.0210 0.2376 0.2759
Renal failure                     0.1095    0.0164 0.0778 0.1090
HIV/AIDS related death            0.1083    0.0135 0.0851 0.1078
Other and unspecified cardiac dis 0.0690    0.0125 0.0462 0.0683
Other and unspecified neoplasms   0.0672    0.0134 0.0443 0.0662
                                   Upper
Other and unspecified infect dis  0.3172
Renal failure                     0.1422
HIV/AIDS related death            0.1370
Other and unspecified cardiac dis 0.0966
Other and unspecified neoplasms   0.0978
```

All stated functions in the previous sections works in the same way for the fitted object with multiple sub-population. Additional visualization tools are also available. By specify `type = "compare"`, we can plot the CSMFs for two sub-populations on the same plot.

```
R> plotVA(fit_ins1, type = "compare", title = "Comparing CSMFs")
```

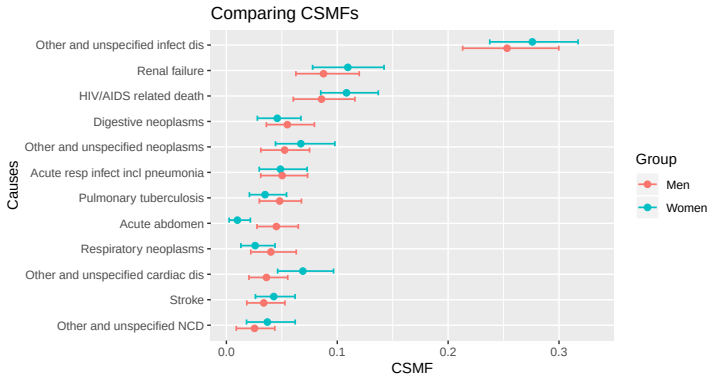

Figure 10: Comparing CSMF for different sub-population.

By default, the comparison plots will select all the CODs that are included in the top K for each of the sub-populations. We can also plot only subsets of them by specifying with causes are of interest.

```
R> plotVA(fit_ins1, type = "compare", title = "Comparing CSMFs",
+                  causelist = c("HIV/AIDS related death",
+                                "Pulmonary tuberculosis",
+                                "Other and unspecified infect dis",
+                                "Other and unspecified NCD"))
```

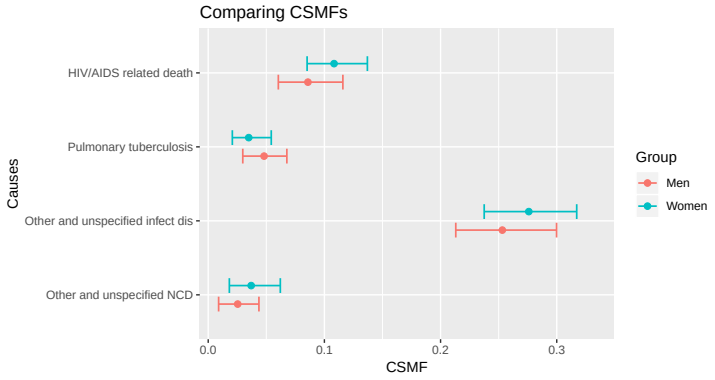

Figure 11: Comparing CSMF for different sub-population in selected CODs.

We can also plot each sub-population alone.

```
R> plotVA(fit_ins1, which.sub = "Men", title = "Men")
R> plotVA(fit_ins1, which.sub = "Women", title  ="Women")
```

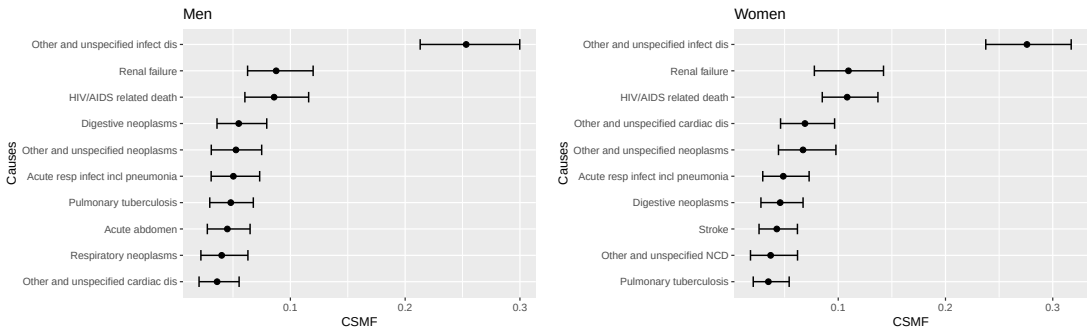

Figure 12: Comparing CSMF for different sub-population in separate plots.

This also applies to the stack plot discussed before.

```
R> stackplot(fit_ins1)
R> stackplot(fit_ins1, type = "dodge")
```

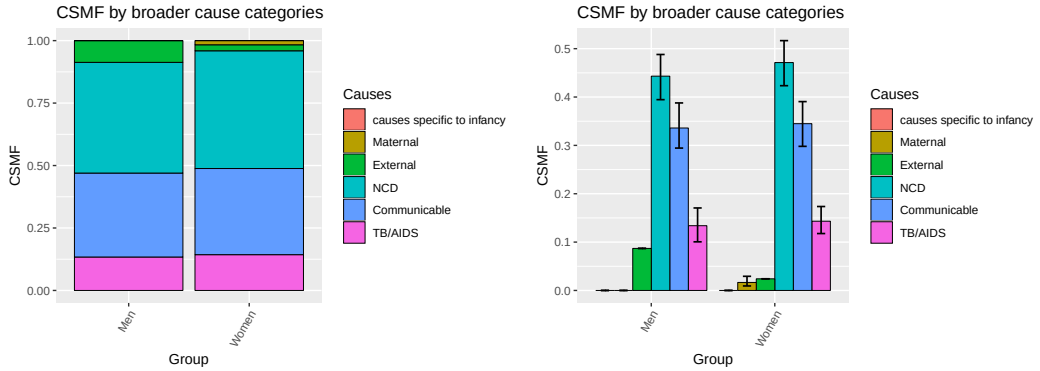

Figure 13: Comparing aggregated CSMF for two different sub-population.

```
R> stackplot(fit_ins2)
R> stackplot(fit_ins2, type = "dodge")
```

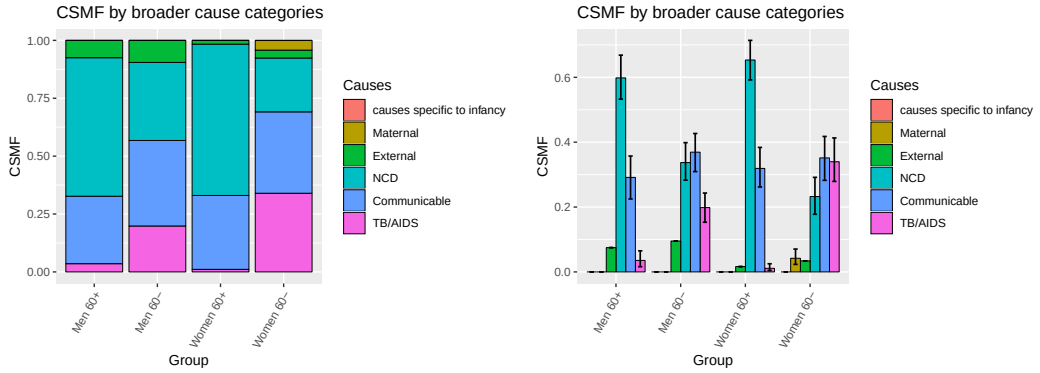

Figure 14: Comparing aggregated CSMF for four different sub-population.

6.4. Aggregated COD distribution of sub-populations

Sub-population specification is useful for many scenarios, yet it is usually not recommended to specify many sub-populations with small sample sizes. This is because as the number of

sub-population increases, the complexity of the model also increases, and so does the uncertainty around the estimated CSMFs. Another approach to obtain sub-population level CSMFs with small sample sizes is to approximate them by the aggregated COD distribution within that sub-population. In fact, for other methods except InSilicoVA, the CSMF and aggregated COD distribution are considered to be the same. For InSilicoVA, on the other hand, since the CSMFs are specifically estimated, they typically do not equal the aggregated individual COD distribution and contain larger uncertainties. The difference is usually subtle but conceptually very important. So in this subsection, we provide a brief overview of the difference with the focus only on InSilicoVA.

Assuming the deaths in the data are a representable sample of some population researchers want to study, InSilicoVA aims to estimate the CSMF for this larger population directly, instead of estimating the fraction of causes within the samples. However, once individual COD distribution is estimated, by aggregating the probability of each COD across every deaths in the data, one could obtain another COD distribution, which we refer to the "aggregated COD distribution" and is usually used and interpreted as CSMF by all other VA algorithms.

When making inference about the underlying population from which the data are collected, the true CSMF estimator tends to provide a more robust estimation and more accurate uncertainties than the aggregated COD distribution. However, the aggregated COD distribution could still be useful in some situations. For example, sometimes the research question is to compare COD distributions within different subsets of data, such as 5-year age groups. Sample sizes within each age group could be too small to estimate its own population CSMF, but if researchers are willing to assume the population CSMF does not differ dramatically within some larger age group categories, one could fit the model with the larger age group as sub-population, and then aggregate COD distribution for each 5-year age groups. However, it should always be noticed that such analysis compares the average probability distribution in the collected data, rather than the underlying population.

The aggregated COD distribution for the whole dataset could be easily obtained by the `get.indiv` function with the argument `is.aggregate` set to be `TRUE`.

```
R> agg.csmf <- get.indiv(data = RandomVA2, insilico1, CI = 0.95,
+                         is.aggregate = TRUE, by = NULL)
R> head(agg.csmf)
```

```
                                     Mean    Median     Lower
Sepsis (non-obstetric)           3.642e-04 3.587e-04 2.373e-04
Acute resp infect incl pneumonia 4.931e-02 4.906e-02 4.564e-02
HIV/AIDS related death           1.004e-01 1.005e-01 9.725e-02
Diarrhoeal diseases              5.923e-03 5.934e-03 4.747e-03
Malaria                          6.552e-05 2.097e-05 5.469e-06
Measles                          1.876e-09 5.626e-10 6.024e-11
                                     Upper
Sepsis (non-obstetric)           4.903e-04
Acute resp infect incl pneumonia 5.589e-02
HIV/AIDS related death           1.036e-01
Diarrhoeal diseases              7.146e-03
Malaria                          3.446e-04
Measles                          1.028e-08
```

Aggregation by sub-population is also simple to specify. Here we use the fitted model from previous sections where no sub-population is specified.

```
R> agg.by.sex.age <- get.indiv(data = RandomVA2, insilico1, CI = 0.95,
```

```
+                              is.aggregate = TRUE, by = list("sex", "age"))
R> head(agg.by.sex.age$mean)

                                   Men 60+   Men 60- Women 60+
Sepsis (non-obstetric)           3.998e-04 7.212e-04 1.273e-04
Acute resp infect incl pneumonia 3.738e-02 6.206e-02 4.350e-02
HIV/AIDS related death           1.351e-02 1.417e-01 2.532e-03
Diarrhoeal diseases              2.247e-04 3.898e-03 3.526e-03
Malaria                          9.322e-05 4.838e-05 9.460e-05
Measles                          1.223e-10 2.146e-09 5.824e-10
                                 Women 60-
Sepsis (non-obstetric)           2.284e-04
Acute resp infect incl pneumonia 5.151e-02
HIV/AIDS related death           2.487e-01
Diarrhoeal diseases              1.592e-02
Malaria                          2.566e-05
Measles                          4.637e-09
```

We can visualize these aggregated distributions with

```
R> indivplot(agg.csmf, top = 5, title = "Population aggregated COD distribution")
R> indivplot(agg.by.sex.age, which.plot = "Men 60-", top = 5,
+            title = "Men 60- aggregated COD distribution")
```

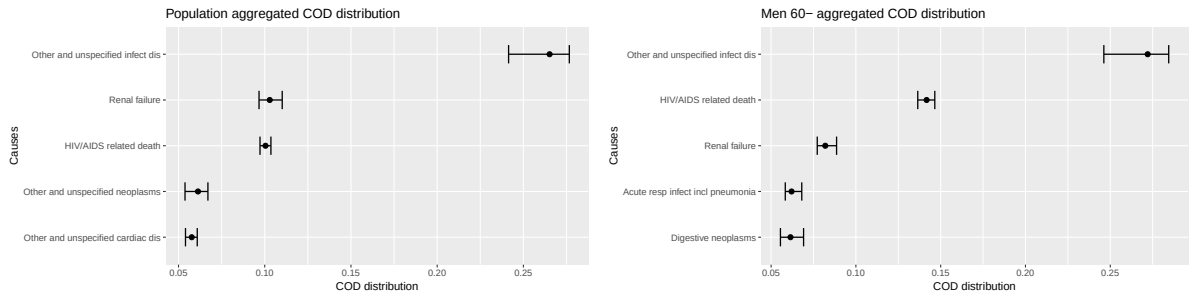

Figure 15: Aggregated COD distribution for all data or a subset of data

And similarly we can compare the specified subsets of the aggregated distribution

```
R> indivplot(agg.by.sex.age, which.plot = list("Men 60+", "Men 60-"),
+            top = 5, title = "Top CODs: Men 60+ v.s. Men 60-")
R> indivplot(agg.by.sex.age, which.plot = list("Men 60+", "Men 60-"),
+                            top = 0, causelist = c(
+                              "HIV/AIDS related death",
+                              "Pulmonary tuberculosis",
+                              "Other and unspecified infect dis",
+                              "Other and unspecified NCD"),
+                            title = "Selected CODs: Men 60+ v.s. Men 60-")
```

6.5. Physician coding

This section illustrates very briefly how physician coded death can be incorporated. Currently we only allow physician coded causes to be either the same as the CODs used for the final algorithm, or a higher level aggregation of them. For each death coded by physicians, we first de-bias the multiple codes provided from different physicians.

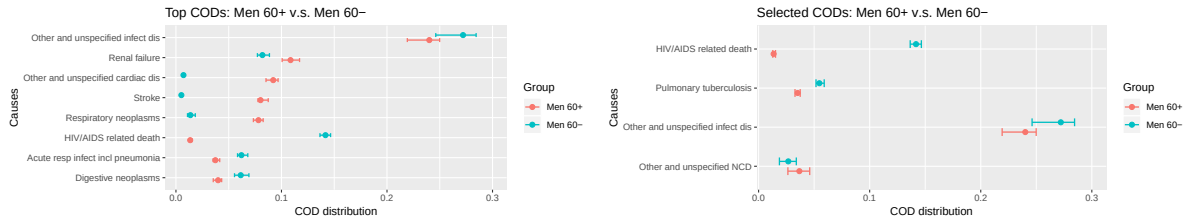

Figure 16: Comparing aggregated COD distribution for different subsets of data

```
R> data(SampleCategory)
R> data(RandomPhysician)
R> head(RandomPhysician[, 245:250])
```

```
  smobph scosts        code1 rev1        code2 rev2
1      .      .          NCD doc9          NCD doc6
2      .      .          NCD doc4          NCD doc3
3      .      .          NCD doc1          NCD doc5
4      .      .      TB/AIDS doc4      TB/AIDS doc7
5      .      .      TB/AIDS doc5      TB/AIDS doc9
6      .      . Communicable doc9 Communicable <NA>
```

The de-bias process is described in more detail in McCormick *et al.* (2016). We omit the details here. For the purpose of implementation, we only need to specify which columns are physician IDs, and which are their coded causes respectively.

```
R> doctors <- paste0("doc", c(1:15))
R> causelist <- c("Communicable", "TB/AIDS", "Maternal",
+                 "NCD", "External", "Unknown")
R> phydebias <- physician_debias(RandomPhysician,
+            phy.id = c("rev1", "rev2"), phy.code = c("code1", "code2"),
+            phylist = doctors, causelist = causelist,
+            tol = 0.0001, max.itr = 100)
```

The de-biased step essentially creates a prior probability distribution for each death over the broader categories of causes. Then to run InSilicoVA with the de-biased physician coding, we can simply pass the fitted object from the previous step to the model. Additional arguments are needed to specify the external cause category, since they are handled by separated heuristics, and the unknown category, which is equivalent to an uniform probability distribution over all other categories, i.e., the same as the case where no physician coding exist.

```
R> fit_ins_phy <- codeVA(RandomVA1, model = "InSilicoVA",
+                phy.debias = phydebias, phy.cat = SampleCategory,
+                phy.external = "External", phy.unknown = "Unknown",
+                Nsim = 10000, auto.length = FALSE)
```

This can be compared with the previous results without including physicians' codes.

```
R> plotVA(insilico1)
R> plotVA(fit_ins_phy)
```

7. Conclusion

In this paper, we introduce the **openVA** package. This is the first open-sourced software that implements and compares the major VA methods. The **openVA** package allows researchers to

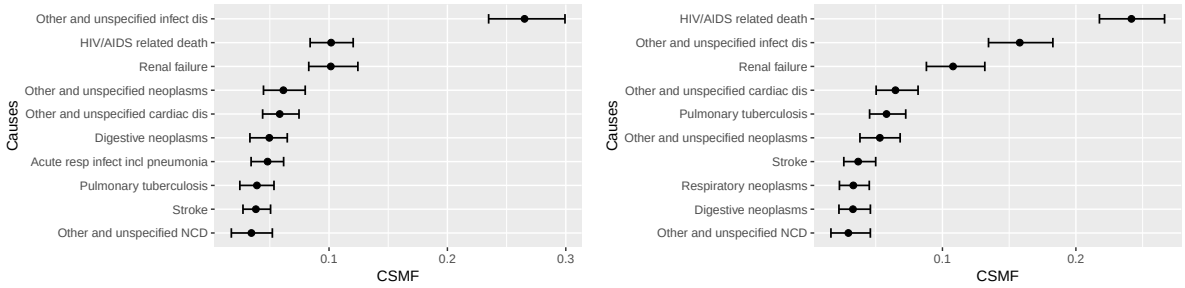

Figure 17: Comparing fitted CSMF with and without physicians

easily access the most recent tools that are previously difficult to work with or unavailable. It also enables the compatibility of multiple data input formats and significantly reduces the tedious work to pre-process different data formats specific to each algorithm. The software framework of the **openVA** package allows the integration of new methods in the future, such as the InterVA-5 model currently under development. The **openVA** package makes all the steps involved in analyzing VA data, i.e., data processing, model tunning and fitting, result summarization, and evaluation metrics, transparent and reproducible. This contributes significantly to the public health community using VA.

Finally, We note that two additional features will be helpful for future development. First, for users not familiar with the command line tools or does not have access to R on their local machines, a shiny front end hosted on a centralized server can be very useful, and may allow field workers to obtain real-time cause-of-death assessment. Second, although in this paper, we aim to provide users with all the available methods for assigning causes of death, without comparing the accuracy and robustness between them, much future work is needed to systematically assess these methods and maybe also combine them to provide a better analysis framework.

References

Byass P (2015). "InterVA-4 Software [Windows Executable]." *www.interva.net*.

Byass P (2018). "InterVA-5 Software [Windows Executable]." *www.interva.net*.

Byass P, Chandramohan D, Clark SJ, D'Ambruoso L, Fottrell E, Graham WJ, Herbst AJ, Hodgson A, Hounton S, Kahn K, *et al.* (2012). "Strengthening Standardised Interpretation of Verbal Autopsy Data: The New InterVA-4 Tool." *Global Health Action*, **5**.

Choi E, Clark S, Li ZR, Maire N, Thomas J (2019). ***CrossVA**: Verbal Autopsy Data Transform for Use with Various Coding Algorithms*. R package version 0.9.5, URL https://CRAN.R-project.org/package=CrossVA.

D'Ambruoso L, Boerma T, Byass P, Fottrell E, Herbst K, Kallander K, Mullan Z (2017). "The case for verbal autopsy in health systems strengthening." *LANCET GLOBAL HEALTH*, **5**(1), E20–E21. ISSN 2214-109X. doi:{10.1016/S2214-109X(16)30332-1}.

Garenne M (2014). "Prospects for automated diagnosis of verbal autopsies." *BMC Medicine*, **12**(1), 18.

James SL, Flaxman AD, Murray CJ, Consortium Population Health Metrics Research (2011). "Performance of the Tariff Method: validation of a simple additive algorithm for analysis of verbal autopsies." *Population Health Metrics*, **9**(31).

Kahn K, Collinson MA, Gómez-Olivé FX, Mokoena O, Twine R, Mee P, Afolabi SA, Clark BD, Kabudula CW, Khosa A, *et al.* (2012). "Profile: Agincourt Health and Socio-Demographic Surveillance System." *International Journal of Epidemiology*, **41**(4), 988–1001.

Li Z, McCormick T, Clark S (2019). *openVA: Automated Method for Verbal Autopsy*. R package version 1.0.8, URL https://github.com/verbal-autopsy-software/openVA.

Li ZR, McCormick TH, Clark SJ (2014). "InterVA4: An R package to analyze verbal autopsy data." *Center for Statistics and the Social Sciences Working Paper, No.146*.

Li ZR, McCormick TH, Clark SJ (2018a). ***InSilicoVA**: Probabilistic Verbal Autopsy Coding with 'InSilicoVA' Algorithm*. R package version 1.2.6, URL http://CRAN.R-project.org/package=InSilicoVA.

Li ZR, McCormick TH, Clark SJ (2018b). ***Tariff**: Replicate Tariff Method for Verbal Autopsy*. R package version 1.0.6, URL http://CRAN.R-project.org/package=Tariff.

Li ZR, McCormick TH, Clark SJ, Byass P (2018c). ***InterVA4**: Replicate and Analyse 'InterVA4'*. R package version 1.7.5, URL http://CRAN.R-project.org/package=InterVA4.

McCormick TH, Li ZR, Calvert C, Crampin AC, Kahn K, Clark SJ (2016). "Probabilistic Cause-of-Death Assignment Using Verbal Autopsies." *Journal of the American Statistical Association*, **111**(515), 1036–1049. doi:10.1080/01621459.2016.1152191. http://dx.doi.org/10.1080/01621459.2016.1152191, URL http://dx.doi.org/10.1080/01621459.2016.1152191.

Miasnikof P, Giannakeas V, Gomes M, Aleksandrowicz L, Shestopaloff AY, Alam D, Tollman S, Samarikhalaj A, Jha P (2015). "Naive Bayes classifiers for verbal autopsies: comparison to physician-based classification for 21,000 child and adult deaths." *BMC medicine*, **13**(1), 1.

Murray CJ, Lopez AD, Black R, Ahuja R, Ali SM, Baqui A, Dandona L, Dantzer E, Das V, Dhingra U, *et al.* (2011a). "Population Health Metrics Research Consortium gold standard verbal autopsy validation study: design, implementation, and development of analysis datasets." *Population health metrics*, **9**(1), 27.

Murray CJ, Lozano R, Flaxman AD, Vahdatpour A, Lopez AD (2011b). "Robust metrics for assessing the performance of different verbal autopsy cause assignment methods in validation studies." *Popul Health Metr*, **9**, 28.

Nichols EK, Byass P, Chandramohan D, Clark SJ, Flaxman AD, Jakob R, Leitao J, Maire N, Rao C, Riley I, Setel PW, Grp WVAW (2018). "The WHO 2016 verbal autopsy instrument: An international standard suitable for automated analysis by InterVA, InSilicoVA, and Tariff 2.0." *PLOS MEDICINE*, **15**(1). ISSN 1549-1676. doi:{10.1371/journal.pmed.1002486}.

R Core Team (2014). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. URL http://www.R-project.org.

Serina P, Riley I, Stewart A, Flaxman AD, Lozano R, Mooney MD, Luning R, Hernandez B, Black R, Ahuja R, *et al.* (2015a). "A shortened verbal autopsy instrument for use in routine mortality surveillance systems." *BMC medicine*, **13**(1), 1.

Serina P, Riley I, Stewart A, James SL, Flaxman AD, Lozano R, Hernandez B, Mooney MD, Luning R, Black R, *et al.* (2015b). "Improving performance of the Tariff Method for assigning causes of death to verbal autopsies." *BMC medicine*, **13**(1), 1.

Taylor CE, Parker R, Reinke W, Faruquee R, *et al.* (1983). *Child and maternal health services in rural India. The Narangwal experiment. 2. Integrated family planning and health care.* Johns Hopkins University Press.

Thomas J, Li ZR, McCormick TH, Clark SJ, Byass P (2018). ***InterVA5**: Replicate and Analyse 'InterVA5'*. R package version 1.0.2, URL http://CRAN.R-project.org/package=InterVA5.

Wen R, Miasnikof P, Giannakeas V, Gomes M (2018). ***nbc4va**: Bayes Classifier for Verbal Autopsy Data*. R package version 1.1, URL http://CRAN.R-project.org/package=nbc4va.

World Health Organization and others (2012). "Verbal Autopsy Standards: The 2012 WHO Verbal Autopsy Instrument Release Candidate 1." *Technical Report http://www.who.int/healthinfo/statistics/WHO_VA_2012_RC1_Instrument.pdf*, World Health Organization (WHO), Geneva.

Affiliation:

Zehang Richard Li
Department of Statistics
University of Washington
Seattle, WA USA
E-mail: zehangli@uw.edu
URL: zehangli.com

Journal of Statistical Software http://www.jstatsoft.org/
published by the Foundation for Open Access Statistics http://www.foastat.org/
MMMMMM YYYY, Volume VV, Issue II *Submitted:* yyyy-mm-dd
doi:10.18637/jss.v000.i00 *Accepted:* yyyy-mm-dd