The **openVA** toolkit for Verbal Autopsies

Zehang Richard Li, Tyler H. McCormick, Samuel J. Clark

December 7, 2017

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-4, 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.

Contents

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 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 Organization 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 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 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**, **nbc4va**, **InSilicoVA**, and **Tariff**, each of which implements one coding algorithm. Two survey formats

are considered in this paper: the WHO 2012 instrument after standard dichotomization procedures into binary variables (World Health Organization and others, 2012), and the IHME questionnaire in the format of the Population Health Medical Research Consortium (PHMRC) dataset (Murray 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 two of them, InterVA-4 and Tariff, are also available in proprietary software program that can be run only on Windows systems. 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. This feature 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.

[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 Structure of the openVA package

The **openVA** suite currently consist of four 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.

- **InterVA4** (Li et al., 2014) is an R package that implements the InterVA-4 model (Byass 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.

- **nbc4va** (Miasnikof et al., 2015) 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 et al., 2016a) 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 et al. (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 et al., 2016b) is an R package that implements the Tariff algorithm (James et al., 2011). It most closely reflects the description of Tariff 2.0 method (Serina et al., 2015a). 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 et al. (2015b). 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** algorithm (Byass et al., 2012) calculates 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 pre-defined 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 Tariff algorithm produce the COD distribution for each death in terms of their

rankings instead of the probability distributions. And thus the CSMF for each cause k are 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 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** (Maire, 2017). Thus in this section, we focus the discussion on the latter two tasks.

3.1 Standardizing format: InterVA

For users familiar with InterVA software and the associated data processing steps, the standard input format 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/index1.html.

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 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:

```
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.

```
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
## 177 binary symptoms generated
##
## Number of Yes        21027
## Number of No         124732
## Number of Not known  31241
```

```
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
## 177 binary symptoms generated
##
## Number of Yes        21715
## Number of No         124044
## Number of Not known  31241
```

Notice that the original PHMRC data is useful for comparing and validating new methods, as well as used 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[2].

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 ".".

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

```
toyData

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

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

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

```
data(RandomVA1)
dim(RandomVA1)
```

```
## [1] 1000  246
```

```
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 online, and then organize them into training and testing data.

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

```
## [1] 1554  946
```

```
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:

```
interva1_4_02 <- codeVA(data = RandomVA1, data.type = "WHO",
                        model = "InterVA", version = "4.02")
```

```
interva1_4_03 <- codeVA(data = RandomVA1, data.type = "WHO",
                        model = "InterVA", version = "4.03")
```

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

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

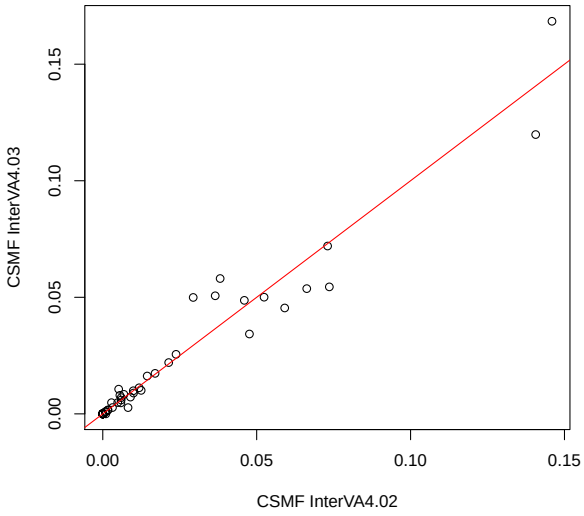

Figure 1: Comparing InterVA-4-4.02 and InterVA-4.03 fitted CSMF on the same dataset

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

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

4.2 PHMRC data: all methods

First, to fit the InterVA-4 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.

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

```
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.

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

```
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
## 177 binary symptoms generated
##
## Number of Yes        162507
## Number of No         978272
## Number of Not known  247078
##
## 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),

```
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
## 177 binary symptoms generated
##
## Number of Yes        162972
## Number of No         977807
## Number of Not known  247078
```

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

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

```
csmf_defaut <- getCSMF(tariff2)
csmf_adapt <- getCSMF(tariff2_a)
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")
abline(a=0,b=1,col = "red")
```

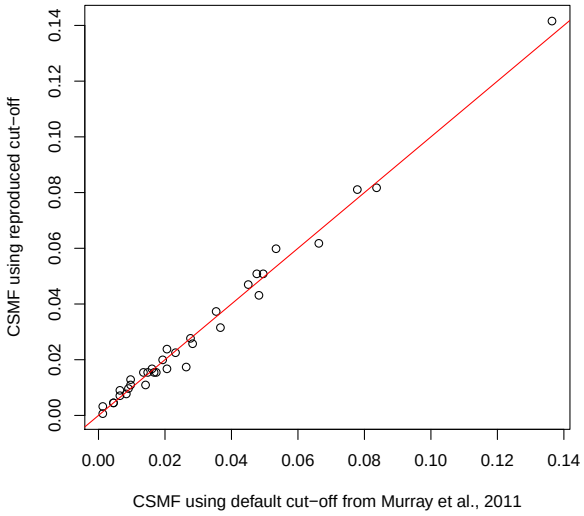

Figure 2: Comparing Tariff fitted CSMF under two different implementations

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

```
summary(tariff2)
summary(interva2)
summary(nbc2)
summary(insilico2)
```

We can also extract some more detailed results and compare them respectively.

5.1 CSMF

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

```
csmf.tariff2 <- getCSMF(tariff2)
csmf.interva2 <- getCSMF(interva2)
csmf.nbc2 <- getCSMF(nbc2)
csmf.insilico2 <- getCSMF(insilico2)
cbind(Tariff = csmf.tariff2, InterVA = csmf.interva2[1:34],
      NBC = csmf.nbc2, InSilicoVA = csmf.insilico2[, "Mean"])
```

We can now compare them to the true cause distribution of the dataset

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

csmf.all <- list(Tariff = csmf.tariff2, InterVA = csmf.interva2,
                 NBC = csmf.nbc2, InSilicoVA = csmf.insilico2[, "Mean"])

par(mfrow = c(2, 2))
for(i in 1:length(csmf.all)){
  plot(csmf.true, csmf.all[[i]][1:34], xlim = c(0, 0.15), ylim = c(0, 0.15),
      xlab = "True CSMF", ylab = "Fitted CSMF",
      main = names(csmf.all)[i])
  abline(a=0, b=1, col="red")
}
```

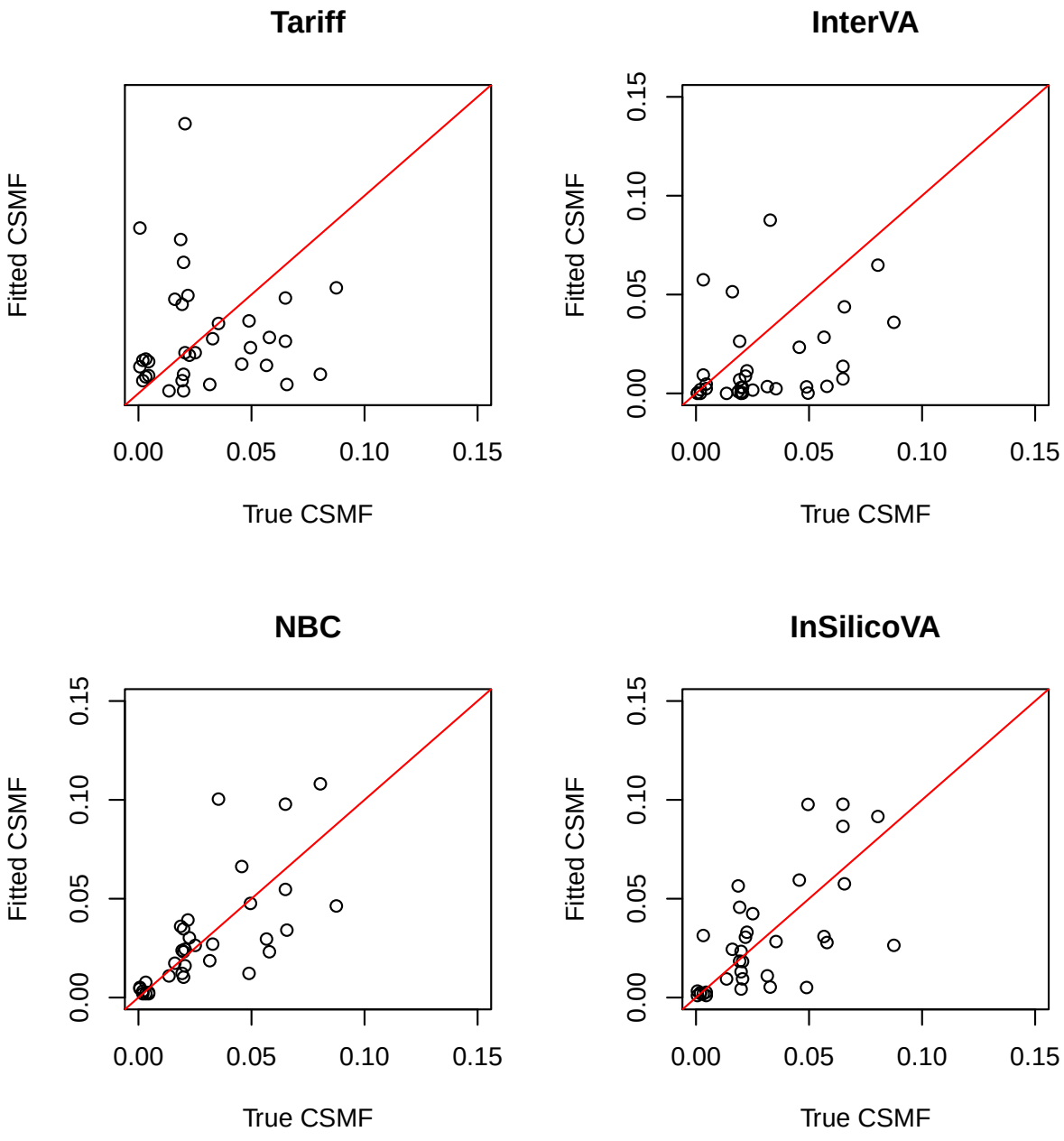

Figure 3: Comparing fitted CSMF with the truth for the four methods

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

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

```
getCSMF_accuracy(csmf.tariff2, csmf.true)

## [1] 0.5428

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

## [1] 0.5961

getCSMF_accuracy(csmf.nbc2, csmf.true)

## [1] 0.7682

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

## [1] 0.7282
```

Notice that, in particular 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:

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

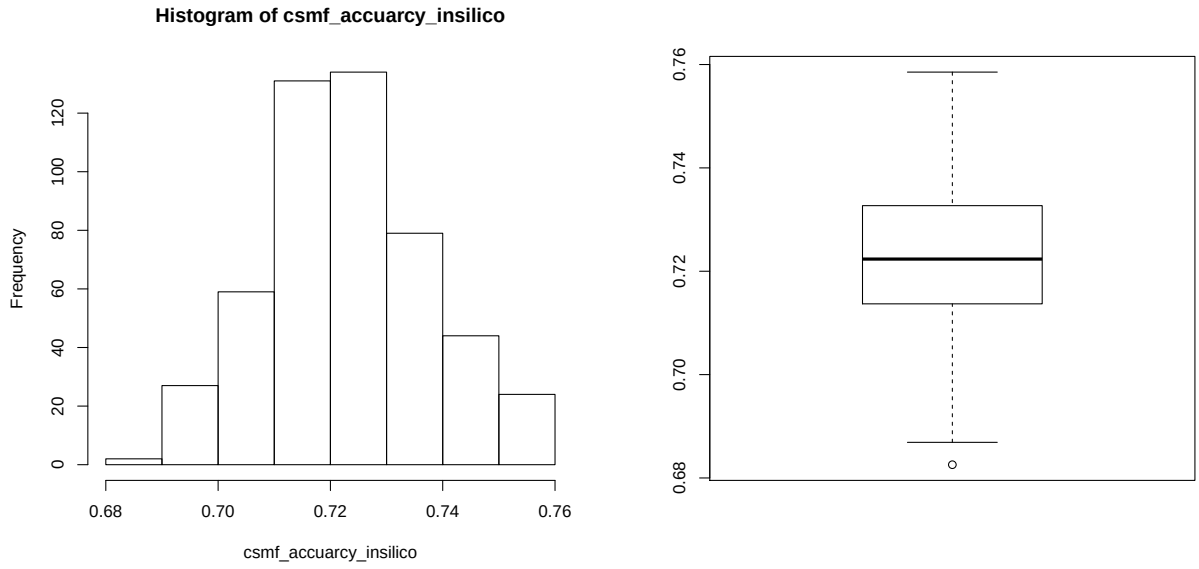

Figure 4: CSMF accuarcy histogram and box plot for InSilicoVA fit

Similarly we can compare the two versions of InterVA-4 and InSilico on the ALPHA dataset. More visualization options are explored in the later sections.

```
csmf.interva1_4_02 <- getCSMF(interva1_4_02)
csmf.interva1_4_03 <- getCSMF(interva1_4_03)
csmf.insilico1 <- getCSMF(insilico1)
```

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

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

Similarly we can compare these COD assignments to the true labels

```
# install.packages(c("lattice", "gplots"))
library(lattice)
library(gplots)
cod_names <- c(levels(test[, "gs_text34"]), "Undetermined")
cod.true <- factor(test[, "gs_text34"], levels = cod_names)
cod.all <- list(Tariff = cod.tariff2, InterVA = cod.interva2,
                NBC = cod.nbc2, InSilicoVA = cod.insilico2)
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")
  )
}
```

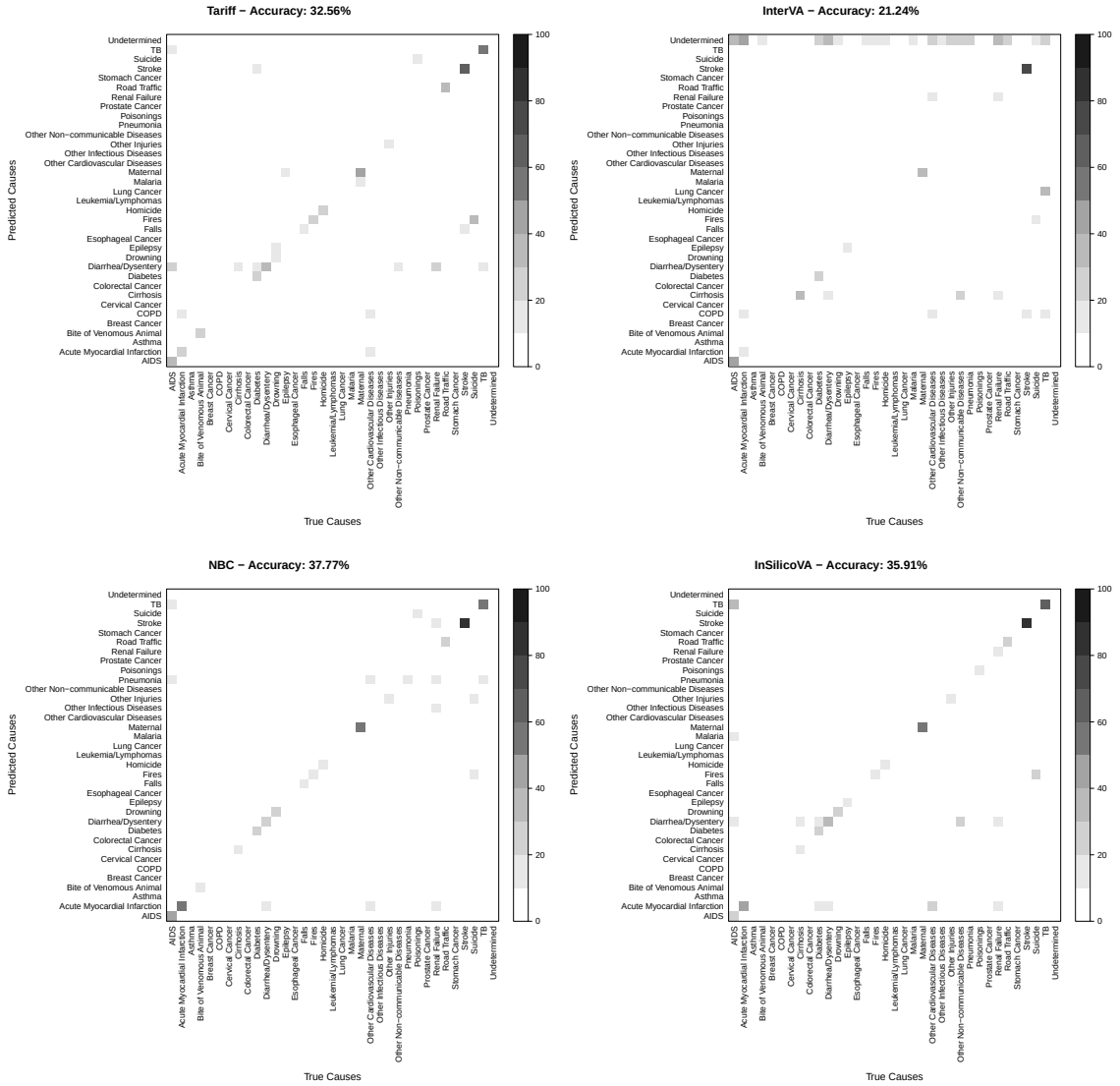

Figure 5: Comparing most likely COD assignments with the truth for the four methods

Another commonly used metric of individual COD assignment accuracy is the chance-corrected concordance (CCC). This metric can be defined as follows 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 et al. (2011b)

```
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)
}
cod.true <- test[, "gs_text34"]
ccc.tariff2 <- getCCC(cod.tariff2[,2], cod.true)
ccc.interva2 <- getCCC(cod.interva2[,2], cod.true)
ccc.nbc2 <- getCCC(cod.nbc2[,2], cod.true)
ccc.insilico2 <- getCCC(cod.insilico2[,2], cod.true)
c(ccc.tariff2, ccc.interva2, ccc.nbc2, ccc.insilico2)
```

```
## [1] 0.2833 0.1462 0.3223 0.3087
```

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

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

```
## Tariff fitted top 5 causes for death ID: 6288
##
##  Rank Cause
##  1    Stomach Cancer
##  2    Colorectal Cancer
##  3    Fires
##  4    Esophageal Cancer
##  5    Cervical Cancer
```

This can be done similarly for other methods too:

```
summary(interva2, id = "6288")
```

```
## InterVA-4 fitted top 5 causes for death ID: 6288
##
##  Cause                          Likelihood
##  COPD                           0.2846
##  Stroke                         0.2191
##  Pneumonia                      0.1651
```

```
##  Other Cardiovascular Diseases 0.1432
##  Diabetes                      0.0613
```

```
summary(nbc2, id = "6288")
```

```
## Naive Bayes Classifier (NBC) fitted on 6287 deaths
##
##  Top 5 causes by probabilities:
##                                Probability
## Stroke                           0.5844687
## Pneumonia                        0.3372839
## Other.Infectious.Diseases        0.0703078
## COPD                             0.0060602
## Other.Cardiovascular.Diseases    0.0007386
```

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

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

```
## InSilicoVA fitted top  causes for death ID: 6288
## Credible intervals shown: %
##                                  Mean Lower Median Upper
## Pneumonia                   0.4666968    NA     NA    NA
## Stroke                      0.4578644    NA     NA    NA
## Other Infectious Diseases   0.0588422    NA     NA    NA
## Epilepsy                    0.0086214    NA     NA    NA
## COPD                        0.0054392    NA     NA    NA
## Malaria                     0.0006427    NA     NA    NA
## Diabetes                    0.0004755    NA     NA    NA
## Acute Myocardial Infarction 0.0004039    NA     NA    NA
## Falls                       0.0003250    NA     NA    NA
## Renal Failure               0.0002995    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 measurement, we can either run the chain with the additional argument `indiv.CI = 0.95`,

```
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,

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

## Calculating individual COD distributions...
```

And then the summary method will provide the credible intervals

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

## InSilicoVA fitted top  causes for death ID: 6288
## Credible intervals shown: 95%
##                                  Mean     Lower    Median     Upper
## Pneumonia                   0.4666968 0.3312673 0.4675548 0.6003460
## Stroke                      0.4578644 0.3350943 0.4567035 0.5835298
## Other Infectious Diseases   0.0588422 0.0362410 0.0584487 0.0876825
## Epilepsy                    0.0086214 0.0054236 0.0084612 0.0149769
## COPD                        0.0054392 0.0032738 0.0053473 0.0078578
## Malaria                     0.0006427 0.0004008 0.0006368 0.0009619
## Diabetes                    0.0004755 0.0002485 0.0004535 0.0008069
## Acute Myocardial Infarction 0.0004039 0.0002771 0.0004023 0.0005510
## Falls                       0.0003250 0.0001386 0.0003291 0.0005572
## Renal Failure               0.0002995 0.0001931 0.0002945 0.0004388
```

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.

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

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

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

## [1] 1554   34

# 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

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

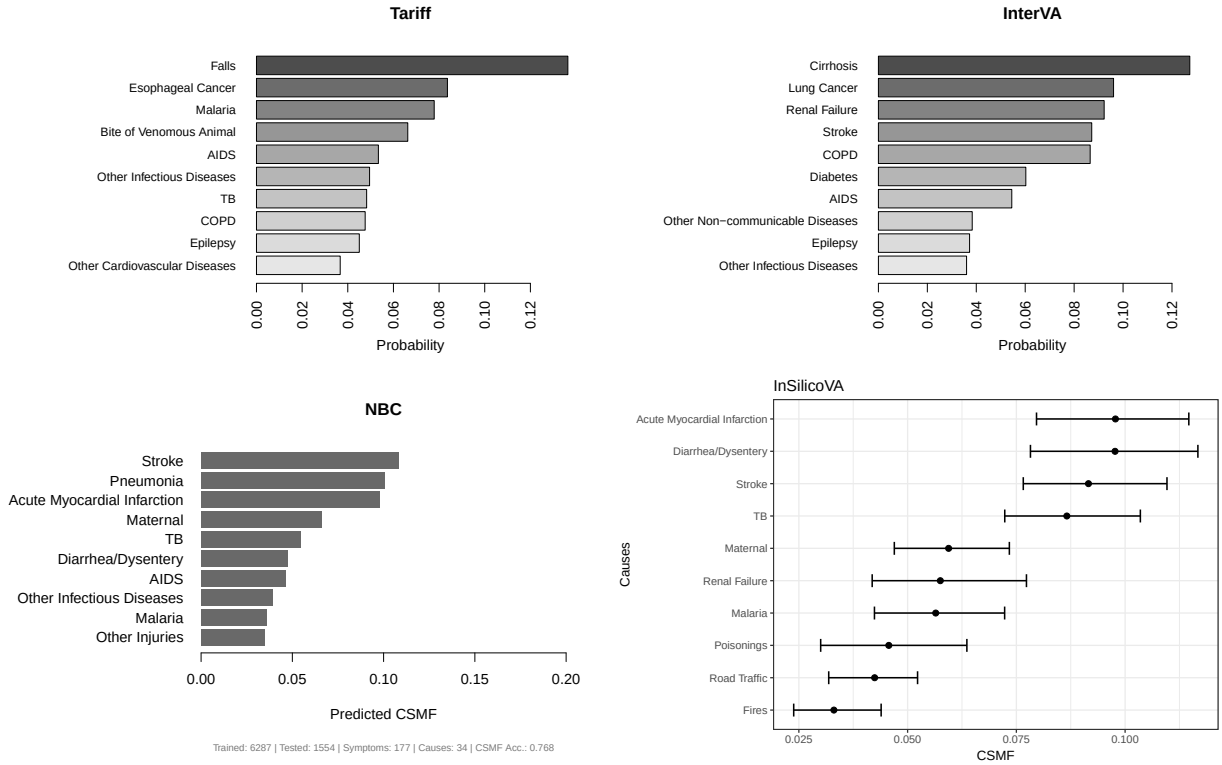

Figure 6: Plot of top 10 CSMFs for each method using plotVA() function.

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

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

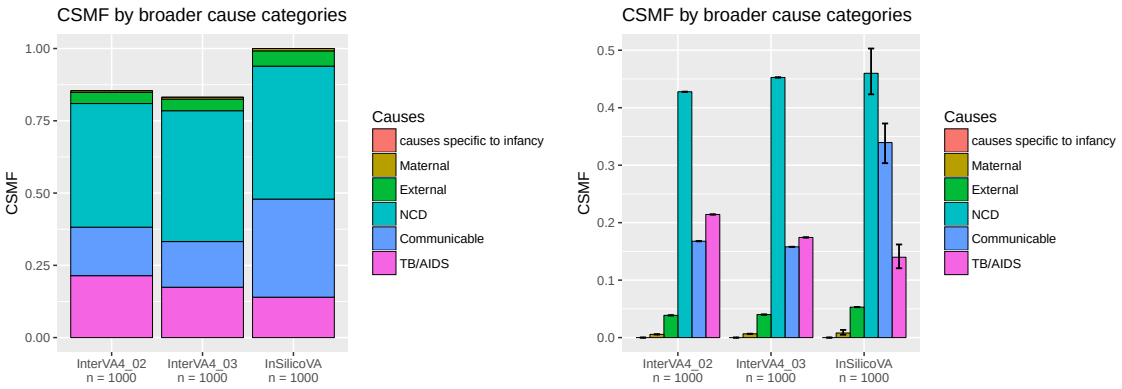

Figure 7: Comparing aggregated CSMF for InterVA-4 and InSilicoVA.

One may notice the missing proportion from the above plot due to the "undetermined" category. It can be easily added to the plot by editing the grouping rule. We can easily add another row to the grouping matrix:

```
# also show changing group here
data(SampleCategory)
head(SampleCategory)

##                              InterVA    Physician
## 1           Sepsis (non-obstetric) Communicable
## 2 Acute resp infect incl pneumonia Communicable
## 3           HIV/AIDS related death      TB/AIDS
## 4              Diarrhoeal diseases Communicable
## 5                          Malaria Communicable
## 6                          Measles Communicable

grouping <- SampleCategory
grouping[,1] <- as.character(grouping[,1])
grouping <- rbind(grouping, c("Undetermined", "Undetermined"))
tail(grouping)

##                                 InterVA    Physician
## 56                  Obstructed labour     Maternal
## 57           Pregnancy-related sepsis     Maternal
## 58               Anaemia of pregnancy     Maternal
## 59                  Ruptured uterus     Maternal
## 60 Other and unspecified maternal CoD     Maternal
## 61                      Undetermined Undetermined

stackplotVA(compare, sample.size.print = TRUE,
            xlab = "", angle = 0,
            grouping = grouping)
```

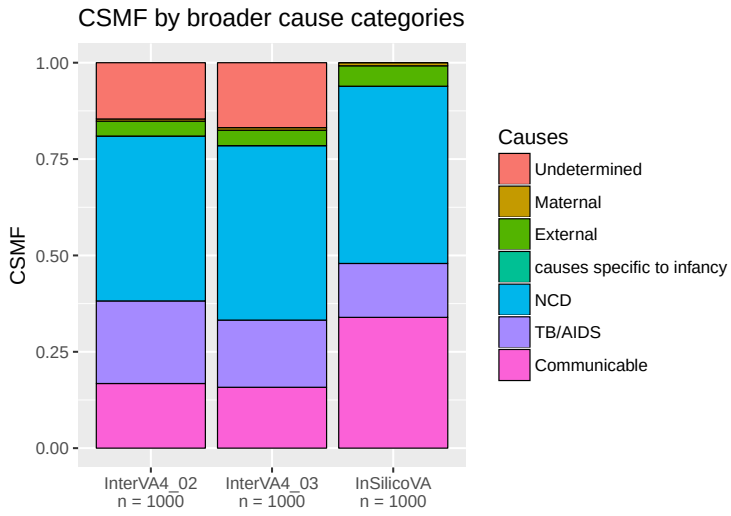

Figure 8: Comparing aggregated CSMF for InterVA-4 and InSilicoVA, adding undetermined category.

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

```
order.group <- c("TB/AIDS",  "Communicable", "NCD",
            "External", "Maternal",
            "causes specific to infancy",
            "Undetermined")
stackplotVA(compare, sample.size.print = TRUE,
            xlab = "", angle = 0,
            grouping = grouping, order.group = order.group)
```

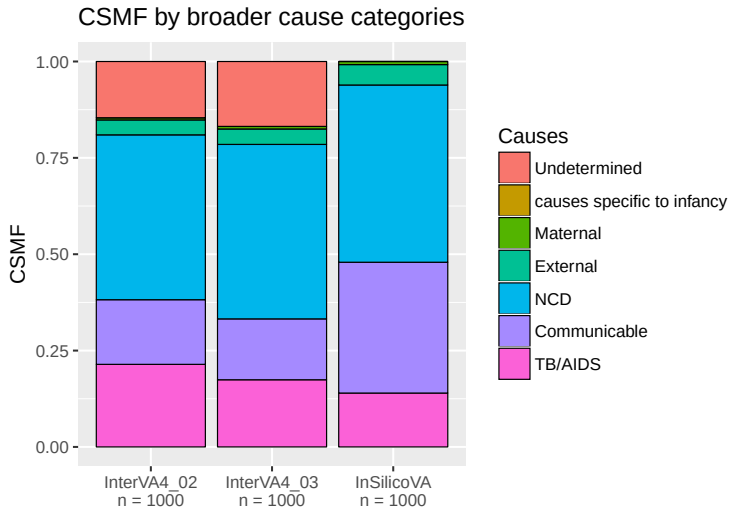

Figure 9: Comparing aggregated CSMF for InterVA-4 and InSilicoVA, with the causes reordered.

Similar plots can be made for other cause lists as long as a grouping rule is provided. For example, with PHMRC data,

```
cod.phmrc <- c(unique(as.character(train[, "gs_text34"])),
               "Undetermined")
cod.phmrc
```

```
##  [1] "Cirrhosis"                       "COPD"
##  [3] "Acute Myocardial Infarction"     "Fires"
##  [5] "Renal Failure"                   "AIDS"
##  [7] "Lung Cancer"                     "Maternal"
##  [9] "Drowning"                        "Other Non-communicable Diseases"
## [11] "Falls"                           "Stroke"
## [13] "Road Traffic"                    "Diabetes"
## [15] "Other Cardiovascular Diseases"   "Other Infectious Diseases"
## [17] "Pneumonia"                       "Suicide"
## [19] "Cervical Cancer"                 "TB"
## [21] "Malaria"                         "Asthma"
## [23] "Diarrhea/Dysentery"              "Colorectal Cancer"
## [25] "Homicide"                        "Breast Cancer"
## [27] "Leukemia/Lymphomas"              "Other Injuries"
## [29] "Poisonings"                      "Epilepsy"
## [31] "Prostate Cancer"                 "Esophageal Cancer"
## [33] "Stomach Cancer"                  "Bite of Venomous Animal"
## [35] "Undetermined"
```

```
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")
grouping2 <- cbind(cod.phmrc, group.phmrc)
compare2 <- list(Tariff = tariff2,
                 InterVA = interva2,
                 NBC = nbc2,
                 InSilicoVA = insilico2)
stackplotVA(compare2, sample.size.print = TRUE,
            xlab = "", angle = 0,
            grouping = grouping2)
```

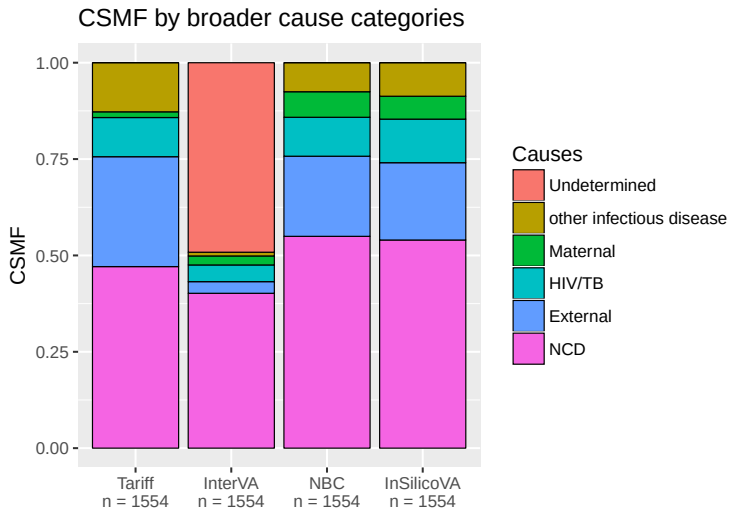

Figure 10: Comparing aggregated CSMF for InterVA-4 and InSilicoVA, with the causes reordered.

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 decomposition when 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 such 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.

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 section, 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.

```
data(RandomVA2)
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.

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

```
## Performing data consistency check...
## Data check finished.
## Calculating individual COD distributions...
```

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

```
## Performing data consistency check...
## Data check finished.
## Not all causes with CSMF > 0.02 are convergent.
##  Please check using csmf.diag() for more information.
## Calculating individual COD distributions...
```

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

```
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  Upper
## Other and unspecified infect dis 0.2558    0.0217 0.2158 0.2551 0.2992
## Renal failure                    0.0885    0.0153 0.0597 0.0876 0.1167
## HIV/AIDS related death           0.0851    0.0148 0.0581 0.0854 0.1142
## Digestive neoplasms              0.0533    0.0122 0.0332 0.0524 0.0793
## Other and unspecified neoplasms  0.0513    0.0117 0.0297 0.0509 0.0736
##
## Women - Top 5 CSMFs:
##                                     Mean Std.Error  Lower Median  Upper
## Other and unspecified infect dis  0.2739    0.0209 0.2319 0.2748 0.3160
## Renal failure                     0.1118    0.0152 0.0816 0.1111 0.1441
## HIV/AIDS related death            0.1083    0.0136 0.0837 0.1073 0.1345
## Other and unspecified cardiac dis 0.0722    0.0123 0.0509 0.0710 0.1010
## Other and unspecified neoplasms   0.0678    0.0124 0.0462 0.0669 0.0949
```

```
summary(fit_ins2, 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 60+   Men 60- Women 60+ Women 60-
##       187       273       305       235
```

```
##
## Men 60+ - Top 5 CSMFs:
##                                    Mean Std.Error  Lower Median  Upper
## Other and unspecified infect dis  0.2280    0.0347 0.1590 0.2269 0.2971
## Renal failure                     0.1059    0.0275 0.0535 0.1046 0.1633
## Other and unspecified cardiac dis 0.0966    0.0235 0.0558 0.0956 0.1514
## Respiratory neoplasms             0.0830    0.0222 0.0415 0.0822 0.1316
## Stroke                            0.0819    0.0206 0.0478 0.0807 0.1271
##
## Men 60- - Top 5 CSMFs:
##                                    Mean Std.Error  Lower Median  Upper
## Other and unspecified infect dis 0.2755    0.0284 0.2179 0.2747 0.3289
## HIV/AIDS related death           0.1414    0.0247 0.0951 0.1411 0.1910
## Renal failure                    0.0759    0.0183 0.0450 0.0739 0.1131
## Acute resp infect incl pneumonia 0.0658    0.0148 0.0392 0.0653 0.0994
## Digestive neoplasms              0.0649    0.0157 0.0371 0.0648 0.0966
##
## Women 60+ - Top 5 CSMFs:
##                                    Mean Std.Error  Lower Median  Upper
## Other and unspecified infect dis  0.2634    0.0307 0.2026 0.2628 0.3294
## Renal failure                     0.1925    0.0252 0.1472 0.1913 0.2467
## Other and unspecified cardiac dis 0.1177    0.0223 0.0789 0.1161 0.1674
## Other and unspecified neoplasms   0.0727    0.0172 0.0437 0.0701 0.1065
## Other and unspecified NCD         0.0663    0.0185 0.0368 0.0647 0.1044
##
## Women 60- - Top 5 CSMFs:
##                                    Mean Std.Error  Lower Median  Upper
## HIV/AIDS related death           0.2838    0.0314 0.2165 0.2861 0.3421
## Other and unspecified infect dis 0.2534    0.0323 0.1945 0.2529 0.3206
## Digestive neoplasms              0.0707    0.0175 0.0384 0.0698 0.1083
## Pulmonary tuberculosis           0.0598    0.0137 0.0368 0.0589 0.0883
## Acute resp infect incl pneumonia 0.0499    0.0157 0.0197 0.0484 0.0834
```

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.

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

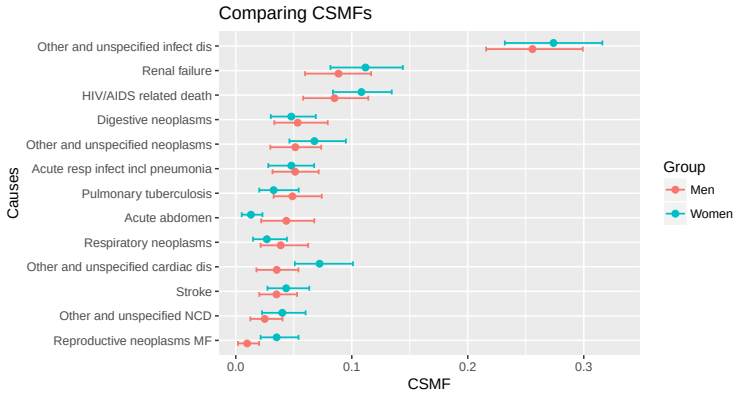

Figure 11: Comparing CSMF for different sub-population.

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.

```
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"))
```

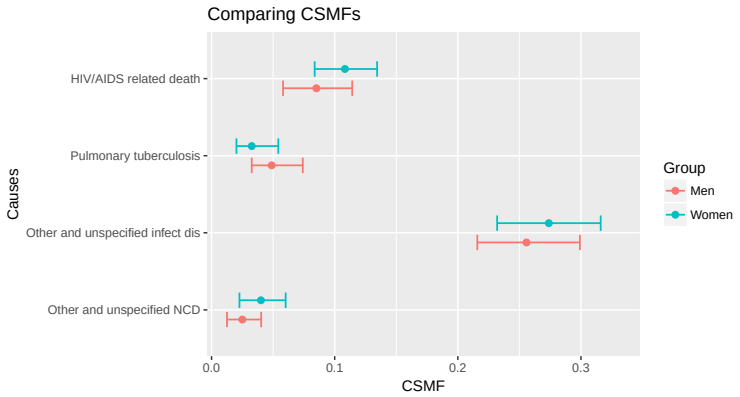

Figure 12: Comparing CSMF for different sub-population in selected CODs.

We can also plot each sub-population alone.

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

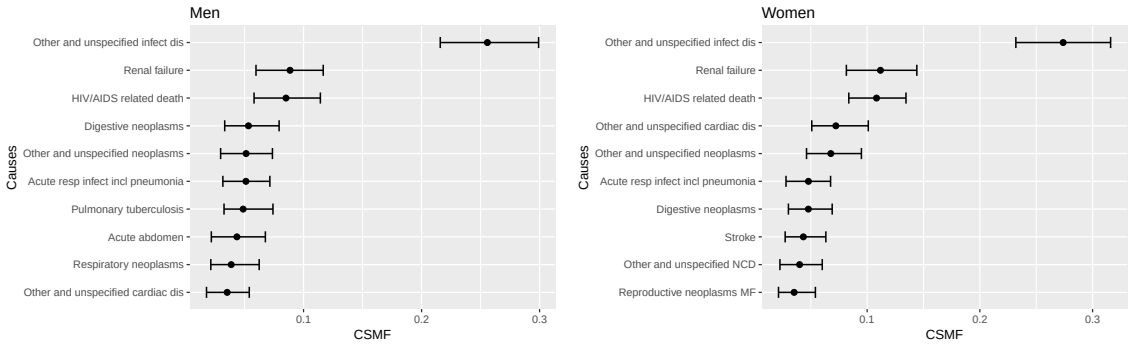

Figure 13: Comparing CSMF for different sub-population in separate plots.

This also applies to the stack plot discussed before.

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

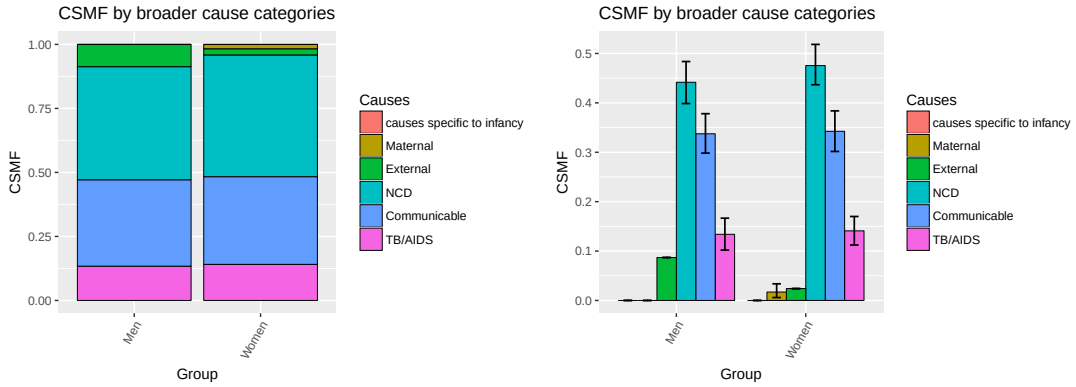

Figure 14: Comparing aggregated CSMF for two different sub-population.

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

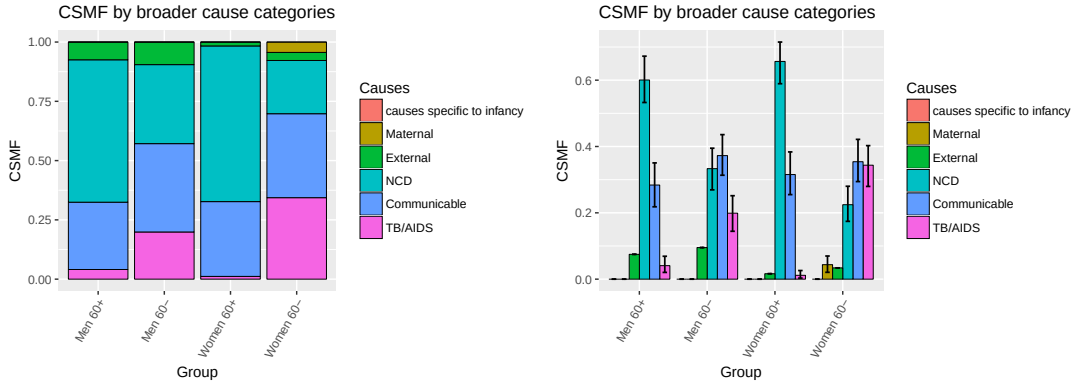

Figure 15: Comparing aggregated CSMF for four different sub-population.

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`.

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

## No groups specified, aggregate for all deaths
## 1000 deaths matched from data to the fitted object
## Aggregating individual COD distributions...

head(agg.csmf)

##                                      Mean    Median     Lower     Upper
## Sepsis (non-obstetric)           1.514e-04 1.381e-04 9.036e-05 2.693e-04
## Acute resp infect incl pneumonia 4.990e-02 4.980e-02 4.620e-02 5.485e-02
## HIV/AIDS related death           9.984e-02 9.985e-02 9.624e-02 1.031e-01
```

```
## Diarrhoeal diseases                5.767e-03 5.747e-03 4.810e-03 6.895e-03
## Malaria                            6.744e-04 6.287e-04 1.365e-04 1.439e-03
## Measles                            6.961e-09 2.014e-09 1.687e-10 5.697e-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.

```
agg.by.sex.age <- get.indiv(data = RandomVA2, insilico1, CI = 0.95,
                            is.aggregate = TRUE, by = list("sex", "age"))

## 1000 deaths matched from data to the fitted object
## Aggregating individual COD distributions...

head(agg.by.sex.age$mean)

##                                      Men 60+   Men 60- Women 60+ Women 60-
## Sepsis (non-obstetric)           1.627e-04 3.131e-04 4.448e-05 9.329e-05
## Acute resp infect incl pneumonia 3.755e-02 6.281e-02 4.441e-02 5.186e-02
## HIV/AIDS related death           1.340e-02 1.405e-01 2.447e-03 2.477e-01
## Diarrhoeal diseases              1.908e-04 3.716e-03 3.385e-03 1.568e-02
## Malaria                          9.600e-04 5.222e-04 9.235e-04 3.005e-04
## Measles                          4.595e-10 8.067e-09 2.148e-09 1.709e-08
```

We can visualize these aggregated distributions with

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

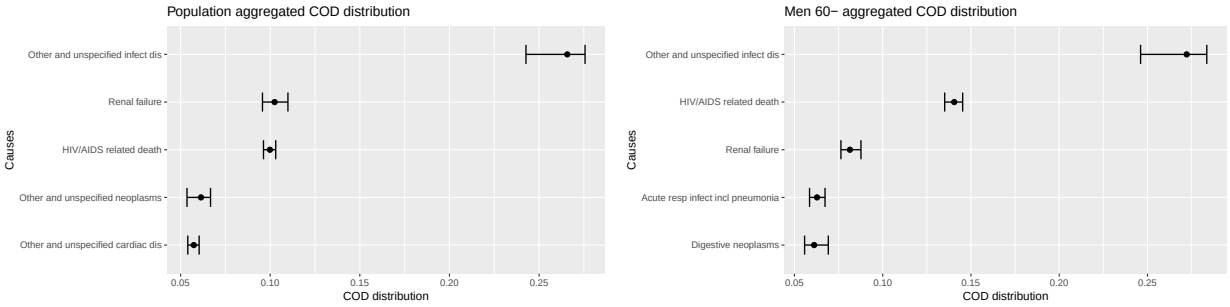

Figure 16: Aggregated COD distribution for all data or a subset of data

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

```
indivplot(agg.by.sex.age, which.plot = list("Men 60+", "Men 60-"),
          top = 5, title = "Top CODs: Men 60+ v.s. Men 60-")
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-")
```

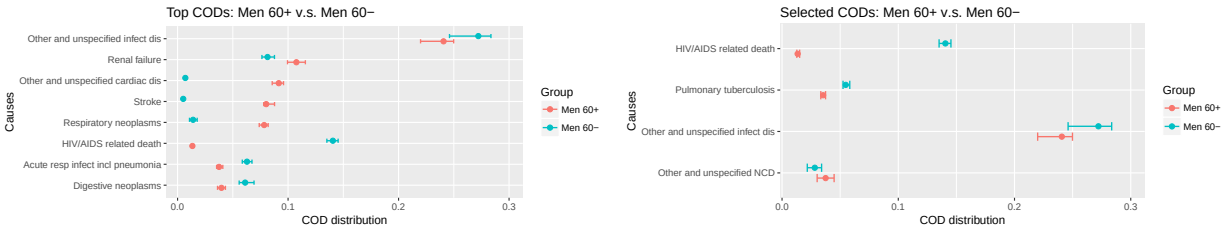

Figure 17: Comparing aggregated COD distribution for different subsets of data

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.

```
data(SampleCategory)
data(RandomPhysician)
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.

```
doctors <- paste0("doc", c(1:15))
causelist <- c("Communicable", "TB/AIDS", "Maternal",
               "NCD", "External", "Unknown")
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.

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

## Performing data consistency check...
## Data check finished.
## 941 deaths found known physician coding after removing deaths from external causes.

summary(fit_ins_phy)

## 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
##
## Top 10 CSMFs:
##                                    Mean Std.Error  Lower Median  Upper
## HIV/AIDS related death           0.2416    0.0129 0.2176 0.2415 0.2665
## Other and unspecified infect dis 0.1579    0.0129 0.1345 0.1578 0.1828
## Renal failure                    0.1079    0.0111 0.0880 0.1076 0.1318
## Other and unspecified cardiac dis 0.0648   0.0080 0.0503 0.0641 0.0817
## Pulmonary tuberculosis           0.0581    0.0074 0.0454 0.0577 0.0725
## Other and unspecified neoplasms  0.0531    0.0080 0.0382 0.0529 0.0683
## Stroke                           0.0369    0.0063 0.0261 0.0366 0.0500
## Respiratory neoplasms            0.0332    0.0060 0.0228 0.0327 0.0452
## Digestive neoplasms              0.0329    0.0061 0.0225 0.0323 0.0460
## Other and unspecified NCD        0.0295    0.0072 0.0165 0.0287 0.0460
```

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

```
plotVA(insilico1)
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

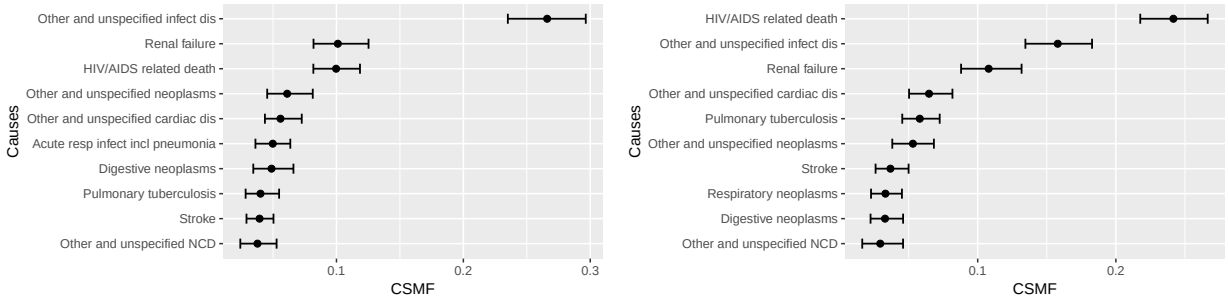

Figure 18: Comparing fitted CSMF with and without physicians

allows researchers to 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

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

Carl E Taylor, RL Parker, WA Reinke, R Faruquee, et al. *Child and maternal health services in rural India. The Narangwal experiment. 2. Integrated family planning and health care.* Johns Hopkins University Press, 1983.

World Health Organization and others. 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, accessed 2015-01. 2012.

Kathleen Kahn, Mark A Collinson, F Xavier Gómez-Olivé, Obed Mokoena, Rhian Twine, Paul Mee, Sulaimon A Afolabi, Benjamin D Clark, Chodziwadziwa W Kabudula, Audrey Khosa, et al. Profile: Agincourt health and socio-demographic surveillance system. *International Journal of Epidemiology*, 41(4):988–1001, 2012.

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

Christopher JL Murray, Alan D Lopez, Robert Black, Ramesh Ahuja, Said M Ali, Abdullah Baqui, Lalit Dandona, Emily Dantzer, Vinita Das, Usha Dhingra, et al. Population health metrics research consortium gold standard verbal autopsy validation study: design, implementation, and development of analysis datasets. *Population health metrics*, 9(1):27, 2011a.

Zehang Richard Li, Tyler H McCormick, and Samuel J Clark. Interva4: An R package to analyze verbal autopsy data. 2014.

Peter Byass, Daniel Chandramohan, Samuel J Clark, Lucia D'Ambruoso, Edward Fottrell, Wendy J Graham, Abraham J Herbst, Abraham Hodgson, Sennen Hounton, Kathleen Kahn, et al. Strengthening standardised interpretation of verbal autopsy data: The new InterVA-4 tool. *Global Health Action*, 5, 2012.

Peter Byass. InterVA-4 Software [Windows Executable]. *www.interva.net*, 2015.

Pierre Miasnikof, Vasily Giannakeas, Mireille Gomes, Lukasz Aleksandrowicz, Alexander Y Shestopaloff, Dewan Alam, Stephen Tollman, Akram Samarikhalaj, and Prabhat Jha. Naive bayes classifiers for verbal autopsies: comparison to physician-based classification for 21,000 child and adult deaths. *BMC medicine*, 13(1):1, 2015.

Zehang R Li, Tyler H McCormick, and Sam J Clark. *InSilicoVA: Probabilistic Verbal Autopsy Coding with 'InSilicoVA' Algorithm*, 2016a. R package version 1.1.5.

Tyler H. McCormick, Zehang Richard Li, Clara Calvert, Amelia C. Crampin, Kathleen Kahn, and Samuel J. Clark. Probabilistic cause-of-death assignment using verbal autopsies. *Journal of the American Statistical Association*, 111(515):1036–1049, 2016. doi: 10.1080/01621459.2016.1152191. URL http://dx.doi.org/10.1080/01621459.2016.1152191.

Zehang R Li, Tyler H McCormick, and Sam J Clark. *Tariff: Replicate Tariff Method for Verbal Autopsy*, 2016b. R package version 1.0.2.

S. L. James, A. D. Flaxman, C. J. Murray, and Consortium Population Health Metrics Research. Performance of the tariff method: validation of a simple additive algorithm for analysis of verbal autopsies. *Population Health Metrics*, 9(31), 2011.

Peter Serina, Ian Riley, Andrea Stewart, Spencer L James, Abraham D Flaxman, Rafael Lozano, Bernardo Hernandez, Meghan D Mooney, Richard Luning, Robert Black, et al. Improving performance of the tariff method for assigning causes of death to verbal autopsies. *BMC medicine*, 13(1):1, 2015a.

Peter Serina, Ian Riley, Andrea Stewart, Abraham D Flaxman, Rafael Lozano, Meghan D Mooney, Richard Luning, Bernardo Hernandez, Robert Black, Ramesh Ahuja, et al. A shortened verbal autopsy instrument for use in routine mortality surveillance systems. *BMC medicine*, 13(1):1, 2015b.

Nicolas Maire. *CrossVA: Verbal Autopsy Data Transform for Use with Various Coding Algorithms*, 2017. R package version 0.9.1.

Christopher JL Murray, Rafael Lozano, Abraham D Flaxman, Alireza Vahdatpour, and Alan D Lopez. Robust metrics for assessing the performance of different verbal autopsy cause assignment methods in validation studies. *Popul Health Metr*, 9:28, 2011b.