InSilicoVA package

Richard Li

April 22, 2015

1 Install Java and rJava package

Key: 1. add

2 Install InSilicoVA package

Before it is updated to CRAN, installing the package is kind of pain. The source file is available at the following address: `https://github.com/richardli/InSilicoVA-beta/blob/master/InSilicoVA_1.0.tar.gz`. Click the "view raw" option and it will download the package .tar.gz file. After putting it in the working directory, the package could be installed using (change the file directory to where you store the file):

Note before it goes online on CRAN, you might need to install the dependencies by yourself before running the above codes using:

Now the basics of the package could be browsed by

3 InSilicoVA methodologies revisited

More later, see `http://arxiv.org/abs/1411.3042` for now.

4 Running InSilicoVA model

Here I use the Karonga data to show some quick demo of InSilicoVA package. The data input is exactly as InterVA input, except the missing data are represented as ".". The data should consist of 246 columns, as in InterVA input.

```
data <- read.csv("data/random_interva4.csv")
## use only the first 2000 deaths to illustrate in this document
data <- data[1:2000, 1:246]
```

Here we could also specify sub-populations, it is just a vector of length N (the same length as the data). It could be either strings or numeric values. The names of the sub-populations will maintain through the model. For example, we construncted this fake subpopulation:

```
N <- dim(data)[1]
subpop <- c(rep("Group1", round(N/2)), rep("Group2", N-round(N/2)))
table(subpop)

## subpop
## Group1 Group2
##   1000   1000
```

The core algorithm is performed using the function

```
fit<- insilico( data, subpop = subpop, HIV = "h", Malaria = "h",
                    length.sim = 400, burnin = 200, thin = 10 , seed = 2,
                    InterVA.prior = TRUE, external.sep = TRUE,
                    keepProbbase.level = TRUE, auto.length = FALSE,
                    java_option = "-Xmx1g")

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

A few key parameters to set here:

- `length.sim`, `burnin`, `thin`
  The length of chain is decided by these three factors. The common practice is to run the MCMC chain for K iterations and discard the first half of the chain, which means setting `length.sim = K` and `burnin = K/2`. The *thin* variables specifies the frequency of saving the chain. For instance, $thin = 10$ means all variables gets saved every 10 iterations. It is useful especially when the computer's memory is limited. In practice, a chain of length $5,000$ should be a reasonable starting value for most cases.

- `auto.length`, `conv.csmf`
  If the length of the chain is not clear ahead of time, there is the option of setting `auto.length` to be TRUE, which will keep doubling the length of the chain (at most twice to avoid too heavy RAM usage) automatically if the top causes are not convergent after the initial iterations. In practice, the rare causes will be very difficult to converge by traditional test, but as long as the their fractions are very little, say smaller than 2% in the population, the influence is minimum. Thus by setting `conv.csmf = 0.02` will tell the algorithm only to check causes with fraction higher than 0.02.

- `HIV`, `Malaria`
  The same as in InterVA.

Several more advanced parameters controlling the sampler are listed below. It is suggested to check the original paper first for better understanding of the influence.

- `alpha.scale`
  Parameter for Dirichlet model. α value used in Dirichlet prior. Larger α gives less volatile CSMF.

- `InterVA.prior`, `csmf.prior`
  `InterVA.prior` is an indicator of whether to use InterVA prior as prior on CSMF. If set to FALSE, the algorithm will assume all causes have the same distribution, unless there is a CSMF distribution provided by `csmf.prior`. If a known CSMF distribution is available, it should be entered as a vector of length 60.

- `jump.scale`
  The scale of Metropolis proposal in the Normal model. Default to be 0.1.

- `levels.prior`, `levels.strength`
  `levels.prior` is a vector of prior expectation of conditional probability levels. They do not have to be scaled. The algorithm internally calibrate the scale to the working scale through `levels.strength`. If NULL the algorithm will use InterVA table as prior. `levels.strength` is the scaling factor for the strength of prior beliefs in the conditional probability levels. Larger value constrain the posterior estimates to be closer to prior expectation. Defult value 1 scales `levels.prior` to a suggested scale that works empirically.

- `trunc.min`, `trunc.max`
  Minimum and maximum possible value for estimated conditional probability table. Default to be 0.0001 and 0.9999

Several other things controlling the algorithm implementation:

- `useProbbase`, `keepProbbase.level`
  The first indicator `useProbbase` tells the algorithm if it needs to re-estimate $\Pr(S|C)$, and the indicator `keepProbbase.level` tells if re-estimation is only performed on the $\Pr(S|C)$ table but not the whole matrix. In general, the robust way is to estimate $\Pr(S|C)$, but only estimate the table that consists of 15 levels. That is, estimating the corresponding probabilities for each of the levels: "I", "A+", "A", etc..

- `datacheck`, `warning.write`
  In InterVA, if the user enters some death with impossible symptom combinations, e.g., male with pregnant symptoms, the algorithm will correct the input following a set of rules. This step is the same as in InSilicoVA if `datacheck` is set to be TRUE, and the messages will be printed out in a txt file if warning.write is set to TRUE.

- `external.sep`
  For external causes such as traffic accident, accidental fall, suicide, etc., deciding a death to be caused by them is relatively easy and separated from other general health symptoms. Setting this indicator to TRUE will tell the algorithm to estimate those causes separately, which is usually more robust.

- `seed`
  Setting Seed is usually good practice. Since the algorithm is stochastic, there could be some randomness in the outcome. But it will produce the same outcome with the same seed in each machine.

After running the chain, you could see some summary of the chain by simply calling

```
fit
```

```
## InSilicoVA fitted object:
## 1930 death processed
## 400 iterations performed, with first 200 iterations discarded
##  20 iterations saved after thinning
## Fitted with re-estimated InterVA4 conditional probability level table
```

Or more summary of CSMFs by

```
summary(fit, top = 10)
```

```
## InSilicoVA Call:
## 1930 death processed
## 400 iterations performed, with first 200 iterations discarded
##  20 iterations saved after thinning
## Fitted with re-estimated InterVA4 conditional probability level table
## Data consistency check performed as in InterVA4
## HIV level:      h
## Malaria level:  h
## Sub population frequencies:
## Group1 Group2
##    857    875
##
## Group1 - Top 10 CSMFs:
##                                    Mean Std.Error  Lower Median  Upper
## HIV/AIDS related death           0.3936    0.0382 0.3330 0.3988 0.4636
## Acute resp infect incl pneumonia 0.1855    0.0230 0.1539 0.1801 0.2312
## Other and unspecified neoplasms  0.1106    0.0220 0.0851 0.1040 0.1507
## Pulmonary tuberculosis           0.0736    0.0363 0.0136 0.0848 0.1181
## Stroke                           0.0189    0.0052 0.0107 0.0191 0.0264
## Tetanus                          0.0181    0.0076 0.0081 0.0182 0.0315
## Reproductive neoplasms MF        0.0163    0.0044 0.0095 0.0156 0.0238
## Diabetes mellitus                0.0136    0.0089 0.0030 0.0156 0.0245
## Pregnancy-related sepsis         0.0125    0.0049 0.0038 0.0134 0.0194
```

```
## Accid fall                       0.0112    0.0116 0.0000 0.0073 0.0332
##
## Group2 - Top 10 CSMFs:
##                                    Mean Std.Error  Lower Median  Upper
## HIV/AIDS related death           0.3751    0.0209 0.3425 0.3758 0.4071
## Acute resp infect incl pneumonia 0.1554    0.0194 0.1225 0.1530 0.1863
## Pulmonary tuberculosis           0.1137    0.0135 0.0975 0.1124 0.1371
## Other and unspecified infect dis 0.1004    0.0183 0.0702 0.0967 0.1325
## Other and unspecified neoplasms  0.0776    0.0174 0.0508 0.0803 0.1116
## Diabetes mellitus                0.0326    0.0185 0.0145 0.0229 0.0638
## Other and unspecified cardiac dis 0.0177   0.0070 0.0083 0.0179 0.0281
## Accid fall                       0.0093    0.0083 0.0005 0.0073 0.0249
## Accid poisoning & noxious subs   0.0092    0.0083 0.0005 0.0073 0.0249
## Accid drowning and submersion    0.0092    0.0084 0.0000 0.0073 0.0249

plot(fit, top = 10)

## Error in plot.insilico(fit, top = 10):  More than one groups detected.  Please specify which to
plot
```

All CSMF's at each iterations, as well as the mean, could be extracted by

```
csmf.each.iteraions <- fit$csmf
# indiv <- fit$indiv
# csmf.mean <- summary(fit)$
```

5 Run with Subpopulations