Further Features of openVA

Jason Thomas

March 6th, 2018

New Slides & Files

- https://github.com/verbal-autopsy-software/Indonesia

Overview

- Morning

Overview

- Morning
    - Data Checks

Overview

- Morning
  - Data Checks
  - Technical details of InSilicoVA

Overview

- Morning
  - Data Checks
  - Technical details of InSilicoVA
  - Producing results for individuals

Overview

- Morning
  - Data Checks
  - Technical details of InSilicoVA
  - Producing results for individuals
- Afternoon

Overview

- Morning
  - Data Checks
  - Technical details of InSilicoVA
  - Producing results for individuals
- Afternoon
  - Practice with openVA

Overview

- Morning
    - Data Checks
    - Technical details of InSilicoVA
    - Producing results for individuals
- Afternoon
    - Practice with openVA
    - Using **openVA** to run InterVA5 algorithm (in yesterday's slides)

Data Checks

- InSilicoVA & InterVA perform *data consistency checks* to ensure the symptoms do not suggest conflicting information (2 passes through the data). For example:

Data Checks

- InSilicoVA & InterVA perform *data consistency checks* to ensure the symptoms do not suggest conflicting information (2 passes through the data). For example:
  - *age In Days* 14

Data Checks

- InSilicoVA & InterVA perform *data consistency checks* to ensure the symptoms do not suggest conflicting information (2 passes through the data). For example:
  - *age In Days* 14
  - *How long did (s)he have a cough?* 4 weeks

Data Checks

- InSilicoVA & InterVA perform *data consistency checks* to ensure the symptoms do not suggest conflicting information (2 passes through the data). For example:
    - *age In Days* 14
    - *How long did (s)he have a cough?* 4 weeks
- The software will change these data, but only on a copy of the data (not your data frame that you pass as an argument to `codeVA`)

Data Checks

- InSilicoVA & InterVA perform *data consistency checks* to ensure the symptoms do not suggest conflicting information (2 passes through the data). For example:
  - *age In Days* 14
  - *How long did (s)he have a cough?* 4 weeks
- The software will change these data, but only on a copy of the data (not your data frame that you pass as an argument to `codeVA`)
- These changes are described in the log file errorlog_insilico.txt with the argument

Data Checks

- InSilicoVA & InterVA perform *data consistency checks* to ensure the symptoms do not suggest conflicting information (2 passes through the data). For example:
  - *age In Days* 14
  - *How long did (s)he have a cough?* 4 weeks
- The software will change these data, but only on a copy of the data (not your data frame that you pass as an argument to `codeVA`)
- These changes are described in the log file errorlog_insilico.txt with the argument

Data Checks

- InSilicoVA & InterVA perform *data consistency checks* to ensure the symptoms do not suggest conflicting information (2 passes through the data). For example:
  - *age In Days* 14
  - *How long did (s)he have a cough?* 4 weeks
- The software will change these data, but only on a copy of the data (not your data frame that you pass as an argument to `codeVA`)
- These changes are described in the log file errorlog_insilico.txt with the argument

```
results1 <- codeVA(data = data1, data.type = "WHO2016",
    model = "InSilicoVA", warning.write = TRUE)
```

- Info in Error log: (1) record ID; (2) index symptom; and (3) don't ask / ask if / neonate symptom

Data Checks: Three types of consistency checks

1. **Don't ask:** if the index symptom is inconsistent with a "don't ask" symptom, then the index symptom is set to missing

Data Checks: Three types of consistency checks

1. **Don’t ask:** if the index symptom is inconsistent with a “don’t ask” symptom, then the index symptom is set to missing
    - *Error log*: ID – index symptom – don’t ask symptom – "cleared in working information"

Data Checks: Three types of consistency checks

1. **Don’t ask:** if the index symptom is inconsistent with a “don’t ask” symptom, then the index symptom is set to missing
   - *Error log*: ID – index symptom – don’t ask symptom – "cleared in working information"
   - *inconsistent* index symptom $==$ *substantive* value (as shown in the `subst` column of probbase.xls – the InterVA5 SCI) and “don’t ask” symptom $==$ last character in `dontask` column

Data Checks: Three types of consistency checks

1. **Don't ask:** if the index symptom is inconsistent with a "don't ask" symptom, then the index symptom is set to missing
   - *Error log*: ID – index symptom – don't ask symptom – "cleared in working information"
   - *inconsistent* index symptom $==$ *substantive* value (as shown in the `subst` column of probbase.xls – the InterVA5 SCI) and "don't ask" symptom $==$ last character in `dontask` column
   - both symptoms need to have non-missing values

Data Checks: Three types of consistency checks

1. **Don't ask:** if the index symptom is inconsistent with a "don't ask" symptom, then the index symptom is set to missing
   - *Error log*: ID – index symptom – don't ask symptom – "cleared in working information"
   - *inconsistent* index symptom == *substantive* value (as shown in the `subst` column of probbase.xls – the InterVA5 SCI) and "don't ask" symptom == last character in `dontask` column
   - both symptoms need to have non-missing values

2. **Ask if:** if the index symptom == *substantive* value and the "ask if" symptom != last character in `askif` column, then assign "ask if" symptom to the last character in `askif` column

Data Checks: Three types of consistency checks

1. **Don't ask:** if the index symptom is inconsistent with a "don't ask" symptom, then the index symptom is set to missing
   - *Error log*: ID – index symptom – don't ask symptom – "cleared in working information"
   - *inconsistent* index symptom == *substantive* value (as shown in the `subst` column of probbase.xls – the InterVA5 SCI) and "don't ask" symptom == last character in `dontask` column
   - both symptoms need to have non-missing values

2. **Ask if:** if the index symptom == *substantive* value and the "ask if" symptom != last character in `askif` column, then assign "ask if" symptom to the last character in `askif` column
   - *Error log*: ID – index symptom – ask if symptom – "updated in working information"

Data Checks: Three types of consistency checks

1. **Don't ask:** if the index symptom is inconsistent with a "don't ask" symptom, then the index symptom is set to missing
   - *Error log*: ID – index symptom – don't ask symptom – "cleared in working information"
   - *inconsistent* index symptom $==$ *substantive* value (as shown in the `subst` column of probbase.xls – the InterVA5 SCI) and "don't ask" symptom $==$ last character in `dontask` column
   - both symptoms need to have non-missing values

2. **Ask if:** if the index symptom $==$ *substantive* value and the "ask if" symptom $!=$ last character in `askif` column, then assign "ask if" symptom to the last character in `askif` column
   - *Error log*: ID – index symptom – ask if symptom – "updated in working information"
   - index symptom need to have non-missing value

Data Checks: Three types of consistency checks

3. **Neonate Only:** if the index symptom $==$ *substantive* value and the deceased was NOT, then the index symptom is set to missing

Data Checks: Three types of consistency checks

3. **Neonate Only:** if the index symptom $==$ *substantive* value and the deceased was NOT, then the index symptom is set to missing
    - *Error log*: ID – index symptom – don't ask symptom – "only required for neonates - cleared in working information"

Data Checks: Three types of consistency checks

3. **Neonate Only:** if the index symptom $==$ *substantive* value and the deceased was NOT, then the index symptom is set to missing
    - *Error log*: ID – index symptom – don't ask symptom – "only required for neonates - cleared in working information"
    - index symptom need to have non-missing value

Data Checks: Three types of consistency checks

3. **Neonate Only:** if the index symptom $==$ *substantive* value and the deceased was NOT, then the index symptom is set to missing
   - *Error log*: ID – index symptom – don't ask symptom – "only required for neonates - cleared in working information"
   - index symptom need to have non-missing value

- data consistency checks performed in this order

Data Checks: Three types of consistency checks

3. **Neonate Only:** if the index symptom == *substantive* value and the deceased was NOT, then the index symptom is set to missing
   - *Error log*: ID – index symptom – don't ask symptom – "only required for neonates - cleared in working information"
   - index symptom need to have non-missing value

- data consistency checks performed in this order
  1. don't ask

Data Checks: Three types of consistency checks

3. **Neonate Only:** if the index symptom $==$ *substantive* value and the deceased was NOT, then the index symptom is set to missing
   - *Error log*: ID – index symptom – don't ask symptom – "only required for neonates - cleared in working information"
   - index symptom need to have non-missing value

- data consistency checks performed in this order
  1. don't ask
  2. ask if

Data Checks: Three types of consistency checks

3. **Neonate Only:** if the index symptom $==$ *substantive* value and the deceased was NOT, then the index symptom is set to missing
   - *Error log*: ID – index symptom – don't ask symptom – "only required for neonates - cleared in working information"
   - index symptom need to have non-missing value

- data consistency checks performed in this order
  1. don't ask
  2. ask if
  3. neonates only

Data Checks: Three types of consistency checks

3. **Neonate Only:** if the index symptom == *substantive* value and the deceased was NOT, then the index symptom is set to missing
    - *Error log*: ID – index symptom – don't ask symptom – "only required for neonates - cleared in working information"
    - index symptom need to have non-missing value

- data consistency checks performed in this order
    1. don't ask
    2. ask if
    3. neonates only
    - (then this is repeated with a second pass through the data)

InSilicoVA and CSMF

- This is a very brief discussion of some of the more technical questions about InSilicoVA.

InSilicoVA and CSMF

- This is a very brief discussion of some of the more technical questions about InSilicoVA.
- Perhaps the most conceptually different aspect of InSilicoVA, compared to the other algorithms, is how CSMF is viewed.

InSilicoVA and CSMF

- This is a very brief discussion of some of the more technical questions about InSilicoVA.
- Perhaps the most conceptually different aspect of InSilicoVA, compared to the other algorithms, is how CSMF is viewed.
- Let us first look at how CSMF is calculated in other algorithms

InSilicoVA and CSMF

- This is a very brief discussion of some of the more technical questions about InSilicoVA.
- Perhaps the most conceptually different aspect of InSilicoVA, compared to the other algorithms, is how CSMF is viewed.
- Let us first look at how CSMF is calculated in other algorithms
    - InterVA: take **up to top three** causes and aggregate their probabilities.

InSilicoVA and CSMF

- This is a very brief discussion of some of the more technical questions about InSilicoVA.
- Perhaps the most conceptually different aspect of InSilicoVA, compared to the other algorithms, is how CSMF is viewed.
- Let us first look at how CSMF is calculated in other algorithms
  - InterVA: take **up to top three** causes and aggregate their probabilities.
  - NBC: take the average of the full **individual probabilities**.

InSilicoVA and CSMF

- This is a very brief discussion of some of the more technical questions about InSilicoVA.
- Perhaps the most conceptually different aspect of InSilicoVA, compared to the other algorithms, is how CSMF is viewed.
- Let us first look at how CSMF is calculated in other algorithms
  - InterVA: take **up to top three** causes and aggregate their probabilities.
  - NBC: take the average of the full **individual probabilities**.
- The common theme is the CSMF can be directly derived from individual results. However, InSilicoVA parameterize CSMF as a separate set of parameters to be learned from the data.

Population estimates

- To see what it means, consider two datasets, both randomly sampled from a large population:

Population estimates

- To see what it means, consider two datasets, both randomly sampled from a large population:
  - a small dataset with 100 observations

Population estimates

- To see what it means, consider two datasets, both randomly sampled from a large population:
  - a small dataset with 100 observations
  - a large dataset by exactly repeating the small dataset 10 times.

Population estimates

- To see what it means, consider two datasets, both randomly sampled from a large population:
  - a small dataset with 100 observations
  - a large dataset by exactly repeating the small dataset 10 times.
- For any deterministic algorithm, the distribution of causes in both datasets should match exactly.

Population estimates

- To see what it means, consider two datasets, both randomly sampled from a large population:
    - a small dataset with 100 observations
    - a large dataset by exactly repeating the small dataset 10 times.
- For any deterministic algorithm, the distribution of causes in both datasets should match exactly.
- But knowing there are more data may change what we believe about the unknown population: we may be more certain about our estimators.

Population estimates

- To see what it means, consider two datasets, both randomly sampled from a large population:
  - a small dataset with 100 observations
  - a large dataset by exactly repeating the small dataset 10 times.
- For any deterministic algorithm, the distribution of causes in both datasets should match exactly.
- But knowing there are more data may change what we believe about the unknown population: we may be more certain about our estimators.
- Essentially, this is the idea behind the InSilicoVA logic: our observations are samples from a larger population, and CSMF measures the distribution of causes in that population.

Fine tuning InSilicoVA

- The stochastic nature of the sampling-based approach adopted by InSilicoVA makes it flexible with nice characterization of uncertainties.

Fine tuning InSilicoVA

- The stochastic nature of the sampling-based approach adopted by InSilicoVA makes it flexible with nice characterization of uncertainties.
- But it also means the algorithm may need to be tunned with more care.

Fine tuning InSilicoVA

- The stochastic nature of the sampling-based approach adopted by InSilicoVA makes it flexible with nice characterization of uncertainties.
- But it also means the algorithm may need to be tunned with more care.
- First, the convergence depends on how long the algorithm is run

Fine tuning InSilicoVA

- The stochastic nature of the sampling-based approach adopted by InSilicoVA makes it flexible with nice characterization of uncertainties.
- But it also means the algorithm may need to be tunned with more care.
- First, the convergence depends on how long the algorithm is run
    - `Nsim`: The total number of iterations to run the algorithm.

Fine tuning InSilicoVA

- The stochastic nature of the sampling-based approach adopted by InSilicoVA makes it flexible with nice characterization of uncertainties.
- But it also means the algorithm may need to be tunned with more care.
- First, the convergence depends on how long the algorithm is run
  - `Nsim`: The total number of iterations to run the algorithm.
  - `auto.length`: Whether or not to automatic double the number of iterations at the end if convergence test fails.

Fine tuning InSilicoVA: Example 1

```
out <- codeVA(data = RandomVA5[1:25,], data.type = "WHO2016",
              Nsim = 100, auto.length = FALSE)

InSilico Sampler initiated, 100 iterations to sample ......

Iteration: 50
Sub-population 0 acceptance ratio: 0.72
0.00min elapsed, 0.00min remaining
....
Overall acceptance ratio
Sub-population 0 : 0.7300
Organizing output, might take a moment...
Not all causes with CSMF > 0.02 are convergent.
 Please check using csmf.diag() for more information.
```

Fine tuning InSilicoVA: Example

```
csmf.diag(out, conv.csmf = 0.01)
                                   Halfwidth Mean   Halfwidth
                                   test
Measles                            failed    0.0762 0.018977
Severe malnutrition                failed    0.0678 0.023181
Other and unspecified infect dis   passed    0.0533 0.002164
Renal failure                      failed    0.0497 0.010359
Pertussis                          failed    0.0350 0.005081
Pulmonary tuberculosis             failed    0.0437 0.008504
Haemorrhagic fever (non-dengue)    failed    0.0436 0.005170
Diabetes mellitus                  failed    0.0384 0.009008
Congenital malformation            failed    0.0374 0.012316
Pregnancy-related sepsis           failed    0.0309 0.005913
Anaemia of pregnancy               failed    0.0308 0.004454
Diarrhoeal diseases                failed    0.0293 0.008432
Liver cirrhosis                    failed    0.0289 0.004361
Other and unspecified maternal CoD failed    0.0263 0.005518
```

Fine tuning InSilicoVA (cont.)

Convergence also depends on how many proposed new parameters are accepted.

- This is directly reflected in `jump.scale` and the **‘acceptance rate’** printed to the screen when running InSilicoVA.

Fine tuning InSilicoVA (cont.)

Convergence also depends on how many proposed new parameters are accepted.

- This is directly reflected in `jump.scale` and the **‘acceptance rate’** printed to the screen when running InSilicoVA.
- If `jump.scale` is too large, at each iteration, the algorithm ‘tries’ more wild guesses, leading to many of such guesses rejected. This can waste many iterations of sampling.

Fine tuning InSilicoVA (cont.)

Convergence also depends on how many proposed new parameters are accepted.

- This is directly reflected in `jump.scale` and the **‘acceptance rate’** printed to the screen when running InSilicoVA.
- If `jump.scale` is too large, at each iteration, the algorithm ‘tries’ more wild guesses, leading to many of such guesses rejected. This can waste many iterations of sampling.
- If `jump.scale` is too small, at each iteration, the algorithm makes new guesses that are very similar to current values. This may prevent the algorithm to explore the right range of parameters.

Fine tuning InSilicoVA (cont.)

Convergence also depends on how many proposed new parameters are accepted.

- This is directly reflected in `jump.scale` and the **‘acceptance rate’** printed to the screen when running InSilicoVA.
- If `jump.scale` is too large, at each iteration, the algorithm ‘tries’ more wild guesses, leading to many of such guesses rejected. This can waste many iterations of sampling.
- If `jump.scale` is too small, at each iteration, the algorithm makes new guesses that are very similar to current values. This may prevent the algorithm to explore the right range of parameters.
- Ideally, we want to ‘tune’ the algorithm so that the **acceptance rate** is neither too large or too small. {\blue{20% to 25% is usually recommended}}.

Fine tuning InSilicoVA (cont.)

Convergence also depends on how many proposed new parameters are accepted.

- This is directly reflected in `jump.scale` and the **‘acceptance rate’** printed to the screen when running InSilicoVA.
- If `jump.scale` is too large, at each iteration, the algorithm ‘tries’ more wild guesses, leading to many of such guesses rejected. This can waste many iterations of sampling.
- If `jump.scale` is too small, at each iteration, the algorithm makes new guesses that are very similar to current values. This may prevent the algorithm to explore the right range of parameters.
- Ideally, we want to ‘tune’ the algorithm so that the **acceptance rate** is neither too large or too small. {\blue{20% to 25% is usually recommended}}.
- In practice, typically as long as it is not very small (<5%) or very large (>50%), we have found InSilicoVA to be mostly robust, at least for causes with higher prevalence.

Fine tuning InSilicoVA: changing jump.scale

```
out2 <- codeVA(RandomVA5[1:25,], data.type = "WHO2016",
               jump.scale = 0.4)

InSilico Sampler initiated, 10000 iterations to sample
...................................................
Iteration: 500
Sub-population 0 acceptance ratio: 0.24
0.01min elapsed, 0.27min remaining
..................................................
Iteration: 1000
Sub-population 0 acceptance ratio: 0.25
0.03min elapsed, 0.25min remaining
..................................................
Iteration: 1500
Sub-population 0 acceptance ratio: 0.27
0.04min elapsed, 0.24min remaining
..................................................
```

Fine tuning InSilicoVA

- Changing `jump.scale` so that the acceptance ratio is in the recommended range may help with the warning

```
Not all causes with CSMF > 0.02 are convergent.
 Please check using csmf.diag() for more information.
```

Fine tuning InSilicoVA

- Changing `jump.scale` so that the acceptance ratio is in the recommended range may help with the warning

```
Not all causes with CSMF > 0.02 are convergent.
 Please check using csmf.diag() for more information.
```

- Ultimately, check the results of `csmf.diag()` and note that for causes that fail the test, we do not have conclusive results.

Fine tuning InSilicoVA

- Changing `jump.scale` so that the acceptance ratio is in the recommended range may help with the warning

```
Not all causes with CSMF > 0.02 are convergent.
 Please check using csmf.diag() for more information.
```

- Ultimately, check the results of `csmf.diag()` and note that for causes that fail the test, we do not have conclusive results.
  - (just not enough information in the data to estimate the fraction of deaths due to these causes).

Obtain individual summary

- We may also look more closely into some individuals

```
summary(out2, id = "d1", size = "scriptsize")
```

```
## Warning in summary.insilico(out2, id = "d1", size = "scriptsize"): C.I. for

## InSilicoVA fitted top  causes for death ID: d1
## Credible intervals shown: %
##                                        Mean Lower
## Stroke                            0.5546241    NA
## Digestive neoplasms               0.4120542    NA
## Other and unspecified neoplasms   0.0139036    NA
## Other and unspecified infect dis  0.0098763    NA
## Other and unspecified cardiac dis 0.0032896    NA
## Tetanus                           0.0032327    NA
## Renal failure                     0.0016230    NA
## Pulmonary tuberculosis            0.0003069    NA
##  Other and unspecified NCD        0.0002442    NA
## Severe anaemia                    0.0001715    NA
##                                   Median Upper
## Stroke                                NA    NA
## Digestive neoplasms                   NA    NA
## Other and unspecified neoplasms       NA    NA
## Other and unspecified infect dis      NA    NA
## Other and unspecified cardiac dis     NA    NA
## Tetanus                               NA    NA
```

Obtain individual summary

- As suggested in the warning message, for InSilicoVA, uncertainties associated with individual probabilities are not calculated by default to save computation time.

```
out2 <- updateIndiv(out2, CI = 0.95)
```

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

```
summary(out2, id = "d1")
```

```
## InSilicoVA fitted top  causes for death ID: d1
## Credible intervals shown: 95%
##                                         Mean
## Stroke                            0.5546241
## Digestive neoplasms               0.4120542
## Other and unspecified neoplasms   0.0139036
## Other and unspecified infect dis  0.0098763
## Other and unspecified cardiac dis 0.0032896
## Tetanus                           0.0032327
## Renal failure                     0.0016230
## Pulmonary tuberculosis            0.0003069
##  Other and unspecified NCD        0.0002442
```