Package 'InSilicoVA'

May 11, 2015

Type Package

Title Java implementation of InSilicoVA

Version 1.0

Date 2015-02-06

Author Zehang Li, Tyler McCormick, Sam Clark

Maintainer Zehang Li <lizehang@uw.edu>

Depends R (>= 2.15.0), rJava, coda, ggplot2

Description Computes individual cause of death and population cause-specific mortality fractions using the InSilicoVA algorithm.

License GPL-2

R topics documented:

`InsilicoVA-package` *What the package does (short line) ~~ package title ~~*

Description

Computes individual cause of death and population cause-specific mortality fractions using the InSilicoVA algorithm. Provides a simple graphical representation of the result.

Details

Package:	InsilicoVA
Type:	Package
Version:	1.0
Date:	2015-02-06
License:	GPL-2

Author(s)

Zehang Li, Tyler McCormick, Sam Clark

Maintainer: Zehang Li <lizehang@uw.edu>

References

Tyler H. McCormick, Zehang R. Li, Clara Calvert, Amelia C. Crampin, Kathleen Kahn and Samuel J. Clark Probabilistic cause-of-death assignment using verbal autopsies, *arXiv preprint arXiv:1411.3042* http://arxiv.org/abs/1411.3042 (2014)

`causetext` *Translation list of COD codes*

Description

This is the translation of COD abbreviation codes into their corresponding full names.

Format

A data frame with the translation of COD codes to their names on 68 CODs (both the version of COD only and COD with group code).

Examples

```
data(causetext)
```

`condprob` *Conditional probability table used by InterVA-4*

Description

This is a conditional probability matrix used by InterVA-4.2. There are 60 causes and 245 symptoms. The orders of the rows and columns must not be changed.

Format

A data frame with 245 observations on 60 variables. Each observation is the conditional probability.

Examples

```
data(condprob)
```

`csmf.diag` *Convergence test for fitted InSilico model*

Description

Produce convergence test for CSMFs from fitted `"insilico"` objects.

Usage

```
csmf.diag(csmf, conv.csmf = 0.02, test= c("gelman", "heidel")[2], verbose = TRUE,
          autoburnin = FALSE, ...)
```

Arguments

`csmf`	It could be either fitted `"insilico"` object, a list of fitted `"insilico"` object from different chains of the same length but different starting values, i.e., different seed. Or it could be the matrix of CSMF obtained by `insilico`, or the list of matrices of CSMF. All CSMF could contain more than one subpopulations, but should be in the same format and order.
`conv.csmf`	The minimum mean CSMF to be checked. Default to be 0.02, which means any causes with mean CSMF lower than 0.02 will not be tested.
`test`	Type of test. Currently supporting Gelman and Rubin's test (`test = "gelman"`) for multi-chain test, and Heidelberger and Welch's test (`test = "heidel"`) for single-chain test.
`verbose`	Logical indicator to return the test detail instead of one logical outcome for Heidelberger and Welch's test. Default to be TRUE.
`autoburnin`	Logical indicator of whether to omit the first half of the chain as burn in. Default to be FALSE since `insilico` return only the iterations after burnin by default.
`...`	Arguments to be passed to heidel.diag or gelman.diag

Details

The tests are performed using heidel.diag and gelman.diag functions in coda package. The function takes either one or a list of output from `insilico` function, or only the iteration by CSMF matrix. Usually in practice, many causes with very tiny CSMF are hard to converge based on standard tests, thus it is suggested to check convergence for only causes with mean CSMF over certain threshold by setting proper `conv.csmf`.

Note for Gelman and Rubin's test, all chains should have the same length. If the chains are sampled with automatically length determination, they might not be comparable by this test.

Author(s)

Zehang Li, Tyler McCormick, Sam Clark

Maintainer: Zehang Li <lizehang@uw.edu>

References

Tyler H. McCormick, Zehang R. Li, Clara Calvert, Amelia C. Crampin, Kathleen Kahn and Samuel J. Clark Probabilistic cause-of-death assignment using verbal autopsies, *arXiv preprint arXiv:1411.3042* http://arxiv.org/abs/1411.3042 (2014)

Gelman, Andrew, and Donald B. Rubin. Inference from iterative simulation using multiple sequences. *Statistical science* (1992): 457-472.

Brooks, Stephen P., and Andrew Gelman. General methods for monitoring convergence of iterative simulations. *Journal of computational and graphical statistics* 7.4 (1998): 434-455.

Heidelberger, Philip, and Peter D. Welch. A spectral method for confidence interval generation and run length control in simulations. *Communications of the ACM* 24.4 (1981): 233-245.

Heidelberger, Philip, and Peter D. Welch. Simulation run length control in the presence of an initial transient. *Operations Research* 31.6 (1983): 1109-1144.

Schruben, Lee W. Detecting initialization bias in simulation output. *Operations Research* 30.3 (1982): 569-590.

See Also

insilico, summary.insilico

Examples

```
# load sample data together with sub-population list
data(SampleInput_insilico)
## Not run:
# extract INterVA style input data
data <- SampleInput_insilico$data
# extract sub-population information.
# The groups are "HIV Positive", "HIV Negative" and "HIV status unknown".
subpop <- SampleInput_insilico$subpop

# run without sub-population
fit1a<- insilico( data, subpop = NULL,
              length.sim = 400, burnin = 200, thin = 10 , seed = 1,
              auto.length = FALSE)
fit1b<- insilico( data, subpop = NULL,
              length.sim = 400, burnin = 200, thin = 10 , seed = 2,
              auto.length = FALSE)
fit1c<- insilico( data, subpop = NULL,
              length.sim = 400, burnin = 200, thin = 10 , seed = 3,
              auto.length = FALSE)
# single chain check
csmf.diag(fit1a)
# equivalent way of check one chain
csmf.diag(fit1a$csmf)

# multiple chains check
csmf.diag(list(fit1a, fit1b, fit1c), test = "gelman")
# equivalent way of check one chain
csmf.diag(list(fit1a$csmf, fit1b$csmf, fit1c$csmf), test = "gelman")


# with sub-populations
fit2a<- insilico( data, subpop = subpop,
```

```
              length.sim = 400, burnin = 200, thin = 10 , seed = 1,
              auto.length = FALSE)
fit2b<- insilico( data, subpop = subpop,
              length.sim = 400, burnin = 200, thin = 10 , seed = 2,
              auto.length = FALSE)
fit2c<- insilico( data, subpop = subpop,
              length.sim = 400, burnin = 200, thin = 10 , seed = 3,
              auto.length = FALSE)

# single chain check
csmf.diag(fit2a)
# equivalent way of check one chain
csmf.diag(fit2a$csmf)

# multiple chains check
csmf.diag(list(fit2a, fit2b, fit2c), test = "gelman")
# equivalent way of check one chain
csmf.diag(list(fit2a$csmf, fit2b$csmf, fit2c$csmf), test = "gelman")

## End(Not run)
```

`insilico` *Implement InSilicoVA methods*

Description

This function implements InSilicoVA model. The InSilicoVA model is fitted with MCMC implemented in Java. For more detail, see the paper on http://arxiv.org/abs/1411.3042.

For Windows user, this function will produce a popup window showing the progress. For Mac and Unix user, this function will print progress messages on the console. Special notice for users using default R GUI for mac, the output will not be printed on console while the function is running, and will only be printed out after it is completed. Thus if you use a Mac, we suggest using either RStudio for mac, or running R from terminal.

Usage

```
insilico(data, isNumeric = FALSE,useProbbase = FALSE, keepProbbase.level = TRUE,
         cond.prob.touse = NULL, datacheck = TRUE, warning.write = FALSE,
         external.sep = TRUE, length.sim = 4000, thin = 10, burnin = 2000,
         auto.length = TRUE, conv.csmf = 0.02, jump.scale = 0.1,
         levels.prior = NULL, levels.strength = 1, trunc.min = 0.0001,
         trunc.max = 0.9999, subpop = NULL, java_option = "-Xmx1g", seed = 1)
```

Arguments

`data`	The original data to be used. It is suggested to use similar input as InterVA4, with the first column being death IDs. The only difference in input is InsilicoVA takes three levels: "present", "absent", and "missing (no data)". Similar to InterVA software, "present" symptoms takes value "Y"; "absent" symptoms take take value "NA" or "". For missing symptoms, e.g., questions not asked or answered in the original interview, corrupted data, etc., the input should be coded by "." to distinguish from "absent" category.

`isNumeric` Indicator if the input is already in numeric form. If the input is coded numerically such that 1 for "present", 0 for "absent", and -1 for "missing", this indicator could be set to True to avoid conversion to standard InterVA format.

`useProbbase` Logical indicator. If TRUE, then use InterVA conditional probability table without re-estimating.

`keepProbbase.level`
Logical indicator when `useProbbase` is FALSE. If TRUE, then only estimate the InterVA's conditional probability table interpretation table; if FALSE, estimate the whole conditional probability matrix. Default to TRUE.

`cond.prob.touse`
Customized conditional probability table to use. Currently only accepting the same configuration as InterVA-4 software. It should be a matrix of 245 rows of symptoms and 60 columns of causes, arranged in the same order as in InterVA-4 specification. The elements in the matrix should be the conditional probability of corresponding symptom given the corresponding cause, represented in alphabetic form indicating levels. For example input, see `condprob`

`datacheck` Logical indicator for whether to check the data satisfying InterVA rules. Default set to be TRUE. If `warning.write` is set to true, the inconsistent input will be logged in file warnings.txt. It's strongly suggested to be set to TRUE.

`warning.write` Logical indicator for whether to save the changes made to data input by `datacheck`. If set to TRUE, the changes will be logged in file warnings.txt in current working directory.

`external.sep` Logical indicator for whether to separate out external causes first. Default set to be TRUE. If set to TRUE, the algorithm will estimate external causes, e.g., traffic accident, accidental fall, suicide, etc., by checking the corresponding indicator only without considering other medical symptoms. It is strongly suggested to set to be TRUE.

`length.sim` Number of iterations to run. Default to be 4000.

`thin` Proportion of thinning for storing parameters. For example, if thin = k, the output parameters will only be saved every k iterations. Default to be 10

`burnin` Number of iterations as burn-in period. Parameters sampled in burn-in period will not be saved.

`auto.length` Logical indicator of whether to automatically increase chain length if convergence not reached.

`conv.csmf` Minimum CSMF value to check for convergence if auto.length is set to TRUE. For example, under the default value 0.02, all causes with mean CSMF at least 0.02 will be checked for convergence.

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

`levels.prior` 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`
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` Minimum possible value for estimated conditional probability table. Default to be 0.0001

`trunc.max` Maximum possible value for estimated conditional probability table. Default to be 0.9999

`subpop` A vector of sub-population assignments of the same length of death records. It could be numerical indicators or character vectors of names.

`java_option` Option to initialize java JVM. Default to "-Xmx1g", which sets the maximum heap size to be 1GB. If R produces "java.lang.OutOfMemoryError: Java heap space" error message, consider increasing heap size using this option, or one of the following: (1) decreasing `length.sim`, (2) increasing `thin`, or (3) disabling `auto.length`.

`seed` Seed used for initializing sampler. The algorithm will produce the same outcome with the same seed in each machine.

Details

The chains could be set to run automatically longer. If set `auto.length` to be TRUE, the chain will assess convergence after finishing the length K chain input by user using Heidelberger and Welch's convergence diagnostic. If convergence is not reached, the chain will run another K iterations and use the first K iterations as burn-in. If the chain is still not converged after 2K iterations, it will proceed to another 2K iterations and again use the first 2K iterations as burn-in. If convergence is still not reached by the end, it will not double the length again to avoid heavy memory use. A warning will be given in that case. The extended chains will be thinned in the same way.

For more detail of model specification, see the paper on `http://arxiv.org/abs/1411.3042`.

Value

`id` A vector of death ID. Note the order of the ID is in general different from the input file. See `report` for organizing the report.

`indiv.prob` Matrix of individual mean cause of death distribution. Each row corresponds to one death with the corresponding ID.

`csmf` Matrix of CSMF vector at each iterations after burn-in and thinning. Each column corresponds to one cause.

`conditional.probs`
If the model is estimated with `keepProbbase.level` = TRUE, this value gives a matrix of each conditional probability at each level at each iterations. Each column corresponds to one level of probability. If `keepProbbase.level` = FALSE, this value gives a three-dimensional array. If `UseProbbase` = TRUE, the value will be set to NULL. See `report` for more analysis.

`missing.symptoms`
Vector of symptoms missing from all input data.

`external` Logical indicator of whether the model is fitted with external causes separated calculated.

Author(s)

Zehang Li, Tyler McCormick, Sam Clark

Maintainer: Zehang Li <lizehang@uw.edu>

References

Tyler H. McCormick, Zehang R. Li, Clara Calvert, Amelia C. Crampin, Kathleen Kahn and Samuel J. Clark(2014) *Probabilistic cause-of-death assignment using verbal autopsies*, http://arxiv.org/abs/1411.3042
Working paper no. 147, Center for Statistics and the Social Sciences, University of Washington

See Also

plot.insilico, summary.insilico

Examples

```
# load sample data together with sub-population list
data(SampleInput_insilico)
# extract InterVA style input data
data <- SampleInput_insilico$data
# extract sub-population information.
# The groups are "HIV Positive", "HIV Negative" and "HIV status unknown".
subpop <- SampleInput_insilico$subpop

# run without subpopulation
fit1<- insilico( data, subpop = NULL,
              length.sim = 400, burnin = 200, thin = 10 , seed = 1,
              external.sep = TRUE, keepProbbase.level = TRUE)

# re-run with subpopulation
fit2<- insilico( data, subpop = subpop,
              length.sim = 400, burnin = 200, thin = 10 , seed = 1,
              external.sep = TRUE, keepProbbase.level = TRUE)

# run without re-sampling conditional probabilities
fit3<- insilico( data, subpop = subpop,
              length.sim = 400, burnin = 200, thin = 10 , seed = 1,
              external.sep = TRUE, useProbbase = TRUE)
```

---

plot.insilico *plot CSMF from a "insilico" object*

Description

Produce a bar plot of the CSMFs for a fitted "insilico" object.

Usage

```
## S3 method for class 'insilico'
plot(x, type = c("errorbar", "bar", "compare")[1],
  top = 10, causelist = NULL, which.sub = NULL,
  xlab = "Causes", ylab = "CSMF", title = "Top CSMF Distribution",
  horiz = TRUE, angle = 60, fill = "lightblue",
  err_width = .4, err_size = .6, point_size = 2,
  border = "black", bw = FALSE, ...)
```

Arguments

x	fitted "insilico" object
type	An indicator of the type of chart to plot. "errorbar" for line plots of only the error bars on single population; "bar" for bar chart with error bars on single population; "compare" for line charts on multiple sub-populations.
top	The number of top causes to plot. If multiple sub-populations are to be plotted, it will plot the union of the top causes in all sub-populations.
causelist	The list of causes to plot. It could be a numeric vector indicating the position of the causes in the InterVA cause list (see causetext), or a vector of character string of the cause names. The argument supports partial matching of the cause names. e.g., "HIV/AIDS related death" could be abbreviated into "HIV"; "Other and unspecified infect dis" could be abbreviated into "Other and unspecified infect".
which.sub	Specification of which sub-population to plot if there are multiple and type is set to "bar".
xlab	Labels for the causes.
ylab	Labels for the CSMF values.
title	Title of the plot.
horiz	Logical indicator indicating if the bars are plotted horizontally.
angle	Angle of rotation for the texts on x axis when horiz is set to FALSE
fill	The color to fill the bars when type is set to "bar".
border	The color to color the borders of bars when type is set to "bar".
err_width	Size of the error bars.
err_size	Thickness of the error bar lines.
point_size	Size of the points.
bw	Logical indicator for setting the theme of the plots to be black and white.
...	Not used.

Details

To-do

Author(s)

Zehang Li, Tyler McCormick, Sam Clark

Maintainer: Zehang Li <lizehang@uw.edu>

References

Tyler H. McCormick, Zehang R. Li, Clara Calvert, Amelia C. Crampin, Kathleen Kahn and Samuel J. Clark Probabilistic cause-of-death assignment using verbal autopsies, *arXiv preprint arXiv:1411.3042* http://arxiv.org/abs/1411.3042 (2014)

See Also

insilico, summary.insilico

Examples

```
# load sample data together with sub-population list
data(SampleInput_insilico)
# extract INterVA style input data
data <- SampleInput_insilico$data
# extract sub-population information.
# The groups are "HIV Positive", "HIV Negative" and "HIV status unknown".
subpop <- SampleInput_insilico$subpop

# run without sub-population
fit1<- insilico( data, subpop = NULL,
              length.sim = 400, burnin = 200, thin = 10 , seed = 1,
              external.sep = TRUE, keepProbbase.level = TRUE)
# default plot
plot(fit1)

# customized line plot
plot(fit1, top = 15, horiz = FALSE, fill = "gold",
   bw = TRUE, title = "Top 15 CSMFs", angle = 70,
       err_width = .2, err_size = .6, point_size = 2)

# customized bar plot
plot(fit1, type = "bar", top = 15, horiz = TRUE,
       bw = TRUE, title = "Top 15 CSMFs", angle = 70,
       err_width = .5, err_size = .6)

# run with sub-populations
fit2<- insilico( data, subpop = subpop,
              length.sim = 400, burnin = 200, thin = 10 , seed = 1,
              external.sep = TRUE, keepProbbase.level = TRUE)
# default plot
plot(fit2, type = "compare", top = 5, title = "Top 5 causes comparison")
# customized single sub-population plot
plot(fit2, which.sub = "Unknown",
 title = "Top 10 causes in HIV status unknown population")
# customized plot with specified causes, with abbreviation here.
some_causes <- c("HIV", "Pulmonary",
"Other and unspecified infect dis")
plot(fit2, type = "compare", horiz = FALSE,  causelist = some_causes,
   title = "Infectious diseases in three sub-populations",
   angle = 20)
```

---

probbase *Conditional probability of InterVA4*

Description

This is the table of conditional probabilities of symptoms given CODs, together with the data check rules. The values are from InterVA-4.2.

Format

A data frame with 246 observations on 81 variables.

Examples

```
data(probbase)
```

`SampleInput_insilico` *100 records of Sample Input*

Description

This is a dataset consisting of 10 arbitrary sample input deaths in the acceptable format of InSilicoVA and InterVA4. Any data that needs to be analyzed by this package should be in the same format. The orders of the input fields must not be changed.

Format

100 arbitrary input records.

Examples

```
data(SampleInput_insilico)
```

`summary.insilico` *Summarizing InSilicoVA Model Fits*

Description

This function is the summary method for class `insilico`.

Usage

```
## S3 method for class 'insilico'
summary(object, CI.csmf = 0.95, CI.cond = 0.95,
    file = NULL, top = 10, ...)
```

Arguments

`object`	Fitted `"insilico"` object.
`CI.csmf`	Confidence interval for CSMF estimates.
`CI.cond`	Confidence interval for conditional probability estimates
`file`	Optional .csv file to write to. If it is specified, individual cause of death distribution will be saved to the file.
`top`	Number of top causes to display on screen.
`...`	Not used.

Details

`summary.insilico` formats some basic information about the InSilicoVA fitted object on screen and show the several top CSMFs of user's choice. See below for more detail.

Value

id	the ID of the deaths.
indiv	individual Cause of Death distribution matrix.
csmf	CSMF distribution and confidence interval for each cause.
csmf.ordered	CSMF distribution and confidence interval for each cause, ordered by mean.
condprob	Conditional probability matrix and confidence intervals.
useProbbase	Component of "insilico" object.
keepProbbase.level	Component of "insilico" object.
datacheck	Component of "insilico" object.
length.sim	Component of "insilico" object.
thin	Component of "insilico" object.
burnin	Component of "insilico" object.
jump.scale	Component of "insilico" object.
levels.prior	Component of "insilico" object.
levels.strength	Component of "insilico" object.
trunc.min	Component of "insilico" object.
trunc.max	Component of "insilico" object.
subpop_counts	Component of "insilico" object.
showTop	Component of "insilico" object.

Author(s)

Zehang Li, Tyler McCormick, Sam Clark

Maintainer: Zehang Li <lizehang@uw.edu>

References

Tyler H. McCormick, Zehang R. Li, Clara Calvert, Amelia C. Crampin, Kathleen Kahn and Samuel J. Clark Probabilistic cause-of-death assignment using verbal autopsies, *arXiv preprint arXiv:1411.3042* http://arxiv.org/abs/1411.3042 (2014)

See Also

insilico, plot.insilico

Examples

```
# load sample data together with sub-population list
data(SampleInput_insilico)
# extract INterVA style input data
data <- SampleInput_insilico$data
# extract sub-population information.
# The groups are "HIV Positive", "HIV Negative" and "HIV status unknown".
subpop <- SampleInput_insilico$subpop

# run without subpopulation
```

```
fit1<- insilico( data, subpop = NULL,
              length.sim = 400, burnin = 200, thin = 10 , seed = 1,
              external.sep = TRUE, keepProbbase.level = TRUE)
summary(fit1)
summary(fit1, top = 10)
```

Index