Reproducing the ITS-2 model using R

Jason R Pirone[1], Marjolein Smith[1], Nicole Kleinstreuer[2], Thomas Burns[2], Judy Strickland[2], Yuri Dancik[4], Richard Morris[1], Lori Rinckel[2], Warren Casey[3], and Joanna Jaworska[4]

[1]Social & Scientific Systems, Inc., Durham, NC 27713, USA
[2]Integrated Laboratory Systems, Inc., Research Triangle Park, NC 27709, USA
[3]National Toxicology Program Interagency Center for the Evaluation of Alternative Toxicological Methods, Division of the National Toxicology Program, National Institute of Environmental Health Sciences, Research Triangle Park, NC 27709, USA
[4]Procter & Gamble NV, Strombeek - Bever, Belgium

March 31, 2014

Contents

1 Preliminaries

1.1 Required software

R is available for download from the Comprehensive R Archive Network (CRAN). The installation process is simple and well-documented on the CRAN website. Once R is installed, the packages `gRbase`, `gRain`, `poLCA`, and `discretization`, as well as any dependencies should be installed from within R using the `install.packages` function. The packages `graph`, `RBGL`, and `Rgraphviz` are also necessary and may not be available from CRAN; these packages are available from the Bioconductor site and are installed by running the following commands within R:

```
> source("http://bioconductor.org/biocLite.R")
> biocLite(c("graph","RBGL","Rgraphviz"))
```

See Section 7.1 of this document for the hardware configuration and software versions used in conducting this work. The procedure will be similar for most operating systems and versions of R.

1.2 Sweave background

`Sweave` [8] is an R function that combines analysis code and formatted text into a single report. This level of integration ensures that all results, including figures and tables, can be easily reproduced. There is a growing consensus that emphasizes practices enabling published research in general to be reproduced by others ([2], [6], [10], [11]).

To regenerate this report, install an appropriate LaTeX implementation (e.g., Microsoft Windows users should install MiKTeX). The Sweave homepage contains detailed instructions on how to write and compile Sweave documents. An integrated development environment (IDE) like RStudio or Emacs+ESS+AUCTeX will simplify report generation.

A separate file, `ITS2_R_version.R`, that contains only the R code is provided for users that do not want to use `Sweave`.

2 Overview of the approach

The integrated test strategy model (ITS-2) described in [4] uses a Bayesian network to describe the relationships between *in silico*, *in vitro*, and *in vivo* information relevant to skin sensitization. Using a Bayesian network as part of an integrated testing strategy provides a probabilistic framework that goes beyond simple classification and facilitates complex reasoning about the skin sensitization hazard or potency of a chemical given the available evidence. A Bayesian network has a qualitative and a quantitative component. The qualitative part consists of a directed acyclic graph (DAG) where each node represents an assay outcome and each edge (or arrow) indicates that there is a relationship (potentially causal) between the variables it connects (see Figure 2). The quantitative component consists of a set of conditional probability tables that give

the probability that an assay outcome has a particular value given the values taken by its parent[1] nodes.

The original Bayesian network ITS developed by Jaworska et al. (2011; 2013) was developed using a used commercial software package. The use of commercial software is convenient in corporate settings where the model is developed primarily for internal use, but this practice can limit use of the model by the larger scientific and regulatory communities. An implementation using a freely available, open-source software would raise awareness of the approach by making it easier to test, verify, and build upon the analysis framework. We have developed an open-source ITS (OS ITS-2) using tools in the R software package to build and perform exact inference using a Bayesian network.

Figure 1 gives an overview of the process used to build the ITS-2 model. The primary computational steps are:

1. Find cut-points for each assay using a supervised discretization algorithm on the training data. Use cut-points to discretize the training and test datasets.

2. Cluster the assays related to bioavailability (`logKow`, `Cfree`, and `AUC120`) and cysteine reactivity to construct the `Bioavailability` (BA) and `Cysteine` latent variables respectively.

3. Compile the observed and latent variables, the DAG defining the relationships among the variables, and the conditional probability tables for each node into a Bayesian network. The compiled Bayesian network is used to make LLNA potency class predictions on the test data.

We do not have access to the exact algorithms used by the commercial software for variable discretization, latent variable learning, and Bayesian networks inference. However, we do have access to the output from the commercial software at each of the steps described in Figure 1. The output of step 1 is a transformed dataset where the continuous assay values are converted to discrete values. In step 2, the dataset created in step 1 is augmented to include the `Bioavailability` and `Cysteine` latent variables. Finally, the transformed and augmented dataset is used to build the Bayesian network in step 3. In the following sections, we use the datasets produced by the commercial software package as inputs to the R model. For example, we examine the similarity of the R algorithms for Bayesian network inference by using the dataset produced by the commercial software at steps 1 and 2 using as an input. Similarly, to compare the latent variable learning algorithm in R with that of the commercial software package, we use the dataset produced by the commercial software package at step 1 as an input.

[1]The parents of a node are all nodes with a directed edge (arrow) pointing into that node.

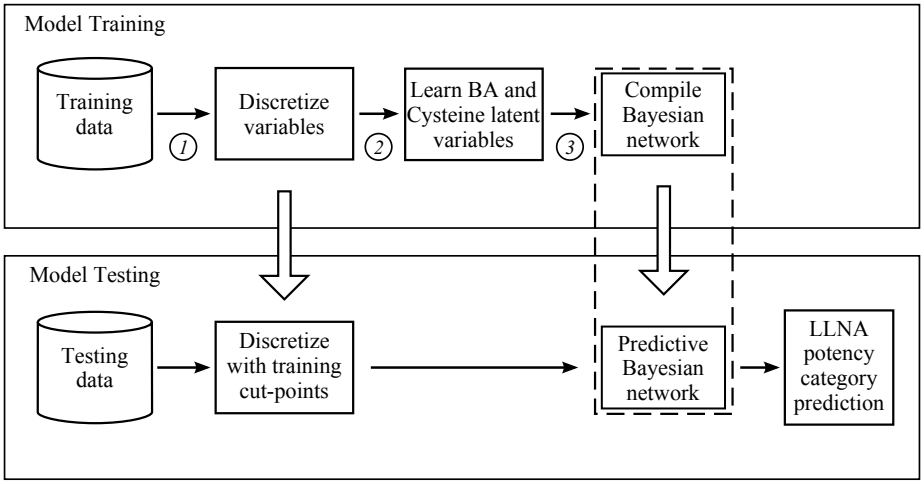

Figure 1: Diagram showing the key computational steps of the ITS-2 modeling process.

3 Demonstration of the equivalence of R using the discretization and latent variable values found using the commercial software package

3.1 Load necessary packages

The following R packages are sufficient for performing an analysis similar to that described in the ITS-2 manuscript [4]. Other R packages available from CRAN might have similar functionality.

- `gRbase` [1] and `gRain` [3] supply the tools for constructing, parameterizing, and performing inference on graphical independence networks.
- `discretization` [5] contains implementations of several algorithms for supervised and unsupervised discretization.
- `poLCA` [9] is used for learning the latent variables.

The path referred to by `projectDirectory` must be changed to reflect the location of the files downloaded from the http://ntp.niehs.nih.gov/go/its. To regenerate this report in its entirety, all of the following files must be in the directory pointed to by `projectDirectory`

1. `ITS2_Lipid_Train_102313.txt`
2. `ITS2_Lipid_Test_102313.txt`
3. `ITS2_Supplemental_R_Functions.R`

4. ITS2_R_version.Rnw

5. ITS2_Refs.bib

6. ITS2_Process_Diagram.pdf

7. Sweave.sty

If only the R code is of interest, the directory needs to contain only items 1–4.

```
> projectDirectory <- "c:/Local_Modeling/AltexSubmission"
```

Load the required packages and source the supplemental R functions.

```
> library(gRain)
> library(poLCA)
> library(discretization)
> library(xtable)
> source(file.path(projectDirectory,"ITS2_Supplemental_R_Functions.R"))
```

Note that `gRbase` is loaded automatically with `gRain`. The `xtable` package is not necessary for the analysis, but is used to format some of the tables included in this document. Additional small functions necessary for the analysis are included in the file `ITS2_Supplemental_R_Functions.R`. The following sections provide step-by-step instructions on conducting an analysis similar to that described in the ITS-2 manuscript [4], but using the ITS-2 lipid data set.

3.2 Load the training and test data

```
> trainData <- read.delim(file.path(projectDirectory,"ITS2_Lipid_Train_102313.txt"),
                          row.names=1)
> names(trainData)
 [1] "LLNA"                "KEC1.5"              "KEC3"
 [4] "IC50"                "CD86"                "DPRACys"
 [7] "DPRALys"             "logKow"              "Cfree"
[10] "AUC120"              "TIMES"               "Cysteine"
[13] "Bioavailability"     "LLNA.1"              "LLNA.2"
[16] "LLNA.3"              "LLNA.4"              "LLNA.Expected.Value"
```

The training dataset contains 124 chemicals and 18 variables. The variables `LLNA, KEC1.5, KEC3, IC50, CD86, DPRACys, DPRALys, logKow, Cfree, AUC120`, and `TIMES` are described in the ITS-2 manuscript [4]. `Cysteine` and `Bioavailability` are the values of the latent variables learned using the commercial software package; the probability that a chemical is in each LLNA class given the evidence is given by the variables `LLNA.1, LLNA.2, LLNA.3`, and `LLNA.4`. The most likely LLNA class is given by `LLNA.Expected.Value`.

```
> testData <- read.delim(file.path(projectDirectory,"ITS2_Lipid_Test_102313.txt"),
                         row.names=1)
```

The test dataset contains 21 chemicals and 24 variables. For the test chemicals, the Bayesian network was used to make predictions for `Bioavailability` and `Cysteine` as well as for `LLNA`, so conditional probability distributions and most probable classes for these variables are included.

3.3 Define the directed acyclic graph (DAG)

Figure 2 depicts the DAG shown in Figure 2 of the ITS-2 manuscript [4].

```
> its2Dag <- dag(~KEC1.5:IC50:Cysteine +
                 KEC3:IC50:Cysteine +
                 TIMES:Cysteine:LLNA +
                 DPRACys:Cysteine +
                 CD86:Cysteine:LLNA +
                 DPRALys:LLNA +
                 Cysteine:LLNA +
                 Bioavailability:LLNA +
                 logKow:Bioavailability +
                 AUC120:Bioavailability +
                 Cfree:Bioavailability)
```

3.4 Discretize the training data

The continuous variables are discretized using the cut-points determined using the discretization procedures in the commercial software package. These values were determined from the ITS-2 lipid dataset, so they are slightly different from the values listed in the supplemental material for the ITS-2 manuscript [4].

```
> its2DiscList <- list(Cfree=c(-Inf,0.021,0.068,0.199,Inf),
                       CD86=c(-Inf,27.15,295.8,1025,Inf),
                       logKow=c(-Inf,0.094,1.919,3.834,Inf),
                       AUC120=c(-Inf,1.569,9.101,25.,Inf),
                       DPRACys=c(-Inf,15,70.45,90,Inf),
                       DPRALys=c(-Inf,70,95.35,Inf),
                       KEC3=c(-Inf,34.115,476.19,945.028,Inf),
                       KEC1.5=c(-Inf,10.085,238.765,1098.969,Inf),
                       IC50=c(-Inf,159.26,763.68,Inf))
> discIts2TrainData <- trainData[,-grep("^LLNA.",colnames(trainData))]
> discIts2TrainData <- cutData(discIts2TrainData,its2DiscList)
```

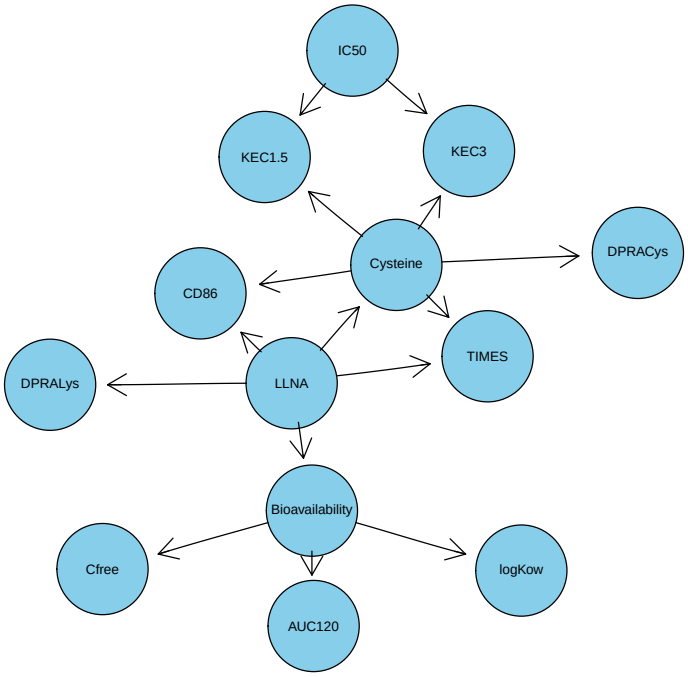

Figure 2: Simple plot of the ITS-2 DAG.

3.5 Extract conditional probability tables (CPTs)

The commercial software package and `gRain` have slightly different methods to handle unobserved combinations of states in CPTs. In `gRain`, smoothing is done by adding a small user-defined number to each cell in a table of occurrences. In the commercial package, $\frac{1}{n_k}$, where n_k is the number of possible states in the CPT, is added to each cell. The smoothing method used by the commercial package can be implemented by making slight modifications to the `extractCPT` function (see the `extractCPT.Commercial` function in the `ITS2_Supplemental_R_Functions.R` file), or by manually constructing the CPTs using the `xtabs` and `parray` functions.

```
> its2Cpt <- extractCPT.Commercial(discIts2TrainData,its2Dag)
> its2Gin <- grain(compileCPT(its2Cpt))
```

3.6 Predict LLNA class for the training dataset

`gRain` contains a simple prediction method. Setting `type = class` returns the most probable class, while setting `type = distribution` returns the conditional distribution.

```
> trainPredClass <- predict.grain(its2Gin,"LLNA",
                                  type="class",
                                  newdata=discIts2TrainData)
> trainPredDist <-  predict.grain(its2Gin,"LLNA",
                                  type="distribution",
                                  newdata=discIts2TrainData)
> trainPredTable <- table(as.numeric(trainPredClass$pred$LLNA),
                        discIts2TrainData$LLNA,
                        dnn=c("predicted","observed"))
```

	1	2	3	4
1	29	1	1	1
2	3	21	2	0
3	3	4	24	4
4	1	2	8	20

Table 1: Confusion matrix for LLNA class predictions on training data. Rows are instances in the predicted class and columns are instances in the observed class. (n = 124).

The structure of the confusion matrix in Table 1 is the same as in the ITS-2 manuscript [4]. Furthermore, the conditional distributions for LLNA predicted by `gRain` and the commercial package are exactly the same. Table 2 shows the `gRain` results and Table 3 shows the commercial package results for 10 randomly

selected chemicals. These identical results are expected, since both software packages carry out exact inference on the Bayesian network using variants of the junction tree algorithm.

```
> select10Train <- sample(nrow(discIts2TrainData),10)
> gRain10 <- cbind(rownames(discIts2TrainData)[select10Train],
                   format(trainPredDist$pred$LLNA[select10Train,],
                          digits = 2, scientific = FALSE))
> colnames(gRain10) <- c("CASRN",paste0("LLNA.", 1:4))
> commercial10 <- format(as.matrix(trainData[select10Train,
                                             grep("LLNA\\.[0-9]",colnames(trainData))]),
                         digits = 2, scientific = FALSE)
> commercial10 <- cbind(rownames(commercial10), commercial10)
> colnames(commercial10) <- c("CASRN",paste0("LLNA.", 1:4))
```

CASRN	LLNA.1	LLNA.2	LLNA.3	LLNA.4
100-06-1	0.971694	0.027977	0.000272	0.000056
122-57-6	0.001713	0.000808	0.793932	0.203547
637-07-0	0.954734	0.021989	0.000309	0.022968
107-75-5	0.000160	0.947495	0.052049	0.000296
3055-86-5	0.906084	0.062214	0.002272	0.029430
121-79-9	0.000443	0.000907	0.776881	0.221769
874-23-7	0.413555	0.086810	0.499461	0.000175
2634-33-5	0.000206	0.000509	0.790698	0.208588
87-86-5	0.057037	0.116919	0.333108	0.492937
119-84-6	0.046210	0.001197	0.207419	0.745174

Table 2: Predicted class by gRain.

CASRN	LLNA.1	LLNA.2	LLNA.3	LLNA.4
100-06-1	0.971694	0.027977	0.000272	0.000056
122-57-6	0.001713	0.000808	0.793932	0.203547
637-07-0	0.954734	0.021989	0.000309	0.022968
107-75-5	0.000160	0.947495	0.052049	0.000296
3055-86-5	0.906084	0.062214	0.002272	0.029430
121-79-9	0.000443	0.000907	0.776881	0.221769
874-23-7	0.413555	0.086810	0.499461	0.000175
2634-33-5	0.000206	0.000509	0.790698	0.208588
87-86-5	0.057037	0.116919	0.333108	0.492937
119-84-6	0.046210	0.001197	0.207419	0.745174

Table 3: Predicted class by commercial software package.

3.7 Predict LLNA class for the test data set

The test data is discretized in the exactly the same way as the training data.

```
> discIts2TestData <- testData[,-grep("^LLNA.|^Bioavailability.|^Cysteine.",
                                      colnames(testData))]
> discIts2TestData <- cutData(discIts2TestData,its2DiscList)
```

In the ITS-2 manuscript [4], a uniform distribution on LLNA was applied before making predictions on the test data. This is done in `gRain` by modifying and recompiling the `its2Gin` object.

```
> unifLlnaCpt <- its2Gin$cptlist
> unifLlnaCpt$LLNA <- parray(c("LLNA"),
                             list(as.character(1:4)),
                             values=rep(.25,4))
> its2UnifGin <- grain(compileCPT(unifLlnaCpt))
> testPredDist <-  predict.grain(its2UnifGin,
                                 c("LLNA","Bioavailability","Cysteine"),
                                 type="distribution",
                                 newdata=discIts2TestData)
> testPredClass <-  predict.grain(its2UnifGin,
                                  c("LLNA","Bioavailability","Cysteine"),
                                  type="class",
                                  newdata=discIts2TestData)
> testPredTable <- table(as.numeric(testPredClass$pred$LLNA),
                         testData$LLNA,
                         dnn=c("predicted","observed"))
```

The `gRain` predictions on the test set chemicals are identical to those of the commercial package. The `Bioavailability` and `Cysteine` predictions are also identical. The conditional distributions for `LLNA` are shown in Table 4 (`gRain` predictions) and Table 5 (commercial software predictions). For the sake of brevity, results are shown for the first 10 chemicals only. The structure of the confusion matrix in Table 6 is the same as that determined using the commercial software package.

4 Evaluation of latent variable learning using the poLCA package

The the R package `poLCA` is used to learn the latent variables. The exact algorithm used by the commercial software to learn the latent variable is not known. However, we can compare the methods if the datset discretized by the commercial software package is used as the input to `poLCA`.

The number of states for the `Bioavailability` and `Cysteine` latent variables was set to 3. The number of states can be optimized by running the

CASRN	LLNA.1	LLNA.2	LLNA.3	LLNA.4
5910-85-0	0.000503	0.001592	0.740604	0.257301
3326-32-7	0.002029	0.010125	0.188737	0.799109
2785-87-7	0.072281	0.003438	0.391915	0.532366
108-90-7	0.939476	0.027764	0.000313	0.032447
67-63-0	0.977885	0.016434	0.000304	0.005376
50-21-5	0.977870	0.016438	0.000304	0.005388
119-36-8	0.976166	0.000455	0.000205	0.023174
69-72-7	0.828779	0.001618	0.001438	0.168164
5392-40-5	0.005458	0.065849	0.906857	0.021835
101-86-0	0.003086	0.968323	0.028381	0.000210

Table 4: Predicted class by gRain.

CASRN	LLNA.1	LLNA.2	LLNA.3	LLNA.4
5910-85-0	0.000503	0.001592	0.740604	0.257301
3326-32-7	0.002029	0.010125	0.188737	0.799109
2785-87-7	0.072281	0.003438	0.391915	0.532366
108-90-7	0.939476	0.027764	0.000313	0.032447
67-63-0	0.977885	0.016434	0.000304	0.005376
50-21-5	0.977870	0.016438	0.000304	0.005388
119-36-8	0.976166	0.000455	0.000205	0.023174
69-72-7	0.828779	0.001618	0.001438	0.168164
5392-40-5	0.005458	0.065849	0.906857	0.021835
101-86-0	0.003086	0.968323	0.028381	0.000210

Table 5: Predicted class by commercial software.

algorithm for multiple numbers of states and using the Akaike Information Criterion (AIC) to guide selecting the best value.

```
> cysteineFormula <- cbind(DPRACys,KEC3,KEC1.5)~1
> bioavailabilityFormula <- cbind(logKow,AUC120,Cfree)~1
> nRepsPoLCA <- 1000
> polcaTrainEvalCysteine <- poLCA(cysteineFormula,
                                  nclass = 3,
                                  nrep = nRepsPoLCA,
                                  data = discIts2TrainData,
                                  verbose = FALSE)
> polcaTrainEvalBioavailability <- poLCA(bioavailabilityFormula,
                                         nclass=3,
                                         nrep=nRepsPoLCA,
                                         data = discIts2TrainData,
                                         verbose = FALSE)
```

Table 7 shows the chemical groupings for the `Cysteine` latent variable, and

	1	2	3	4
1	6	1	0	0
2	0	4	0	0
3	0	0	4	1
4	0	0	1	4

Table 6: Confusion matrix for LLNA class predictions on test data. Rows are instances in the predicted class and columns are instances in the observed class. (n = 21).

Table 8 shows the groupings for the `Bioavailability` latent variable. The same chemicals are grouped together in both cases. However, since the variable learned by `poLCA` are unordered, the (arbitrary) labels may be different.

	1	2	3
Cluster 1	0	43	0
Cluster 2	0	0	26
Cluster 3	55	0	0

Table 7: Chemical clusters according to the Cysteine latent variable. Rows represent the classes learned by the commercial software package, and columns represent the classes found using the R package poLCA.

	1	2	3
Cluster 1	46	0	0
Cluster 2	0	0	63
Cluster 3	0	15	0

Table 8: Chemical clusters according to the Bioavailability latent variable. Rows represent the classes learned by the commercial software package, and columns represent the classes found using the R package poLCA.

To show that the arbitrary class labels have no effect on the LLNA predictions, we compile a Bayesian network using the `Bioavailability` and `Cysteine` latent variables found using `poLCA` and make predictions on the training dataset that was discretized using the cut-points found by the commercial software package.

```
> discTrainEvalpoLCA <- discIts2TrainData
> discTrainEvalpoLCA$Cysteine <- polcaTrainEvalCysteine$predclass
> discTrainEvalpoLCA$Bioavailability <- polcaTrainEvalBioavailability$predclass
> its2EvalpoLCACpt <- extractCPT.Commercial(discTrainEvalpoLCA,its2Dag)
> its2EvalpoLCAGin <- grain(compileCPT(its2EvalpoLCACpt))
> trainPredDistEvalpoLCA <-  predict.grain(its2EvalpoLCAGin,"LLNA",
                                           type="distribution",
```

```
                                                    newdata=discTrainEvalpoLCA)
>
```

The results in Table 9 are identical to those in Table 4.

CASRN	LLNA.1	LLNA.2	LLNA.3	LLNA.4
100-06-1	0.971694	0.027977	0.000272	0.000056
122-57-6	0.001713	0.000808	0.793932	0.203547
637-07-0	0.954734	0.021989	0.000309	0.022968
107-75-5	0.000160	0.947495	0.052049	0.000296
3055-86-5	0.906084	0.062214	0.002272	0.029430
121-79-9	0.000443	0.000907	0.776881	0.221769
874-23-7	0.413555	0.086810	0.499461	0.000175
2634-33-5	0.000206	0.000509	0.790698	0.208588
87-86-5	0.057037	0.116919	0.333108	0.492937
119-84-6	0.046210	0.001197	0.207419	0.745174

Table 9: Predicted class by gRain where the latent variables were determined using poLCA and discretization cut-points were determined using the commercial software.

5 Discretization and evaluation of latent variables using available R packages

The equivalence of the inference methods in `gRain` and the commercial software has been established. Next the discretization and latent variable learning are examined entirely using the `discretization` and `poLCA` packages in R.

5.1 CAIM discretization of the training data

The CAIM (class-attribute interdependence maximization) discretization algorithm is available in the `discretization` R package. Since this is a supervised discretization algorithm, meaning that it uses knowledge of the supervising variable (`LLNA` in this case) to chose optimal cut-points, it is crucial that the algorithm not "see" the test data to avoid overly optimistic predictions. CAIM maximizes the dependency relationship between the class labels and the continuous-valued attribute, simultaneously minimizing the number of discrete intervals [7]. CAIM is a "top-down" approach where a single discretization interval is iteratively divided using the boundary that gives the largest value of the CAIM criterion. The algorithm assumes that the minimum number of intervals needed is equal to the number of classes in the supervising variable (`LLNA`).

Functions in the `discretization` package require that the supervising variable is in the last column of the data.frame.

```
> caimTrainData <- trainData[,-grep("^LLNA.",colnames(trainData))]
> newColOrder <- c(names(caimTrainData)[2:ncol(caimTrainData)],"LLNA")
> newColOrder <- newColOrder[!(newColOrder %in%
                               c("Cysteine","Bioavailability","TIMES"))]
> caimTrainData <- caimTrainData[,newColOrder]
> trainCaim <- disc.Topdown(caimTrainData, method = 1)
> caimDiscList <- trainCaim$cutp
> names(caimDiscList) <- colnames(trainCaim[[2]])[colnames(trainCaim[[2]]) != "LLNA"]
> discCaimTrainData <- trainCaim$Disc.data
> discCaimTrainData$TIMES <- trainData$TIMES
```

5.2 Create the latent variables

```
> cysteinePolcaObj <- poLCA(cysteineFormula,
                            nclass=3,
                            nrep=nRepsPoLCA,
                            data=discCaimTrainData,
                            verbose=FALSE)
> bioavailabilityPolcaObj <- poLCA(bioavailabilityFormula,
                                   nclass=3,
                                   nrep=nRepsPoLCA,
                                   data=discCaimTrainData,
                                   verbose=FALSE)
> discCaimTrainData$Cysteine <- cysteinePolcaObj$predclass
> discCaimTrainData$Bioavailability <- bioavailabilityPolcaObj$predclass
```

5.3 Build the Bayesian network

```
> its2CaimCpt <- extractCPT.Commercial(discCaimTrainData,its2Dag)
> its2CaimGin <- grain(compileCPT(its2CaimCpt))
```

5.4 Predictions for the training dataset

```
> trainCaimPredClass <- predict.grain(its2CaimGin,"LLNA",type="class",
                                      newdata=discCaimTrainData)
> trainCaimPredDist <-  predict.grain(its2CaimGin,"LLNA",
                                      type="distribution",
                                      newdata=discCaimTrainData)
> trainCaimTable <- table(as.numeric(trainCaimPredClass$pred$LLNA),
                          discCaimTrainData$LLNA)
```

	1	2	3	4
1	31	2	1	2
2	3	22	2	0
3	1	3	26	5
4	1	1	6	18

Table 10: Confusion matrix for training data predictions using the CAIM discretized dataset. Rows are instances in the predicted class and columns are instances in the observed class. (n = 124).

5.5 Discretization and predictions for the test dataset

The test data are discretized using the cut-points returned by the `disc.Topdown` function. Note that the cut-points begin and end with the minimum and maximum values for that variable in the training dataset. Thus, if a value in the test dataset lies outside of this range that value will is not discretized (an "NA" will be returned). Values of the three bioavailability variables (`logKow`, `Cfree`, and `AUC120`) for imidazolidinyl urea (CASRN 39236-46-9) were outside of the cut-points found by CAIM on the training set. Also, the value of `logKow` for salicylic acid (CASRN 69-72-7) was outside of the range found by CAIM. These values were not changed, but could have been set to the lowest discrete value. Either approach gives the same result.

```
> discCaimTestData <- testData[,-grep("^LLNA.|^Bioavailability.|^Cysteine.",
                                      colnames(testData))]
> discCaimTestData <- cutData(discCaimTestData,caimDiscList,to.numeric=TRUE)


> unifLlnaCaimCpt <- its2CaimGin$cptlist
> unifLlnaCaimCpt$LLNA <- parray(c("LLNA"),
                                 list(as.character(1:4)),
                                 values=rep(.25,4))
> its2UnifCaimGin <- grain(compileCPT(unifLlnaCaimCpt))
> testCaimPredDist <-  predict.grain(its2UnifCaimGin,
                                     c("LLNA","Bioavailability","Cysteine"),
                                     type="distribution",
                                     newdata=discCaimTestData)
> testCaimPredClass <-  predict.grain(its2UnifCaimGin,
                                      c("LLNA","Bioavailability","Cysteine"),
                                      type="class",
                                      newdata=discCaimTestData)
> testCaimTable <- table(as.numeric(testCaimPredClass$pred$LLNA),
                         testData$LLNA,
                         dnn=c("predicted","observed"))
```

Table 11 shows the confusion matrix for the CAIM discretized test data.

	1	2	3	4
1	6	1	0	0
2	0	4	1	0
3	0	0	4	1
4	0	0	0	4

Table 11: Confusion matrix for test data using the CAIM discretized dataset. Rows are instances in the predicted class and columns are instances in the observed class. (n = 21).

6 Discussion

The analysis described in Section 3 was conducted to test the equivalence of the inference methods used in `gRain` and the commercial software. The discretization cut points and latent variables found using the commercial software were used to train the network using `gRain`. Under this scenario, the conditional distributions for LLNA ($Pr(LLNA|evidence)$) obtained by the both software packages were identical.

In Section 5, the CAIM algorithm implemented in the R package `discretization` was used to discretize the data and the `poLCA` package was used to learn the latent variables. The overall classification accuracies between the R-based method and the commercial software package were found to be the same, with three compounds misclassified by both methods. However, two compounds were classified differently by the two methods. Dihydroeugenol (2-methoxy-4-propyl-phenol) (CASRN 2785-87-7) was correctly classified as a moderate sensitizer by the R-based method and incorrectly classified as a strong sensitizer by the commercial software. Citral (CASRN 5392-40-5) was incorrectly classified as a weak sensitizer by the R-based method and correctly classified as a moderate sensitizer by the commercial software package. Differences in the discretization approaches are the most likely explanation for the discrepancies between the two methods. In the ITS-2 manuscript [4], variables were discretized using either a decision tree or a k-means algorithm. For some variables, additional cut-points were added manually following the initial discretization by decision tree or k-means. Here, a single supervised discretization method, the CAIM algorithm, was used. CAIM was used for its ease of application (there are no adjustable parameters) and because it generally produces small numbers of cut-points. Results from commonly used supervised discretization algorithms may be quite different in terms of both the location and number of cut points. These differences can have a significant impact on the parameterization of the Bayesian network.

With respect to the three steps shown in Figure 1, we find that if the same inputs are provided to steps 2 or 3, both methods give identical results. We expect this to hold for most discrete networks of moderate size.

7 Appendix

7.1 SessionInfo

```
> sessionInfo()
R version 3.0.2 (2013-09-25)
Platform: x86_64-w64-mingw32/x64 (64-bit)

locale:
[1] LC_COLLATE=English_United States.1252
[2] LC_CTYPE=English_United States.1252
[3] LC_MONETARY=English_United States.1252
[4] LC_NUMERIC=C
[5] LC_TIME=English_United States.1252

attached base packages:
[1] grid      stats     graphics  grDevices utils     datasets  methods
[8] base

other attached packages:
 [1] Rgraphviz_2.6.0      xtable_1.7-1         discretization_1.0-1
 [4] poLCA_1.4.1          MASS_7.3-29          scatterplot3d_0.3-34
 [7] gRain_1.2-2          gRbase_1.6-12        Rcpp_0.11.0
[10] graph_1.40.1

loaded via a namespace (and not attached):
[1] BiocGenerics_0.8.0 igraph_0.7.0       lattice_0.20-24    Matrix_1.1-2
[5] parallel_3.0.2     RBGL_1.38.0        stats4_3.0.2       tools_3.0.2
```

References

[1] Claus Dethlefsen and Søren Højsgaard. A common platform for graphical models in R: The gRbase package. *Journal of Statistical Software*, 14(17):1–12, 2005.

[2] Segey Fomel and John F. Claerbout. Reproducible research. *Computing in science engineering*, 11(1):5–7, 2009.

[3] Søren Højsgaard. Graphical independence networks with the gRain package for R. *Journal of Statistical Software*, 46(10):1–26, 2012.

[4] Joanna Jaworska, Yuri Dancik, Petra Kern, Frank Gerberick, and Andreas Natsch. Bayesian integrated testing strategy to assess skin sensitization potency: From theory to practice. *Journal of Applied Toxicology*, 33(11):1353–1364, 2013.

[5] HyunJi Kim. *discretization: Data preprocessing, discretization for classification*, 2012. R package version 1.0-1.

[6] Roger Koenker and Achim Zeileis. On reproducible econometric research. *Journal of Applied Econometrics*, 24(5):833–847, 2009.

[7] Lukasz A. Kurgan and Krzysztof J. Cios. CAIM discretization algorithm. *IEEE Transactions on Knowledge and Data Engineering*, 16(2):145–153, 2004.

[8] Friedrich Leisch. Sweave: Dynamic generation of statistical reports using literate data analysis. In Wolfgang Härdle and Bernd Rönz, editors, *Compstat 2002 — Proceedings in Computational Statistics*, pages 575–580. Physica Verlag, Heidelberg, 2002. ISBN 3-7908-1517-9.

[9] Drew A. Linzer and Jeffrey B. Lewis. poLCA: An R package for polytomous variable latent class analysis. *Journal of Statistical Software*, 42(10):1–29, 2011.

[10] Roger D. Peng. Reproducible research in biostatistics. *Biostatistics*, 10(3):405–408, 2009.

[11] W. Patrick Walters. Modeling, informatics, and the quest for reproducibility. *Journal of Chemical Information and Modeling*, 53(7):1529–1530, 2013.