RNetica Introduction

Russell Almond

6/3/2021

RNetica and Peanut Packages

The RNetica Packages

The RNetica suite consists of a number of packages:

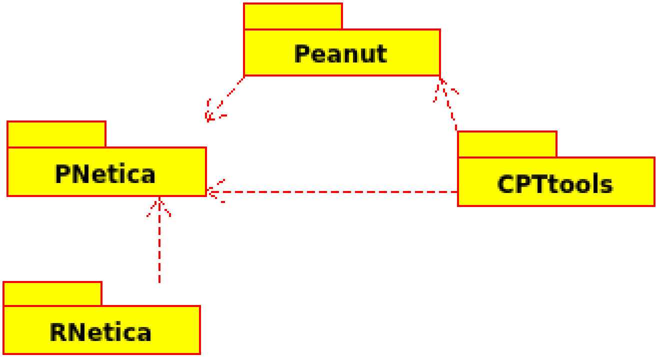


Figure 1: RNetica Package Suite

1. `CPTtools` is a collection of tools for building conditional probability tables (particularly, the DiBello models). It stands alone.

2. `RNetica` links R to the Netica Bayes net engine. Note:

Peanut packages

The Peanut object oriented framework rests on top of `CPTtools` and `RNetica`.

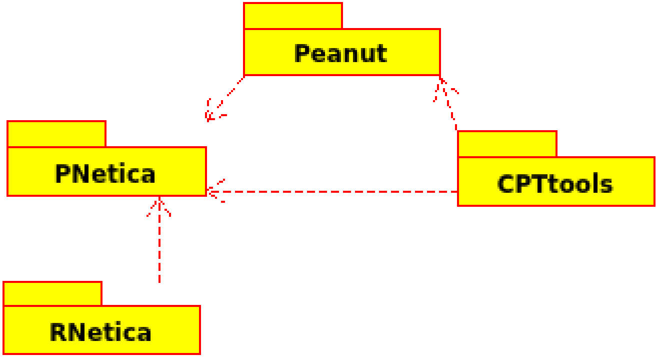


Figure 2: RNetica Package Suite

3. `Peanut` (a corrupt reading of Pnet, or parameterized network) is an object oriented frame work on top of `CPTtools`.
4. `PNetica` is an implementation of the `Peanut` framework using

Installation Preliminaries

You can install the packages from source or (Windows and Mac OS only) from precompiled binaries.

Compiling packages from source require installing the proper compilers. Instructions can be found at Windows or Mac OS.

The following packages are used by the PNetica software and should be installed (usual CRAN mirror) before starting.

```r
install.packages(c("R.utils","futile.logger",
                   "shiny","shinyjs", "ggplot2", "lattice")
                 repos="https://cloud.r-project.org/")
```

You can also install these packages using the RStudio `Tools > Install Packages` menu item.

Installation Method 1 – Precompiled tarballs

This method is available only for Windows and Mac OS.

Download the compiles packages (`.zip` files for Windows, `.tgz` files for MacOS) from https://pluto.coe.fsu.edu/RNetica (follow the links there for the latest version). Make sure to grab all four tarballs.

Windows : CPTtools_0.7-2.zip RNetica_0.8-4.zip Peanut_0.8-4.zip PNetica_0.8-5.zip

MacOS: CPTtools_0.7-2.tgz RNetica_0.8-4.tgz Peanut_0.8-4.tgz PNetica_0.8-5.tgz

install.packages and R CMD INSTALLATION

To install the packages using R Studio, select the “Tools > Install Packages” menu item, then select “Local Package Archive” from the first “Install from” drop down.

You can also use this command:

```
install.packages(file.choose())
```

Finally, if you open a terminal window, you can run the command `R CMD INSTALL <packagefile>` to install the package.

Packages must be installed in order!

Installation Method II – Source and Github

All of the packages are currently available on Github.

Installing RNetica from github requires the appropriate compilers (Rtools or Xcode) installed on your system.

```
#uncomment first line if needed.
#install.packages("devtools",repos="https://cloud.r-projec
library(devtools)
install_github("ralmond/CPTtools")
install_github("ralmond/RNetica")
install_github("ralmond/Peanut")
install_github("ralmond/PNetica")
```

Netica and R

License

- R – GPL-3 (Free and open source)
- RNetica – Artistic (Free and open source)
- Netica.dll/libNetica.so– Commercial (open API, but not open source)
  - Free Student/Demo version
    - Limited number of nodes
    - Limited usage (education, evaluation of Netica)
  - Paid version (seehttp://www.norsys.com/for price information)
    - Need to purchase API not GUI version ofNetica
    - May want both (use GUI to visualize networks built in RNetica)
- CPTtools, Peanut – Artistic (Free and open source), does not depend on Netica
- RNetica – Artistic, but depends on RNetica.

Installing the License Key

- When you purchase a license, Norsys will send you a license key. Something that looks like:

  "+Course/FloridaSU/Ex15-05-30,120,310/XXXXX"

  (Where I've obscured the last 5 security digits)

To install the license key, start R in your project directory and type:

```
NeticaLicenseKey <- "+Course/FloridaSU/Ex15-05-30,120,310/
q("yes")
```

Can also put first line in file in your home directory and run `source("~/NeticaLicense.R")` every time you run RNetica.

Starting RNetica

After you load RNetica you need to start the session. This is when you pass the license key.

```
library(RNetica)
sess <- NeticaSession(LicenseKey=NeticaLicenseKey)
startSession(sess)
```

```
library(RNetica)
```

```
## Loading required package: CPTtools
```

```
sess <- NeticaSession()
startSession(sess)
```

```
## Netica 6.07 Linux (AFCl64), (C) 1992-2019 Norsys Softwar
##
## Netica is operating without a password; there are some l
```

Everything in this tutorial should run without the license.

When to use the session object.

- When starting/restarting Netica
- When creating a network, or reading one from a file.
- When searching for networks.
- Certain global properties

`NeticaBN` objects have a `$session` proprty which points back to the session.

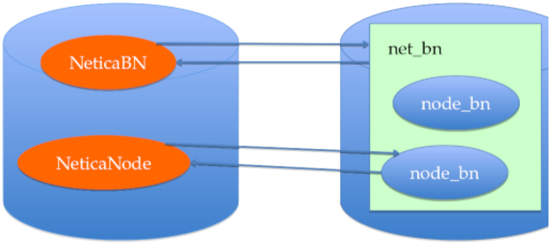
`NeticaNode` objects have a `$node` property which points back to the network (which points to the session).

The R heap and the Netica heap

R and Netica have two different workspaces (memory heaps)

R workspace is saved and restored automatically when you quick and restart R.

Neticaheap must be reconnected manually.



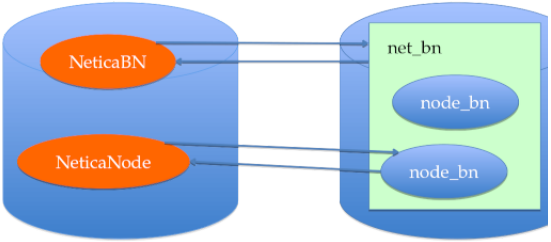
Active and Inactive pointers

When RNetica creates/finds aNeticaobject it creates a corresponding R object

If the R object is active then it points to the Netica object, and the Netica object points back at it.

If the pointer gets broken (saving and restarting R, deleting the network/node then the R object becomes inactive.

The function `is.active(nodeOrNet)` test to see if the node/net is active.



Mini-ACED

Mini-ACED Proficiency model

Subset of ACED network: Shute, Hansen & Almond (2008);
http://ecd.ralmond.net/ecdwiki/ACED

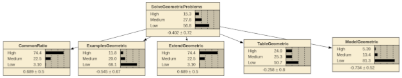


Figure 5: Proficiency Model

Mini-ACED EM Fragments

All ACED tasks were scored correct/incorrect

Each evidence model is represented by a fragment consisting of observables with *stub* edges indicating where it should be *adjoined* with the network.

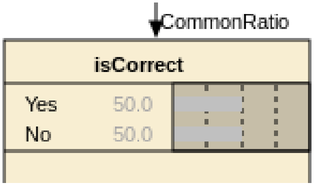
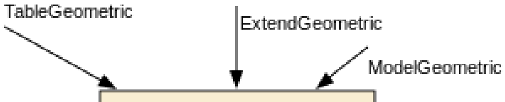


Figure 6: Common Ratio Easy



Task to EM map

Need a table to tell us which EM to use with which task

```
## Read in task->evidence model mapping
EMtable <- read.csv("miniACED/MiniACEDEMTable.csv",row.name
                    as.is=2) #Keep EM names as strings
EMtable
```

```
##                                                  EM    X
## tCommonRatio1a                   CommonRatioEasyEM  108
## tCommonRatio1b                   CommonRatioEasyEM  108
## tCommonRatio2a                    CommonRatioMedEM  108
## tCommonRatio2b                    CommonRatioMedEM  108
## tCommonRatio3a                   CommonRatioHardEM  108
## tCommonRatio3b                   CommonRatioHardEM  108
## tExamplesGeometric1a                ExamplesEasyEM  342
## tExamplesGeometric1b                ExamplesEasyEM  342
## tExamplesGeometric2a                 ExamplesMedEM  342
## tExamplesGeometric2b                 ExamplesMedEM  342
## tExamplesGeometric3a                ExamplesHardEM  342
```

Scoring Script

Preliminaries

The following script assumes that the data files it needs are in the folder `miniACED` and that this folder is in the same directory as this `Rmd` file.

Don't forget to `setwd()` to the `miniACED` folder (as it needs to find its networks).

```r
## Scoring Script
## Preliminaries
library(RNetica)

sess <- NeticaSession()
startSession(sess)
```

```
## Netica environment is already initialized
```

Read in the Network.

```
## Read in network -- Do this every time R is restarted
profModel <- ReadNetworks("miniACED/miniACEDPnet.dne",sess
## If profModels already exists could also use

## Reconnect nodes -- Do this every time R is restarted
allNodes <- NetworkAllNodes(profModel)
sgp <- allNodes$SolveGeometricProblems
sgp
```

```
## Discrete  Netica Node named  SolveGeometricProblems in n
##   Node is currently active.
## States are:  High, Medium, Low
```

Aside 1 – Node Sets

Node sets can be viewed as either

A. a set of labels assigned to each node.

B. a set of nodes which have a particular label.

In RNetica, these are very useful as they define collections of nodes that might be interesting in some way (e.g., Proficiency variables, Observable variable, background variables)

Node set operations yeild a list of nodes; iterating over that set is often very useful.

Node Set Examples

```r
## Node Sets
NetworkNodeSets(profModel)
```

```
## [1] "pnodes"        "Proficiencies"
```

```r
NetworkNodesInSet(profModel,"pnodes")
```

```
## $TableGeometric
## Discrete  Netica Node named  TableGeometric in network
##   Node is currently active.
## States are:  High, Medium, Low
##
## $ModelGeometric
## Discrete  Netica Node named  ModelGeometric in network
##   Node is currently active.
## States are:  High, Medium, Low
##
## $ExtendGeometric
## Discrete  Netica Node named  ExtendGeometric in network
```

More Node Set Examples

```r
profNodes <- NetworkNodesInSet(profModel,"Proficiencies")
NodeSets(sgp)
```

```
## [1] "pnodes"        "Proficiencies"
```

Adding a node to a set.

```r
## These are all settable
NodeSets(sgp) <- c(NodeSets(sgp),"HighLevel")
NodeSets(sgp)
```

```
## [1] "HighLevel"     "pnodes"        "Proficiencies"
```

Aside 2: Common Net operations

Just about everything that can be done through the Netica GUI, can be done through the Netica API, and hence through R Netica. [In practice, the API version has lagged the GUI version, and my RNetica release lag Norsys's API updates.] Many more examples are in the RNetica help.

```
## Querying Nodes
NodeStates(sgp)   #List states
```

```
##     High   Medium      Low
##   "High" "Medium"    "Low"
```

```
NodeParents(sgp)  #List parents
```

```
## named list()
```

More RNetica Queries

```
NodeLevels(sgp)   #List numeric values associated with sta
```

```
##       High     Medium        Low
##  0.9674216  0.0000000 -0.9674216
```

```
NodeProbs(sgp) # Conditional Probability Table (as array)
```

```
## SolveGeometricProblems
##   High Medium    Low
## 0.1532 0.2784 0.5684
## attr(,"class")
## [1] "CPA"   "array"
```

```
## These are all settable (can be used on RHS of <-) for mo
## construction
```

Conditional Probability Tables (as Data Frame)

```
sgp[] # Conditional Probability Table (as data frame)
```

```
##   SolveGeometricProblems.High SolveGeometricProblems.Mec
##                        0.1532                          0.2
##    SolveGeometricProblems.Low
##                        0.5684
```

Can use [] operator to select rows or elements

Can set table or (row or cell).

`CPTtools` package has tools for building tables.

```
help(package="CPTtools")
```

Inference

Networks must be *compiled* before they are used for inference.

```
## Inference
CompileNetwork(profModel) #Lightning bolt on GUI
## Must do this before inference
## Recompiling an already compiled network is harmless
```

```
## Enter Evidence by setting values for these functions
NodeValue(sgp) #View or set the value
```

```
## [1] NA
```

```
NodeLikelihood(sgp) #Virtual evidence
```

```
##   High Medium    Low
##      1      1      1
```

Beliefs (Marginal Probabilities)

```
## Query beliefs
NodeBeliefs(sgp) #Current probability (given entered evide
```

```
##   High Medium    Low
## 0.1532 0.2784 0.5684
```

```
NodeExpectedValue(sgp) #If node has values, EAP
```

```
## [1] -0.4016734
## attr(,"std_dev")
## [1] 0.7169429
```

```
## These aren't settable
```

```
## Retract Evidence
RetractNodeFinding(profNodes$ExamplesGeometric)
RetractNetFindings(profModel)
```

Example: Enter Evidence

```
## Enter Evidence
NodeFinding(profNodes$CommonRatio) <- "Medium"
## Enter Evidence "Not Low" ("High or Medium")
NodeLikelihood(profNodes$ExamplesGeometric) <- c(1,1,0)

NodeBeliefs(sgp) #Current probability (given entered evide
```

```
##      High    Medium       Low
## 0.0000000 0.1811515 0.8188485
```

```
NodeExpectedValue(sgp) #If node has values, EAP
```

```
## [1] -0.7921717
## attr(,"std_dev")
## [1] 0.3725963
```

Example: Retract Evidence

```r
## Retract Evidence
RetractNetFindings(profModel)
```

Many more examples:

```r
help(RNetica)
```

Back to work

Simple Scoring Example

Start New Student Copy the *proficiency model* to make *student model*.

```
Fred.SM <- CopyNetworks(profModel,"Fred")
Fred.SMvars <- NetworkAllNodes(Fred.SM)
CompileNetwork(Fred.SM)
```

Setup score history.

```
prior <- NodeBeliefs(Fred.SMvars$SolveGeometricProblems)
Fred.History <- matrix(prior,1,3)
row.names(Fred.History) <- "*Baseline*"
colnames(Fred.History) <- names(prior)
Fred.History
```

```
##               High Medium    Low
## *Baseline* 0.1532 0.2784 0.5684
```

Fred does a task

Task name and data.

```
t.name <- "tCommonRatio1a"
t.isCorrect <- "Yes"
```

Adjoin SM and EM

```
EMnet <- ReadNetworks(file.path("miniACED",
     paste(EMtable[t.name,"EM"],"dne",sep=".")),
                           session = sess)
obs <- AdjoinNetwork(Fred.SM,EMnet)
names(NetworkAllNodes(Fred.SM))
```

```
## [1] "SolveGeometricProblems" "CommonRatio"            "F
## [4] "ExtendGeometric"        "ModelGeometric"         "T
## [7] "isCorrect"
```

```
## Fred.SM is now the Motif for the current task.
CompileNetwork(Fred.SM)
```

Absorb Evidence

Enter finding

```
NodeFinding(obs$isCorrect) <- t.isCorrect
```

Calculate statistics of interest

```
post <- NodeBeliefs(Fred.SMvars$SolveGeometricProblems)
Fred.History <- rbind(Fred.History,new=post)
rownames(Fred.History)[nrow(Fred.History)] <- paste(t.name
Fred.History
```

```
##                       High    Medium       Low
## *Baseline*        0.153200 0.2784000 0.5684000
## tCommonRatio1a=Yes 0.160016 0.2893454 0.5506387
```

Cleanup

Network and Observable no longer needed, so absorb it:

```
DeleteNetwork(EMnet) ## Delete EM
try(AbsorbNodes(obs))
## Currently, there is a Netica bug with Absorb Nodes, we w
## leave this node in place, as that is mostly harmless.
```

2nd Task

Write a script for scoring the second task.
This time Fred attempts the task `tCommonRatio2a` and gets it incorrect.

```
### Fred does another task
t.name <- "tCommonRatio2a"
t.isCorrect <- "No"

## Load Evidence Model and adjoin

## Recompile

## Add Evidence

## Check Finding and add to history

## Clean up
```

Answer for 2nd Task

```
### Fred does another task
t.name <- "tCommonRatio2a"
t.isCorrect <- "No"

EMnet <- ReadNetworks(file.path("miniACED",
                      paste(EMtable[t.name,"EM"],"dne",
                            sep=".")),
                      session=sess)
obs <- AdjoinNetwork(Fred.SM,EMnet)
#NodeVisPos(obs$isCorrect) <- EMtable[t.name,c("X","Y")]
## Fred.SM is now the Motif for the current task.
CompileNetwork(Fred.SM)

NodeFinding(obs[[1]]) <- t.isCorrect
post <- NodeBeliefs(Fred.SMvars$SolveGeometricProblems)
Fred.History <- rbind(Fred.History,new=post)
rownames(Fred.History)[nrow(Fred.History)] <-
     paste(t.name,t.isCorrect,sep="=")
```

Fred does another task

```
t.name <- "tCommonRatio2a"
t.isCorrect <- "No"
```

```
EMnet <- ReadNetworks(file.path("miniACED",
                paste(EMtable[t.name,"EM"],"dne",sep=".")),
                session=sess)
obs <- AdjoinNetwork(Fred.SM,EMnet)
(NetworkAllNodes(Fred.SM)) ## Fred.SM is now the Motif for
```

```
## $SolveGeometricProblems
## Discrete  Netica Node named  SolveGeometricProblems in n
##   Node is currently active.
## States are:  High, Medium, Low
##
## $CommonRatio
## Discrete  Netica Node named  CommonRatio in network  Fre
##   Node is currently active.
## States are:  High, Medium, Low
##
```

Task 2 continued

```
NodeFinding(obs[[1]]) <- t.isCorrect
post <- NodeBeliefs(Fred.SMvars$SolveGeometricProblems)
Fred.History <- rbind(Fred.History,new=post)
rownames(Fred.History)[nrow(Fred.History)] <- paste(t.name
Fred.History
```

```
##                           High    Medium       Low
## *Baseline*          0.15320002 0.2784000 0.5684000
## tCommonRatio1a=Yes 0.16001597 0.2893454 0.5506387
## tCommonRatio2a=No  0.10649123 0.2057332 0.6877756
## tCommonRatio2a=No  0.04991533 0.1159301 0.8341546
```

Cleanup: Delete EM and Absorb observables

```
DeleteNetwork(EMnet) ## Delete EM
try(AbsorbNodes(obs))
## Currently, there is a Netica bug with Absorb Nodes, we
##this the node in place as that is mostly harmless.
```

Fred logs out

Save network to a file.

```
WriteNetworks(Fred.SM,"FredSM.dne")
DeleteNetwork(Fred.SM)
is.active(Fred.SM)  ## No longer active in Netica space
```

```
## [1] FALSE
```

Fred logs back in

```
Fred.SM <- ReadNetworks("FredSM.dne",session=sess)
is.active(Fred.SM)
```

```
## [1] TRUE
```

Score an entire set of cases.

Read in the scores.

```
miniACED.data <- read.csv("miniACED/miniACED-Geometric.csv'
head(miniACED.data)
```

```
##      Class Treatment Sequencing Feedback Total.Items Co
## S055     1         1          2        2          63
## S058     1         1          2        2          63
## S053     1         2          2        1          63
## S061     1         1          2        2          63
## S063     1         1          2        2          63
## S066     1         1          2        2          63
##      Remaining tCommonRatio1a tCommonRatio1b tCommonRat
## S055         0              1              1
## S058         0              2              2
## S053         0              1              1
## S061         0              1              1
## S063         0              1              2
## S066         0              1              2
##      tCommonRatio2a tCommonRatio2b tExamplesGeometric1a
## S055              1              1                    1
```

Setup for student in sample

Create Student Model from Proficiency Model

```
Student.SM <- CopyNetworks(profModel,"Student")
Student.SMvars <- NetworkAllNodes(Student.SM)
CompileNetwork(Student.SM)
```

Initialize history list

```
prior <- NodeBeliefs(Student.SMvars$SolveGeometricProblems
Student.History <- matrix(prior,1,3)
row.names(Student.History) <- "*Baseline*"
colnames(Student.History) <- names(prior)
```

Now loop over tasks

```
for (itask in first.task:last.task) {

  ## Look up the EM for the task, and adjoin it.
  tid <- names(miniACED.data)[itask]
  EMnet <- ReadNetworks(file.path("miniACED",
                         paste(EMtable[tid,"EM"],"dne",sep='
                         session=sess)
  obs <- AdjoinNetwork(Student.SM,EMnet)
  CompileNetwork(Student.SM)

  ## Add the evidence
  t.val <- t.vals[miniACED.data[Student.row,itask]] #Decod
  NodeFinding(obs[[1]]) <- t.val

  ## Update the history
  post <- NodeBeliefs(Student.SMvars$SolveGeometricProblems
  Student.History <- rbind(Student.History,new=post)
  rownames(Student.History)[nrow(Student.History)] <- paste
```

Now look at the scoring history.

```
Student.History
```

```
##                                              High        Mediu
## *Baseline*                            1.532000e-01 0.27840000З
## tCommonRatio1a=No                     6.939758e-02 0.14382691
## tCommonRatio1b=No                     1.311332e-02 0.04543131
## tCommonRatio3a=Yes                    5.764372e-02 0.12699648
## tCommonRatio3b=No                     3.833026e-02 0.09589105
## tCommonRatio2a=No                     1.222626e-02 0.05103556
## tCommonRatio2b=No                     3.319675e-03 0.03242401
## tExamplesGeometric1a=No               1.006431e-03 0.02298102
## tExamplesGeometric1b=No               6.250255e-04 0.02056839
## tExamplesGeometric3a=No               6.051398e-04 0.02043074
## tExamplesGeometric3b=No               5.891956e-04 0.02031316
## tExamplesGeometric2a=No               5.483014e-04 0.01998808
## tExamplesGeometric2b=No               5.246417e-04 0.01979494
## tExtendGeometric1a=No                 1.739116e-04 0.00748352
## tExtendGeometric1b=No                 2.761918e-05 0.00198672
## tExtendGeometric3a=Yes                1.395514e-04 0.00638344
```

Weight of Evidence

Weight of Evidence

Good 1985

H is binary hypothesis, e.g., *Proficiency* $>$ *Medium*

E is evidence for hypothesis

Weight of Evidence *WOE* is

$$W(H:E) = \log \frac{P(E|H)}{P(E|\overline{H})} = \log \frac{P(H|E)}{P(\overline{H}|E)} - \log \frac{P(H)}{P(\overline{H})}$$

Conditional Weight of Evidence

$$W(H : E_2|E_1) = \log \frac{P(E_2|H, E_1)}{P(E_2|\overline{H}, E_1)}$$

Additive properties

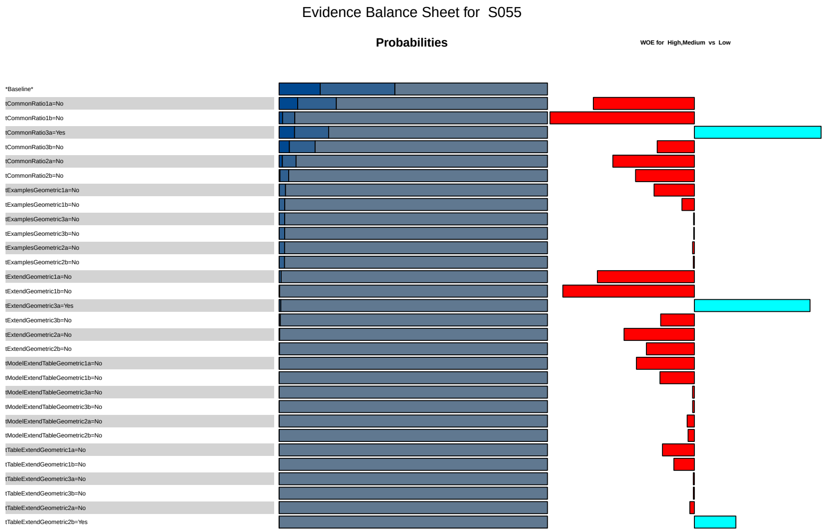
$$W(H : E_1, E2) = W(H : E_1) + W(H : E_2|E_1)$$

Order senstive (evidence seen earlier is worth more)

WOE Balance Sheet:

Now examine scoring history

```
woeBal(Student.History,c("High","Medium"),"Low",
       title=paste("Evidence Balance Sheet for ",
                   rownames(miniACED.data)[Student.row]))
```



For More information

```r
help(RNetica)
help(package="RNetica")
help(CPTtools)
help(package="CPTtools")
```