RNetica Basics

Russell Almond

March 18, 2019

Four Packages

- RNetica – Low-level R/Netica Interface
- CPTtools – Design patterns for Conditional Probability Tables
  - Independent of other packages
- Peanut – Object-oriented parameterized Bayesian Networks
  - Implementation independent
  - Uses CPTtools
- PNetica – RNetica implementation of Peanut protocols

https://pluto.coe.fsu.edu/RNetica

RNetica License Agreements

- R: GPL-3 (Free and open source)
- RNetica, CPTtools, Peanut, PNetica: Artistic-2.0 (Free and open source)
- Netica.dll/libNetica.so: Commercial (openAPI, but not open source)
  - Free Student/Demo Version
    - Limited number of nodes.
    - Limited Usage (education, evaluation of Netica)
  - Paid version (see http://www.norsys.com/ for price information).
    - Need to purchase API, not GUI verison of Netica for RNetica
    - May want GUI for network visualization.
  - Netica.dll/libNetica.so included with binary distributions by permission of Norsys
    - License key still must be purchased.

Installing non-CRAN packages

- Windows and MacOS X: Download object package from pluto
  - Windows: RNetica-0.5-2.zip

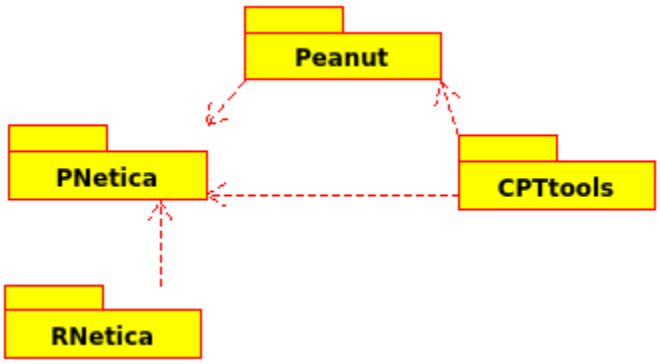


Figure 1: Peanut Package Dependencies

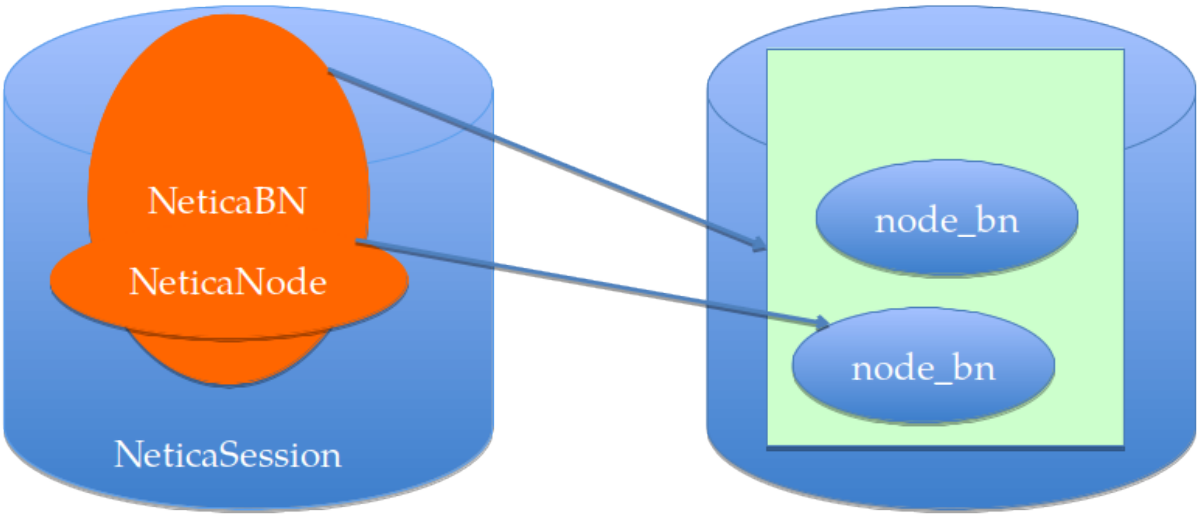


Figure 2: Netica Session and Netica Heap

  - MacOS: RNetica-0.5-2.tgz
- RStudio: Use Tools> Install Packages . . . , select "Install From" = "Package Archive File"
- install.packages("RNetica-0.5-2.zip", repos=NULL)
- Source Installation:
  - Download Source tarball: RNetica-0.5-2.tar.gz
  - Download and unpack Netica C API from https://norsys.com/
  - From Command Line:
    `R CMD INSTALL RNetica --configure-args='--with-netica=/path/to/Netica_API_504'`
  - Windows, need to set a NETICA_HOME environmental variable, see INSTALLATION FILE.
- Now load RNetica

Netica Session: R and Netica Memory

- R and Netica have two different workspaces (memory heaps)
- R workspace is saved and restored automatically when you quick and restart R.
- Netica heap must be reconnected manually.
- Netica Session object
  - Provides link to Netica Memory Space
  - Is a container for all `NeticaNetwork` objects
  - Must be reconnected every R session, using `startSession()` ## Installing a 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:

```
#DefaultNeticaSession <- NeticaSession(LicenseKey="+Course/FloridaSU/Ex15-05-30,120,310/XXXXX")
#q("yes")
```

Restart R and type

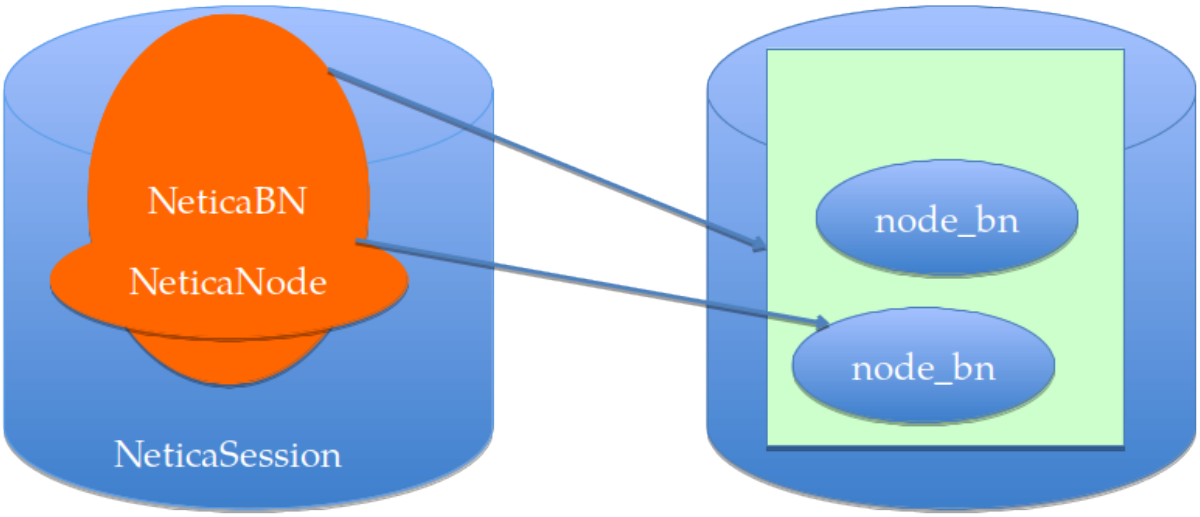


Figure 3: Netica Session and Netica Heap

```
#library(RNetica)
#startSession(DefaultNeticaSession)
```

- If license key is not installed, then you will get the limited/student mode. Most of these examples will run

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

`NeticaNetwork` 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).

Active and Inactive Pointers

- When RNetica creates/finds a Netica object it creates a corresponding R object
- R NeticaBN objects live in the NeticaSession object.
  - R NeticaNode objects live in the NeticaBN.
- If the pointer gets broken (saving & restarting R, deleting the network/node) then the R object becomes inactive.
- The function `is.active()` tests to see if the node/net/session is active

A simple example

A subset of the ACED (http://ecd.ralmond.net/ecdwiki/ACED/ACED/) network.

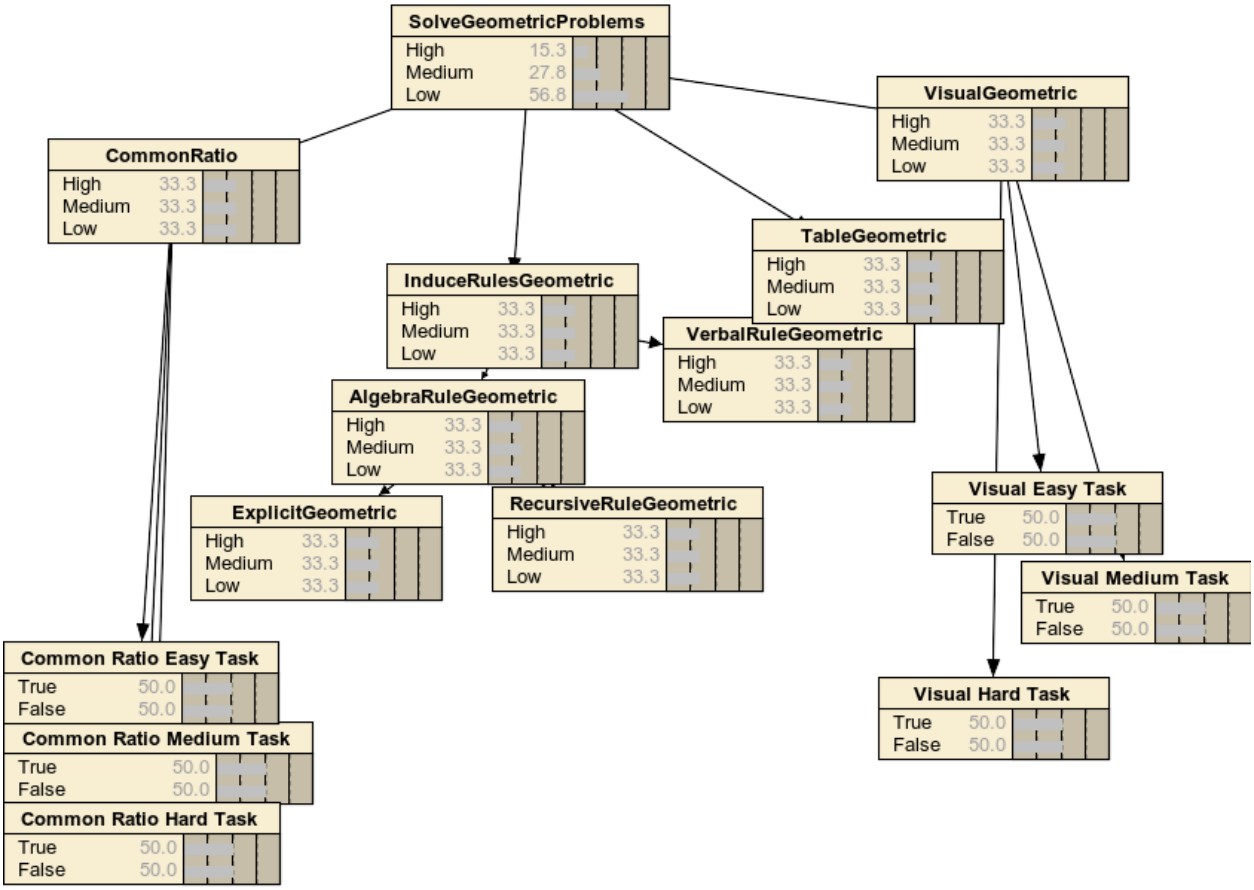


Figure 4: ACED Motif 1

Starting The Netica Interface

Load Libraries

```
library(RNetica)
library(CPTtools)
```

```
## Loading required package: lattice
```

Create a session object and start it.

Session object can be reused across workspaces, but it needs to be started again everytime R is connected.

```
sess <- NeticaSession() # Could add License Key as argument)
startSession(sess)
```

```
## Netica 5.04 Linux (AF), (C) 1992-2012 Norsys Software Corp.
##
## Netica operating without a password; there are some limitations.
```

Must start a session every time R is started. As a fallback, RNetica, will look for an object called `DefaultNeticaSession` and try to start that.

Reloading Nets and Nodes

Note that load networks needs a session as an argument (as networks are created within a session).

Networks must be reloaded every time R is started.

Note also, that the nodes must be reconnected to R objects. `NetworkAllNodes()` is the easiest way of doing this.

```
## Read in Network -- Do this every time R is started.
##DeleteNetwork(motif1)
motif1 <- ReadNetworks("../Nets/ACEDMotif1.dne",session = sess)

## Reconnect nodes -- Do this every time R is restarted
m1.nodes <- NetworkAllNodes(motif1)
m1.sgp <- m1.nodes$SolveGeometricProblems
m1.obs <- NetworkNodesInSet(motif1,"Observables")
m1.obs
```

```
## $X3Hard1
## Discrete  Netica Node named  X3Hard1 in network  GeometricMotif
##   Node is currently active.
## States are:  True, False
##
## $X3Medium1
## Discrete  Netica Node named  X3Medium1 in network  GeometricMotif
##   Node is currently active.
## States are:  True, False
##
## $X3Easy
## Discrete  Netica Node named  X3Easy in network  GeometricMotif
##   Node is currently active.
## States are:  True, False
##
## $X3Hard
```

```
## Discrete  Netica Node named  X3Hard in network  GeometricMotif
##   Node is currently active.
## States are:  True, False
##
## $X3Medium
## Discrete  Netica Node named  X3Medium in network  GeometricMotif
##   Node is currently active.
## States are:  True, False
##
## $CREasy
## Discrete  Netica Node named  CREasy in network  GeometricMotif
##   Node is currently active.
## States are:  True, False
```

Node Sets

Netica defines a node set functionality which Adds a collection of labels (sets) to each node Defines a collection of nodes with that label Netica GUI really only offers the opportunity to color nodes by set RNetica can loop over node sets (lists of nodes)

```
## Node Sets
NetworkNodeSets(motif1)
```

```
## [1] "Observables"   "Proficiencies"
```

```
NetworkNodesInSet(motif1,"Proficiencies")
```

```
## $RecursiveRuleGeometric
## Discrete  Netica Node named  RecursiveRuleGeometric in network  GeometricMotif
##   Node is currently active.
## States are:  High, Medium, Low
##
## $ExplicitGeometric
## Discrete  Netica Node named  ExplicitGeometric in network  GeometricMotif
##   Node is currently active.
## States are:  High, Medium, Low
##
## $VerbalRuleGeometric
## Discrete  Netica Node named  VerbalRuleGeometric in network  GeometricMotif
##   Node is currently active.
## States are:  High, Medium, Low
##
## $AlgebraRuleGeometric
## Discrete  Netica Node named  AlgebraRuleGeometric in network  GeometricMotif
##   Node is currently active.
## States are:  High, Medium, Low
##
## $InduceRulesGeometric
## Discrete  Netica Node named  InduceRulesGeometric in network  GeometricMotif
##   Node is currently active.
## States are:  High, Medium, Low
##
## $TableGeometric
## Discrete  Netica Node named  TableGeometric in network  GeometricMotif
##   Node is currently active.
```

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

```
NodeSets(m1.sgp)
```

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

```
## These are all settable
NodeSets(m1.sgp) <- c(NodeSets(m1.sgp),"HighLevel")
NodeSets(m1.sgp)
```

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

RNetica Functions

```
## Querying Nodes
NodeStates(m1.sgp)   #List states
```

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

```
NodeParents(m1.sgp)  #List parents
```

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

```
NodeLevels(m1.sgp)   #List numeric values associated with states
```

```
##   High Medium    Low
##     NA     NA     NA
```

```
NodeProbs(m1.sgp) # Conditional Probability Table (as array)
```

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

Can use the [] notation for viewing, setting CPTs

```
m1.sgp[] # Conditional Probability Table (as data frame)
```

```
##   SolveGeometricProblems.High SolveGeometricProblems.Medium
##                        0.1532                         0.2784
```

```
##    SolveGeometricProblems.Low
##                        0.5684
```

```
## These are all settable (can be used on RHS of <-) for
## model construction
motif1$nodes$X3Medium[]
```

```
##    theta X3Medium.True X3Medium.False
## 1   High        0.7246         0.2754
## 2 Medium        0.5000         0.5000
## 3    Low        0.2754         0.7246
```

Inference

Before doing inference, must compile the network. This is the lightning bolt icon on the GUI.

```
CompileNetwork(motif1) #Lightning bolt on GUI
## Must do this before inference
## Recompiling an already compiled network is harmless
```

```
## [1] "@NO FINDING"
```

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

```
## Query beliefs
NodeBeliefs(m1.sgp) #Current probability (given entered evidence)
```

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

If we associate numeric values with a node, we can calculate expected values.

```
NodeLevels(m1.sgp) <- c(-1,0,1)
NodeExpectedValue(m1.sgp)
```

```
## [1] 0.4152
## attr(,"std_dev")
## [1] 0.7410863
```

Enter Evidence

Direct Evidence can be entered with the setter method of `NodeFinding`

```
NodeStates(m1.obs$X3Medium)
```

```
##    True   False
##  "True" "False"
```

```
NodeFinding(m1.obs$X3Medium) <- "True"
```

Can also enter virtual evidence in the form of a likelihood.

```
## Enter Evidence "Not Low" ("High or Medium")
NodeLikelihood(motif1$nodes$TableGeometric) <- c(1,1,0)
```

Query the nodes again, and the values should have changed.

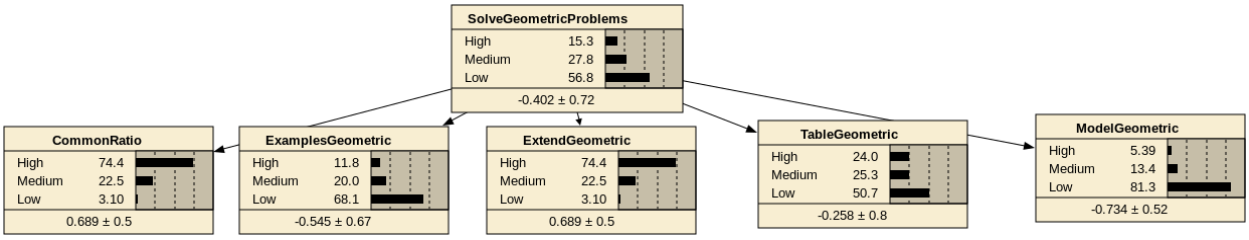


Figure 5: miniACED Proficiency Model

```
NodeBeliefs(m1.sgp) #Current probability (given entered evidence)
```

```
##      High    Medium       Low
## 0.3052434 0.3976996 0.2970570
```

```
NodeExpectedValue(m1.sgp) #If node has values, EAP
```

```
## [1] -0.00818643
## attr(,"std_dev")
## [1] 0.776037
```

Retract Evidence

Can retract evidence node by node, or on entire network.

```
RetractNodeFinding(m1.obs$X3Medium)
RetractNetFindings(motif1)
```

Query the nodes again, and the values should have returned to the baselines.

Query the nodes again, and the values should have changed.

```
NodeBeliefs(m1.sgp) #Current probability (given entered evidence)
```

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

```
NodeExpectedValue(m1.sgp) #If node has values, EAP
```

```
## [1] 0.4152
## attr(,"std_dev")
## [1] 0.7410863
```

Many more examples

```
help(RNetica)
```

Mini-ACED Proficiency Model

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

Mini-ACED EM Fragments

- All ACED tasks were scored correct/incorrect

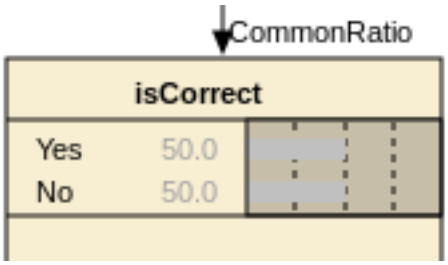


Figure 6: Common Ratio Easy Fragment

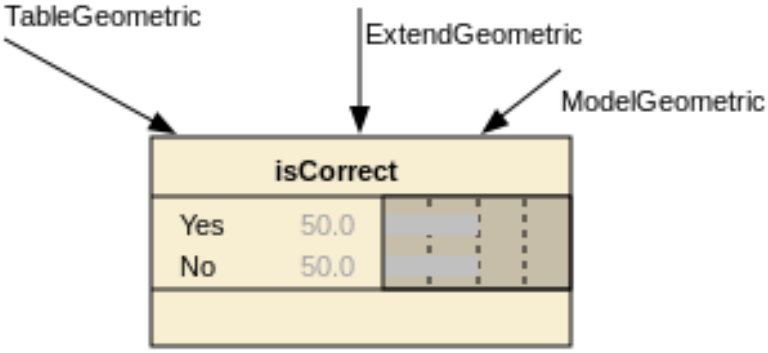


Figure 7: Model Extend Table Hard Fragment

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

Common Ratio Easy:

Model Extend Table Hard:

Task to EM Map

- Need a table to tell which EM to use with which task
- Row names are Task IDs
- EM column contains evidence model name.
- EM filename has suffix ".dne" attached.

```
## Read in task->evidence model mapping
EMtable <- read.csv("../Nets/MiniACEDEMTable.csv",row.names=1,
                    as.is=2) #Keep EM names as strings
head(EMtable)
```

```
##                               EM   X   Y
## tCommonRatio1a CommonRatioEasyEM 108 294
## tCommonRatio1b CommonRatioEasyEM 108 414
## tCommonRatio2a  CommonRatioMedEM 108 534
## tCommonRatio2b  CommonRatioMedEM 108 654
## tCommonRatio3a CommonRatioHardEM 108 774
## tCommonRatio3b CommonRatioHardEM 108 894
```

Scoring Script

1. Start RNetica Session
2. Open Profiency Model netowrk
3. Load EM to net table.

When Student evidence comes in: 1. Find SM for that student (clone PM if no SM) 2. Find EM for the task 3. Adjoin the two models. 4. Instantiate observable variables. 5. Drop EM. 6. Record Statistics & save SM

Load the Proficiency Model

```
## Read in network - Do this every time R is restarted
profModel <- ReadNetworks("../Nets/miniACEDPnet-1.dne",session=sess)
## If profModels already exists could also use
## ReadNewtorks(profModel)

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

A student walks into the test center ...

- Student gives the name "Fred"
- Student is the right grade/age for ACED (8th or 9th grader, pre-algebra)
- Bayes net has three states
  - Fred logs into ACED
  - Fred attempts the task `tCommonRatio1a` and gets it right
  - Fred attempts the task `tCommonRatio2a` and gets it wrong

Start a New Student

```
## Copy the master 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* 0.3333333 0.3333333 0.3333334
```

Score 1st Task

```
### Fred does a task
t.name <- "tCommonRatio1a"
t.isCorrect <- "Yes"

## Adjoin SM and EM
EMnet <- ReadNetworks(file.path("..","Nets",
                    paste(EMtable[t.name,"EM"],"dne",sep=".")),
                    session=sess)
obs <- AdjoinNetwork(Fred.SM,EMnet)
#NodeVisPos(obs$isCorrect) <- EMtable[t.name,c("X","Y")]
NetworkAllNodes(Fred.SM)
```

```
## $SolveGeometricProblems
## Discrete  Netica Node named  SolveGeometricProblems in network  Fred
##   Node is currently active.
## States are:  High, Medium, Low
##
## $CommonRatio
## Discrete  Netica Node named  CommonRatio in network  Fred
##   Node is currently active.
## States are:  High, Medium, Low
##
## $ExamplesGeometric
## Discrete  Netica Node named  ExamplesGeometric in network  Fred
##   Node is currently active.
## States are:  High, Medium, Low
##
## $ExtendGeometric
## Discrete  Netica Node named  ExtendGeometric in network  Fred
##   Node is currently active.
## States are:  High, Medium, Low
##
## $ModelGeometric
## Discrete  Netica Node named  ModelGeometric in network  Fred
##   Node is currently active.
## States are:  High, Medium, Low
##
## $TableGeometric
## Discrete  Netica Node named  TableGeometric in network  Fred
##   Node is currently active.
## States are:  High, Medium, Low
##
## $isCorrect
## Discrete  Netica Node named  isCorrect in network  Fred
##   Node is currently active.
## States are:  Yes, No
```

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

## Enter finding
NodeFinding(obs$isCorrect) <- t.isCorrect
```

Stats for 1st Task

```
##                          High    Medium       Low
## *Baseline*          0.3333333 0.3333333 0.3333334
## tCommonRatio1a=Yes 0.3616656 0.3429957 0.2953387
```

Cleanup from 1st Task

```
## Cleanup and Observable no longer needed, so absorb it:
DeleteNetwork(EMnet) ## Delete EM
AbsorbNodes(obs)
```

```
## Creating node list.
## Absorbing 1 nodes.
## First node address a2bf5f50.
## Its name is isCorrect.
## Absorbed.
```

```
## Currently, there is a Netica bug with Absorb Nodes, we will 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("..","Nets",
                      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.History
```

```
##                          High    Medium       Low
## *Baseline*          0.3333333 0.3333333 0.3333334
## tCommonRatio1a=Yes 0.3616656 0.3429957 0.2953387
## tCommonRatio2a=No  0.2340327 0.3149439 0.4510235
```

```
## Cleanup:  Delete EM and Absorb observables
DeleteNetwork(EMnet) ## Delete EM
AbsorbNodes(obs)
```

```
## Creating node list.
## Absorbing 1 nodes.
## First node address a259f870.
## Its name is isCorrect.
## Absorbed.
```

Save and Restore

```
## Fred logs out
WriteNetworks(Fred.SM,"FredSM.dne")
DeleteNetwork(Fred.SM)
is.active(Fred.SM)
```

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

```
## No longer active in Netica space

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

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

Getting Serious

- ACED field test has 230 students attempt all 63 tasks.
- File miniACED-Geometric contains 30 task subset
  - There may be data registration issues here, don't publish using these data before checking with me for an update
- Each row is one student Record
- Lets score the first student
  - And build a score history

Setup for mini-ACED

```
miniACED.data <- read.csv("../Nets/miniACED-Geometric.csv",row.names=1)
head(miniACED.data)
```

```
##      Class Treatment Sequencing Feedback Total.Items Correct Incorrect
## S055     1         1          2        2          63       8        55
## S058     1         1          2        2          63      33        30
## S053     1         2          2        1          63      12        51
## S061     1         1          2        2          63      21        42
## S063     1         1          2        2          63      21        42
## S066     1         1          2        2          63      19        44
##      Remaining tCommonRatio1a tCommonRatio1b tCommonRatio3a tCommonRatio3b
## S055         0              1              1              2              1
## S058         0              2              2              2              1
## S053         0              1              1              1              1
## S061         0              1              1              1              1
## S063         0              1              2              2              1
## S066         0              1              2              1              1
##      tCommonRatio2a tCommonRatio2b tExamplesGeometric1a
## S055              1              1                    1
## S058              1              2                    2
## S053              1              1                    1
## S061              1              1                    1
## S063              1              1                    2
## S066              1              1                    2
##      tExamplesGeometric1b tExamplesGeometric3a tExamplesGeometric3b
## S055                    1                    1                    1
## S058                    1                    1                    1
## S053                    1                    1                    1
## S061                    1                    1                    1
## S063                    1                    1                    1
## S066                    1                    1                    1
##      tExamplesGeometric2a tExamplesGeometric2b tExtendGeometric1a
## S055                    1                    1                  1
## S058                    1                    1                  1
## S053                    1                    1                  1
## S061                    1                    1                  2
## S063                    1                    1                  1
## S066                    1                    1                  1
##      tExtendGeometric1b tExtendGeometric3a tExtendGeometric3b
## S055                  1                  2                  1
## S058                  2                  2                  1
## S053                  1                  1                  1
## S061                  2                  1                  1
## S063                  2                  2                  2
## S066                  2                  2                  2
##      tExtendGeometric2a tExtendGeometric2b tModelExtendTableGeometric1a
## S055                  1                  1                            1
## S058                  2                  1                            2
## S053                  1                  1                            1
## S061                  1                  2                            1
## S063                  2                  1                            1
## S066                  1                  1                            2
```

```
##      tModelExtendTableGeometric1b tModelExtendTableGeometric3a
## S055                            1                            1
## S058                            2                            2
## S053                            1                            2
## S061                            2                            2
## S063                            2                            1
## S066                            1                            2
##      tModelExtendTableGeometric3b tModelExtendTableGeometric2a
## S055                            1                            1
## S058                            2                            1
## S053                            1                            1
## S061                            1                            1
## S063                            2                            1
## S066                            2                            1
##      tModelExtendTableGeometric2b tTableExtendGeometric1a
## S055                            1                       1
## S058                            2                       1
## S053                            2                       1
## S061                            1                       1
## S063                            1                       2
## S066                            1                       2
##      tTableExtendGeometric1b tTableExtendGeometric3a
## S055                       1                       1
## S058                       1                       1
## S053                       1                       1
## S061                       1                       1
## S063                       1                       1
## S066                       1                       1
##      tTableExtendGeometric3b tTableExtendGeometric2a
## S055                       1                       1
## S058                       2                       1
## S053                       2                       1
## S061                       2                       1
## S063                       1                       1
## S066                       2                       2
##      tTableExtendGeometric2b
## S055                       2
## S058                       1
## S053                       1
## S061                       1
## S063                       1
## S066                       2
```

```
## Mark columns of table corresponding to tasks
first.task <- 9
last.task <- ncol(miniACED.data)
## Code key for numeric values
t.vals <- c("No","Yes")
```

Setup new Student

```
## Pick a student, we might normally iterate over this.
Student.row <- 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)
```

Processing Loop

```
## 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("..","Nets",
                        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]] #Decode integer
  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(tid,t.val,sep="=")

  ## Cleanup, Delete EM and Absob Observables
  DeleteNetwork(EMnet)
  AbsorbNodes(obs) # Still broken
}
```

```
## Creating node list.
## Absorbing 1 nodes.
## First node address a2a24df0.
## Its name is isCorrect.
## Absorbed.
## Creating node list.
## Absorbing 1 nodes.
## First node address a2c440e0.
## Its name is isCorrect.
## Absorbed.
## Creating node list.
## Absorbing 1 nodes.
## First node address a2a24df0.
## Its name is isCorrect.
```

```
## Absorbed.
## Creating node list.
## Absorbing 1 nodes.
## First node address a2c440e0.
## Its name is isCorrect.
## Absorbed.
## Creating node list.
## Absorbing 1 nodes.
## First node address a2a24df0.
## Its name is isCorrect.
## Absorbed.
## Creating node list.
## Absorbing 1 nodes.
## First node address a2c440e0.
## Its name is isCorrect.
## Absorbed.
## Creating node list.
## Absorbing 1 nodes.
## First node address a2a24df0.
## Its name is isCorrect.
## Absorbed.
## Creating node list.
## Absorbing 1 nodes.
## First node address a2c440e0.
## Its name is isCorrect.
## Absorbed.
## Creating node list.
## Absorbing 1 nodes.
## First node address a2a24df0.
## Its name is isCorrect.
## Absorbed.
## Creating node list.
## Absorbing 1 nodes.
## First node address a2c440e0.
## Its name is isCorrect.
## Absorbed.
## Creating node list.
## Absorbing 1 nodes.
## First node address a2a24df0.
## Its name is isCorrect.
## Absorbed.
## Creating node list.
## Absorbing 1 nodes.
## First node address a2c440e0.
## Its name is isCorrect.
## Absorbed.
## Creating node list.
## Absorbing 1 nodes.
## First node address a2a24df0.
## Its name is isCorrect.
## Absorbed.
## Creating node list.
## Absorbing 1 nodes.
## First node address a2c440e0.
```

```
## Its name is isCorrect.
## Absorbed.
## Creating node list.
## Absorbing 1 nodes.
## First node address a2a24df0.
## Its name is isCorrect.
## Absorbed.
## Creating node list.
## Absorbing 1 nodes.
## First node address a2c440e0.
## Its name is isCorrect.
## Absorbed.
## Creating node list.
## Absorbing 1 nodes.
## First node address a2a24df0.
## Its name is isCorrect.
## Absorbed.
## Creating node list.
## Absorbing 1 nodes.
## First node address a2c440e0.
## Its name is isCorrect.
## Absorbed.
## Creating node list.
## Absorbing 1 nodes.
## First node address a2c59f30.
## Its name is isCorrect.
## Absorbed.
## Creating node list.
## Absorbing 1 nodes.
## First node address a290ed40.
## Its name is isCorrect.
## Absorbed.
## Creating node list.
## Absorbing 1 nodes.
## First node address a2c59f30.
## Its name is isCorrect.
## Absorbed.
## Creating node list.
## Absorbing 1 nodes.
## First node address a290ed40.
## Its name is isCorrect.
## Absorbed.
## Creating node list.
## Absorbing 1 nodes.
## First node address a2c59f30.
## Its name is isCorrect.
## Absorbed.
## Creating node list.
## Absorbing 1 nodes.
## First node address a11b3380.
## Its name is isCorrect.
## Absorbed.
## Creating node list.
## Absorbing 1 nodes.
```

```
## First node address a2c440e0.
## Its name is isCorrect.
## Absorbed.
## Creating node list.
## Absorbing 1 nodes.
## First node address 9feb3310.
## Its name is isCorrect.
## Absorbed.
## Creating node list.
## Absorbing 1 nodes.
## First node address 9ed863b0.
## Its name is isCorrect.
## Absorbed.
## Creating node list.
## Absorbing 1 nodes.
## First node address a25fffc0.
## Its name is isCorrect.
## Absorbed.
## Creating node list.
## Absorbing 1 nodes.
## First node address a138c600.
## Its name is isCorrect.
## Absorbed.
## Creating node list.
## Absorbing 1 nodes.
## First node address 9feb3310.
## Its name is isCorrect.
## Absorbed.
```

Weight of Evidence

- Good (1985)
- H is a binary hypothesis, e.g., *Proficiency* > `medium`
- E is evidence for that hypothesis
- *Weight of Evidence* of E for H (WOE) is

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

Conditional weight of evidence

- Observe two different pieces of evidence, E_1 and E_2.
- *Conditional Weight of Evidence* is

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

- Nice Additive Properties

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

- Order Sensitive:
  - Earlier terms have higher WOE
  - Total WOE is always the same
- WOE Balance Sheet (Madigan, Mosurski & Almond, 1997)

Weight Of Evidence Balance Sheet

Weight of Evidences can be calculated by differencing the log values in the history.

```
Student.History
```

```
##                                        High      Medium       Low
## *Baseline*                        3.333333e-01 0.333333313 0.3333334
## tCommonRatio1a=No                 1.238987e-01 0.261908591 0.6141927
## tCommonRatio1b=No                 3.842749e-02 0.211766571 0.7498059
## tCommonRatio3a=Yes                1.205092e-01 0.270862073 0.6086287
## tCommonRatio3b=No                 9.013028e-02 0.259188741 0.6506810
## tCommonRatio2a=No                 4.954598e-02 0.233125374 0.7173287
## tCommonRatio2b=No                 3.490408e-02 0.216613680 0.7484822
## tExamplesGeometric1a=No           1.005295e-02 0.145315900 0.8446311
## tExamplesGeometric1b=No           5.908299e-03 0.127308875 0.8667828
## tExamplesGeometric3a=No           5.698630e-03 0.126277551 0.8680238
## tExamplesGeometric3b=No           5.543364e-03 0.125432283 0.8690243
## tExamplesGeometric2a=No           5.186467e-03 0.123196520 0.8716170
## tExamplesGeometric2b=No           4.986964e-03 0.121875346 0.8731377
## tExtendGeometric1a=No             6.572015e-04 0.049339782 0.9500030
## tExtendGeometric1b=No             1.208575e-04 0.032974172 0.9669050
## tExtendGeometric3a=Yes            6.723817e-04 0.054701813 0.9446258
## tExtendGeometric3b=No             4.459546e-04 0.049828812 0.9497252
## tExtendGeometric2a=No             1.896276e-04 0.040176421 0.9596339
## tExtendGeometric2b=No             1.098957e-04 0.034829482 0.9650606
## tModelExtendTableGeometric1a=No 7.804838e-06 0.015482896 0.9845093
## tModelExtendTableGeometric1b=No 1.187362e-06 0.009638987 0.9903598
## tModelExtendTableGeometric3a=No 9.925629e-07 0.009404771 0.9905942
## tModelExtendTableGeometric3b=No 8.516319e-07 0.009189405 0.9908097
## tModelExtendTableGeometric2a=No 5.280224e-07 0.008351044 0.9916484
## tModelExtendTableGeometric2b=No 3.584744e-07 0.007693673 0.9923060
## tTableExtendGeometric1a=No      3.248774e-07 0.006830624 0.9931691
## tTableExtendGeometric1b=No      3.076568e-07 0.006449320 0.9935504
## tTableExtendGeometric3a=No      3.069094e-07 0.006432992 0.9935667
## tTableExtendGeometric3b=No      3.061927e-07 0.006417527 0.9935822
## tTableExtendGeometric2a=No      3.030247e-07 0.006352164 0.9936475
## tTableExtendGeometric2b=Yes     3.464628e-07 0.007213552 0.9927861
```

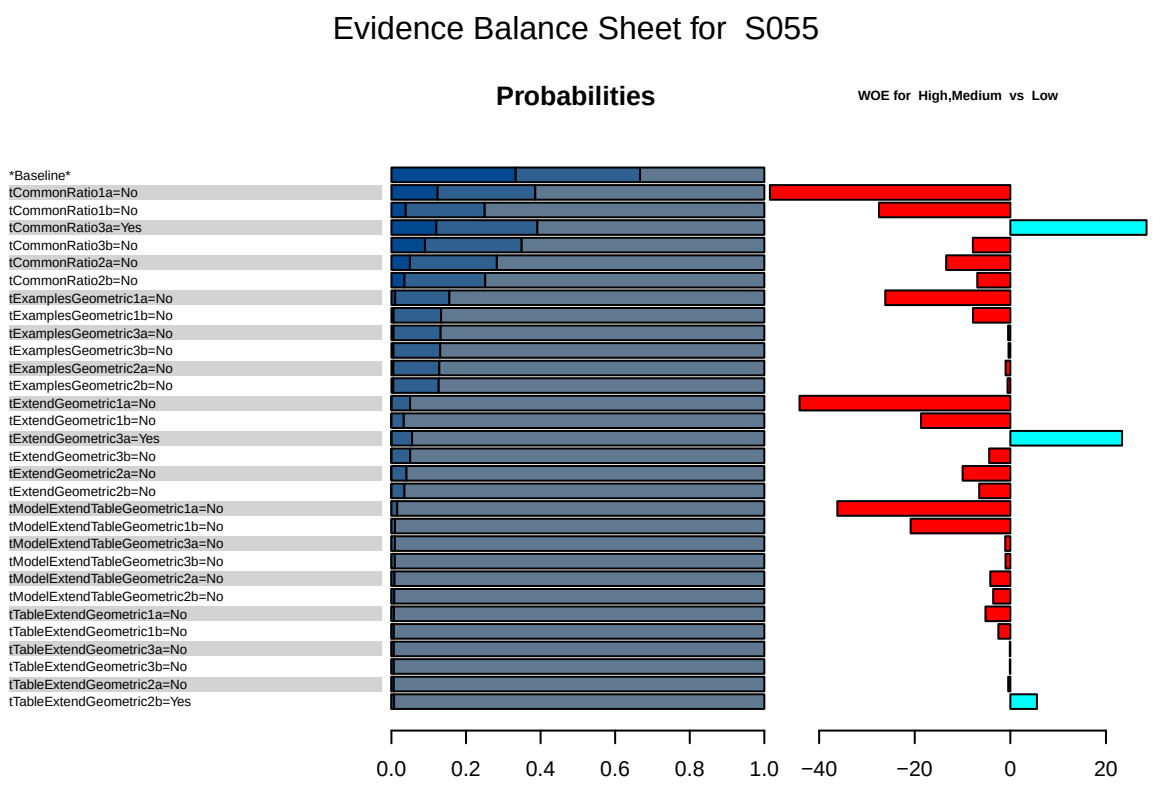
```
woeHist(Student.History,c("High","Medium"),"Low")
```

```
##               tCommonRatio1a=No                tCommonRatio1b=No
##                      -50.2964166                      -27.4737650
##              tCommonRatio3a=Yes                tCommonRatio3b=No
##                       28.4908326                       -7.8382306
##               tCommonRatio2a=No                tCommonRatio2b=No
##                      -13.4290529                       -6.9176446
##         tExamplesGeometric1a=No          tExamplesGeometric1b=No
##                      -26.1690290                       -7.8046967
##         tExamplesGeometric3a=No          tExamplesGeometric3b=No
```

```
##                          -0.4685993                          -0.3805303
##          tExamplesGeometric2a=No          tExamplesGeometric2b=No
##                          -0.9976840                          -0.5931907
##            tExtendGeometric1a=No            tExtendGeometric1b=No
##                         -44.1031050                         -18.6839843
##           tExtendGeometric3a=Yes            tExtendGeometric3b=No
##                          23.3668646                          -4.4295518
##            tExtendGeometric2a=No            tExtendGeometric2b=No
##                          -9.9841460                          -6.5150052
## tModelExtendTableGeometric1a=No tModelExtendTableGeometric1b=No
##                         -36.1909340                         -20.8559402
## tModelExtendTableGeometric3a=No tModelExtendTableGeometric3b=No
##                          -1.0793572                          -1.0160879
## tModelExtendTableGeometric2a=No tModelExtendTableGeometric2b=No
##                          -4.1926906                          -3.5902171
##       tTableExtendGeometric1a=No       tTableExtendGeometric1b=No
##                          -5.2050450                          -2.5113118
##       tTableExtendGeometric3a=No       tTableExtendGeometric3b=No
##                          -0.1108048                          -0.1052084
##       tTableExtendGeometric2a=No      tTableExtendGeometric2b=Yes
##                          -0.4474560                           5.5604248
```

We can graph this as a balansh sheet.

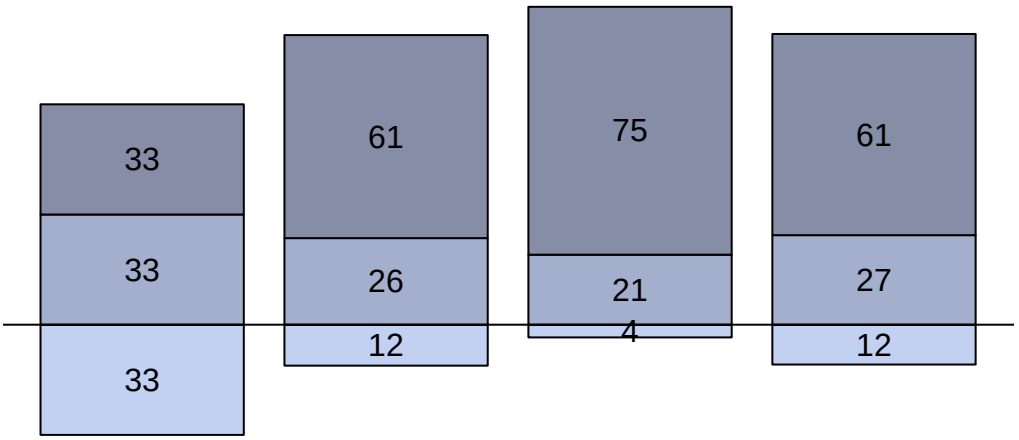
```
woeBal(Student.History,c("High","Medium"),"Low",
       title=paste("Evidence Balance Sheet for ",
       rownames(miniACED.data)[Student.row]))
```



Stacked Bar Charts

These are like ordinary stacked bar charts, but offset at a given level.

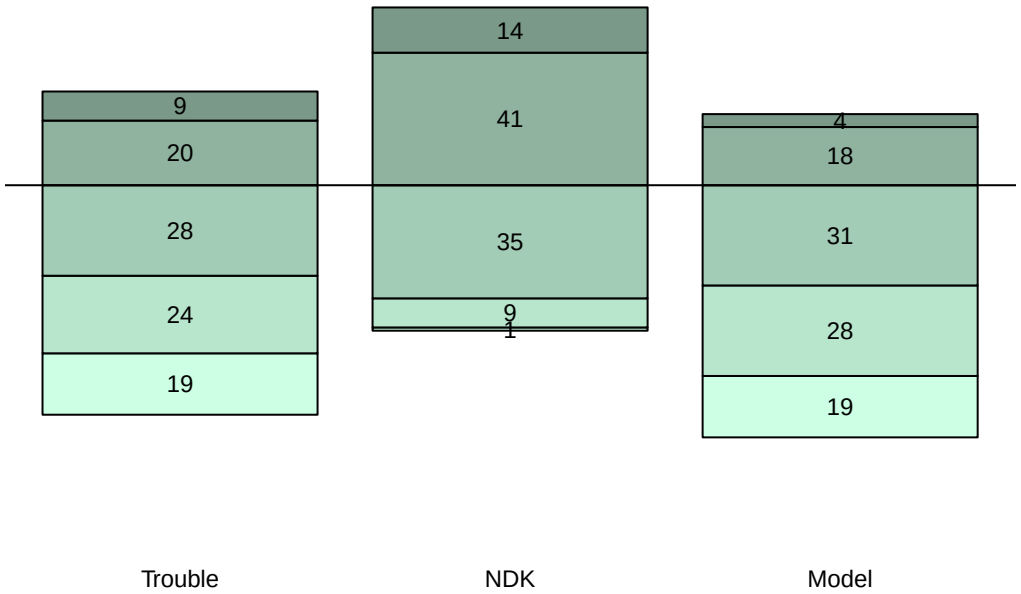
```
stackedBars(t(Student.History[1:4,]),1,
            col=hsv(223/360,.2,0.15*(3:1)+.5) )
```



Baseline tCommonRatio1b=No

```
margins <- data.frame (
 Trouble=c(Novice=.19,Semester1=.24,Semester2=.28,Semseter3=.20,Semester4=.09),
 NDK=c(Novice=.01,Semester1=.09,Semester2=.35,Semseter3=.41,Semester4=.14),
 Model=c(Novice=.19,Semester1=.28,Semester2=.31,Semseter3=.18,Semester4=.04)
)

stackedBars(margins,3,
            main="Marginal Distributions for NetPASS skills",
            sub="Baseline at 3rd Semester level.",
            cex.names=.75, col=hsv(148/360,.2,0.10*(5:1)+.5))
```



Comparing multiple students

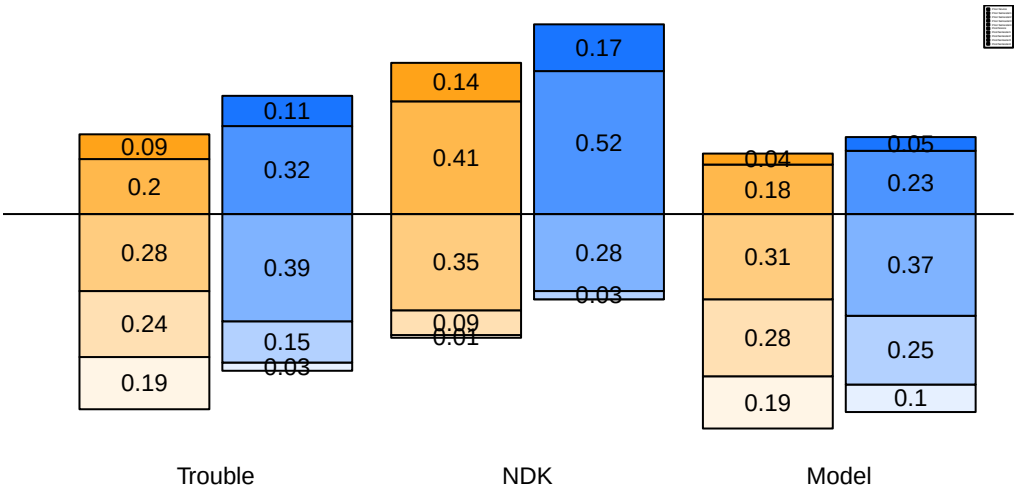
Compare one student to class average. Subgroup to full group Two groups to each other. Before to after.

```
margins.prior <- data.frame (
 Trouble=c(Novice=.19,Semester1=.24,Semester2=.28,Semseter3=.20,Semester4=.09),
 NDK=c(Novice=.01,Semester1=.09,Semester2=.35,Semseter3=.41,Semester4=.14),
 Model=c(Novice=.19,Semester1=.28,Semester2=.31,Semseter3=.18,Semester4=.04)
)

margins.post <- data.frame(
 Trouble=c(Novice=.03,Semester1=.15,Semester2=.39,Semseter3=.32,Semester4=.11),
 NDK=c(Novice=.00,Semester1=.03,Semester2=.28,Semseter3=.52,Semester4=.17),
 Model=c(Novice=.10,Semester1=.25,Semester2=.37,Semseter3=.23,Semester4=.05))

compareBars(margins.prior,margins.post,3,c("Prior","Post"),
            main="Margins before/after Medium Trouble Shooting Task",
            sub="Observables:  cfgCor=Medium, logCor=High, logEff=Medium",
            legend.loc = "topright",legend.cex=.1,
            cex.names=.75, col1=hsv(h=.1,s=.2*1:5-.1,alpha=1),
            col2=hsv(h=.6,s=.2*1:5-.1,alpha=1))
```

Margins before/after Medium Trouble Shooting Task



Observables: cfgCor=Medium, logCor=High, logEff=Medium

```
stopSession(sess)
```