RNetica
Quick Start Guide
Scoring A Student
Copyright 2017-8 Russell G. Almond

Session IIIa -- RNetica Quick Start

1

Downloading

• http://pluto.coe.fsu.edu/RNetica/
• Four Packages:
  – RNetica – R to Netica link
  – CPTtools – Design patterns for CPTs
  – Peanut/PNetica – Object-Oriented Parameterized Network
• Source & binary version (Win 64, Mac OS X)
  – Binary versions include Netica.dll/libNetica.so
    • In RStudio select “Package Archive” rather than CRAN
  – Source version need to download from http://www.norsys.com/ first
    • See INSTALLATION

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
  – Paid version (see http://www.norsys.com/ for price information)
  – Need to purchase API not GUI version of Netica
  • May want both (use GUI to visualize networks build in RNetica)
• CPTtools – Artistic (Free and open source), does not depend on Netica

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:
  > DefaultNeticaSession <- NeticaSession(LicenseKey = ”+Course/FloridaSU/ Ex15-05-30,120,310/XXXXX”)
  > q(”yes”)
• Restart R and type
  > library(RNetica)
  > startSession(DefaultNeticaSession)
• If license key is not installed, then you will get the limited/student mode. Most of these examples will run

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.
• Netica heap must be reconnected manually.

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() test to see if the node/net/session is active
Mini-ACED Proficiency model

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

![](image)

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 *adjointed* with the network.

Task to EM map

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

<table>
<thead>
<tr>
<th>Task ID</th>
<th>EM Filename</th>
<th>X</th>
<th>Y</th>
</tr>
</thead>
<tbody>
<tr>
<td>tCommonRatio1b</td>
<td>CommonRatioEasyEM</td>
<td>108</td>
<td>414</td>
</tr>
<tr>
<td>tCommonRatio2a</td>
<td>CommonRatioMedEM</td>
<td>108</td>
<td>534</td>
</tr>
<tr>
<td>tCommonRatio2b</td>
<td>CommonRatioMedEM</td>
<td>108</td>
<td>654</td>
</tr>
<tr>
<td>tCommonRatio3a</td>
<td>CommonRatioHardEM</td>
<td>108</td>
<td>774</td>
</tr>
<tr>
<td>tCommonRatio3b</td>
<td>CommonRatioHardEM</td>
<td>108</td>
<td>894</td>
</tr>
<tr>
<td>tExamplesGeometric1a</td>
<td>ExamplesEasyEM</td>
<td>342</td>
<td>294</td>
</tr>
<tr>
<td>tExamplesGeometric1b</td>
<td>ExamplesEasyEM</td>
<td>342</td>
<td>414</td>
</tr>
</tbody>
</table>

Scoring Script

• Follow along using the script found in ScoringScript.R in the miniACED folder.
• Don’t forget to `setwd()` to the miniACED folder (as it needs to find its networks).
• Don’t forget to start the Netica session using the license key (if you have one).

Loading and starting the Session

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

# start the session
sess <- NeticaSession(key)
startSession(sess)
```

Reloading Nets and Nodes

```r
# Read in network - Do this every time R is restarted
profModel <- ReadNetworks("miniACEDPnet.dne")

# If profModels already exists could also use

# Reconnect nodes - Do this every time R is restarted
allNodes <- NetworkAllNodes(profModel)
sgp <- allNodes
profNodes <- NetworkNodesInSet(profModel, "Proficiencies")
```
Aside 1: 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)

```r
# Node Sets
NetworkNodeSets(profModel)
NetworkNodesInSet(profModel, "pnodes")
NodeSets(sgp)
```

These are all settable

```r
NodeSets(sgp) <- c(NodeSets(sgp), "HighLevel")
```

## Node Sets

### Querying Nodes

- `NodeStates(sgp)` #List states
- `NodeParents(sgp)` #List parents
- `NodeLevels(sgp)` #List numeric values associated with states
- `NodeProbs(sgp)` #Conditional Probability Table (as array)
- `sgp[]` #Conditional Probability Table (as data frame)

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

### Inference

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

### Example

- `NodeValue(profNodes$CommonRatio) <- "Medium"`
  - Enter Evidence
- `NodeBeliefs(sgp)` #Current probability (given entered evidence)
- `NodeExpectedValue(sgp)` #If node has values, EAP

### Retract Evidence

```r
RetractNodeFinding(profNodes$ExamplesGeometric)
RetractNetFindings(profModel)
```

Many more examples

```r
help(RNetica)
```

Aside 2: RNetica Functions

- **Querying Nodes**
  - `NodeStates(sgp)` #List states
  - `NodeParents(sgp)` #List parents
  - `NodeLevels(sgp)` #List numeric values associated with states
  - `NodeProbs(sgp)` #Conditional Probability Table (as array)
  - `sgp[]` #Conditional Probability Table (as data frame)

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

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

Aside 2: Example

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

Query beliefs

- `NodeBeliefs(sgp)` #Current probability (given entered evidence)
- `NodeExpectedValue(sgp)` #If node has values, EAP

Retract Evidence

- `RetractNodeFinding(profNodes$ExamplesGeometric)`
- `RetractNetFindings(profModel)`

Many more examples

```r
help(RNetica)
```

Back to work

- Load the evidence model table
- Row names are task IDs
- EM column contains evidence model name
- EM filename has suffix ".dne" attached.

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

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

```r
## Copy the master proficiency model
Fred.SM <- CopyNetworks(profModel,"Fred")

## to make student model
Fred.SMvars <- NetworkAllNodes(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)

## Copy the master proficiency model
Fred.SM <- CopyNetworks(profModel,"Fred")

## to make student model
Fred.SMvars <- NetworkAllNodes(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)
```

Score 1st Task

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

### Adjoin SM and EM
EMnet <- ReadNetworks(paste(EMtable[t.name,"EM"],"dne",sep=".
")
ob <- AdJoinNetwork(Fred.SM,EMnet)
NetworkAllNodes(Fred.SM)

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

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

Stats and Cleanup for 1st task

```r
## Calculate statistics of interest
post <- NodeBeliefs(Fred.SMvars$SolveGeometricProblems)
Fred.History <- rhind(Fred.History,new=post)
rownames(Fred.History) <- paste(t.name,t.isCorrect,sep="=")

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

## AbsorbNodes(obs)

## Currently, there is a Netica bug with Absorb
Nodes, we will leave
## this node in place as that is mostly harmless.
```

2nd Task

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

### Adjoin SM and EM
EMnet <- ReadNetworks(paste(EMtable[t.name,"EM"],"dne",sep=".
")
ob <- AdJoinNetwork(Fred.SM,EMnet)
NetworkAllNodes(Fred.SM)

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

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

Save and Restore

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

## No longer active in Netica space

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

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

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

```r
## 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

```r
## 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")
CompileNetwork(Student.SM)
## Initialize history list
prior <- NodeBeliefs(Student.SMvars$SolveGeometricProblems)
rownames(Student.History) <- "Baseline"
colnames(Student.History) <- names(prior)
```

Loop Part 1: Add Evidence

```r
## 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(paste (EMtable[tid,"EM"],"dne",sep="."))
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
```

Loop Part 2: Capture Statistics

```r
## Update the history
post <- NodeBeliefs(Student.SMvars$SolveGeometricProblems)
Student.History <- rbind(Student.History,new=post)
rownames(Student.History) <- paste(tid,t.val,sep="=")
## Cleanup, Delete EM and Absorb Observables
DeleteNetwork(EMnet)
## AbsorbNodes(obs) # Still broken
```

Weight of Evidence

- Good (1985)
- $H$ is binary hypothesis, e.g., $\text{Proficiency} > \text{Medium}$
- $E$ is evidence for hypothesis
- Weight of Evidence (WOE) is

\[
W(H : E) = \log \frac{Pr(E|H)}{Pr(H|E)} = \log \frac{Pr(H|E)}{Pr(H)} - \log \frac{Pr(E)}{Pr(H)}
\]

Conditional Weight of Evidence

- Can define Conditional Weight of Evidence

\[
W(H : E|E_1) = \log \frac{Pr(E_2|H,E_1)}{Pr(E_1|H,E_1)}
\]

- Nice Additive properties

\[
W(H : E, E_2) = W(H : E_1) + W(H : E|E_1)
\]

- Order sensitive

- WOE Balance Sheet (Madigan, Mosurski & Almond, 1997)
Evidence Balance Sheet

Weight of Evidence Balance Sheet

## Now examine scoring history
head(Student.History)

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

## More ways to display scores
help(CPTtools)
```