Cognitive Modeling - Assignment 4: Yazayeri

Steven Bosch (s1861948)

December 14, 2015

1 Code

The following code gives my implementation for the readySetGo experiment. I used two more files, time.R and DM-module.R, which can be found in the appendix.

```
source(”DM–module.R”, TRUE)
source(”time.R”, TRUE)

# Experiment setup parameters
nrSubjects = 6    # The 'number of subjects'
trainingTrials = 500 # The number of training trials
testingTrials = 1000 # The number of testing trials

# Experiment durations, for calculating the activation of the chunks
FixPointDuration = 1000
delayLowerBound = 250
delayUpperBound = 850

# Create the prior distributions
priorDist1 = seq(from = 494, to = 847, by = (847-494)/10)
priorDist2 = seq(from = 671, to = 1023, by = (1023-671)/10)
priorDist3 = seq(from = 847, to = 1200, by = (1200-847)/10)
priorDists = list(priorDist1, priorDist2, priorDist3)

# The readySetGo experiment
readySetGo = function(nrSubjects, testingTrials, trainingTrials) {
  subjectsData = data.frame(Sub = integer(nrSubjects*length(priorDists)*testingTrials),
                 Cond = integer(nrSubjects*length(priorDists)*testingTrials),
                 Ts=double(nrSubjects*length(priorDists)*testingTrials),
                 Tp=double(nrSubjects*length(priorDists)*testingTrials))

  for(subject in 1:nrSubjects) {
    # For every condition (short, medium, long)
    for(condition in 1:length(priorDists)) {
      # Create the declarative memory of the current subject for the current condition
    num.chunks = 30 # The max interval is 1200 ms, which amounts to 25 or 26 ticks, so 30 should be enough
    max.num.encounters = testingTrials + trainingTrials
    DM = create.dm(num.chunks,max.num.encounters)

    # Temporary storage stuff
    activation = array(NA, dim=c(num.chunks))
    priors = array(NA, dim=c(num.chunks))

    # The current time
    curtime = 0

    # Training stage
    for(trial in 1:trainingTrials) {
      # The sample time
      t_s = priorDists[[condition]][sample(1:11, 1)]

      # Measure the sample time in an internal representation
      t_m = timeToTicks(t_s)

      # Determine the current time with the values used in the paper for delay time and fixation point duration
      delay = sample(250:850, 1)
      curtime = curtime + delay + FixPointDuration + t_s # The current time, given that the first trial was at time == 0 ms

      # Add the time the trial takes place as an encounter at the t_m ticks index
      DM = add.encounter(DM, t_m, curtime)

      # Increase the current time by the time of the to be measured sample
      curtime = curtime + t_s
    }

    # Testing stage
    for(trial in 1:testingTrials) {
      # The current sample time
      t_s = priorDists[[condition]][sample(1:11, 1)]

      # Measure the sample time in an internal representation
      t_m = timeToTicks(t_s)

      # Determine the current time with the values used in the paper for delay time and fixation point duration
      delay = sample(250:850, 1)
      curtime = curtime + delay + FixPointDuration + t_s # The current time, given that the first trial was at time == 0 ms

      # Add the time the trial takes place as an encounter at the t_mth index
      DM = add.encounter(DM, t_m, curtime)

      # Calculate activation values for all chunks that have encounters using the current time and the times the encounters took place
      for(chunk in 1:num.chunks) {
          if(!is.na(DM[chunk][1])) {
            activation[chunk] = actr.B(DM[chunk], curtime)
          }
        }

        # Calculate the priors with the activations
        activationSum = sum(exp(activation/curtime), na.rm = TRUE)

        for(chunk in 1:num.chunks) {
          priors[chunk] = 0.4 * exp(activation[chunk]/curtime) / activationSum
        }
        # Increase the prior of the current encounter, which is still fresh in memory
        priors[t_m] = priors[t_m] + 0.6

        # Determine the estimated time by multiplying the priors of all encounters with their measured duration
        # and store the produced duration using ticksToTime
        t_p = ticksToTime(round(sum(priors*c(1:num.chunks), na.rm = TRUE)))

        # Increase the current time by the time of the to be measured sample
        curtime = curtime + t_s

        # Store everything
        subjectsData$Sub[trial+(subject-1)*testingTrials*length(priorDists)+(condition-1)*testingTrials] = subject
        subjectsData$Cond[trial+(subject-1)*testingTrials*length(priorDists)+(condition-1)*testingTrials] = condition
        subjectsData$Ts[trial+(subject-1)*testingTrials*length(priorDists)+(condition-1)*testingTrials] = t_s
        subjectsData$Tp[trial+(subject-1)*testingTrials*length(priorDists)+(condition-1)*testingTrials] = t_p
      }
    }
  }
  subjectsData
}

data = readySetGo(nrSubjects, testingTrials, trainingTrials)

## Plot the data

brown <- "#8b4513";
red <- "#ff1100";
black <- "#000000";
brownT <- "#8b451322";
redT <- "#ff110022";
blackT <- "#00000022";

## ---

par(mfrow=c(1,1))

plotDat <- with(data,aggregate(list(Tp=Tp),list(Ts=Ts,Cond=Cond),
mean))

yrange <- range(plotDat$Ts)*c(.95,1.05)

with(plotDat[plotDat$Cond==3,],plot(Ts,Tp,type=”b”,col=red,lwd=2,ylim=yrange,xlim=yrange,main=”Model data”))
with(plotDat[plotDat$Cond==2,],lines(Ts,Tp,type=”b”,col=brown,lwd=2,ylim=yrange,xlim=yrange))
with(plotDat[plotDat$Cond==1,],lines(Ts,Tp,type=”b”,col=black,lwd=2,ylim=yrange,xlim=yrange))

lines(c(yrange[1],yrange[2]),c(yrange[1],yrange[2]),col=”darkgrey”,lty=2)

with(data[data$Cond==3,],points(jitter(Ts),Tp,col=redT,pch=”.”,cex=3))
with(data[data$Cond==2,],points(jitter(Ts),Tp,col=brownT,pch=”.”,cex=3))
with(data[data$Cond==1,],points(jitter(Ts),Tp,col=blackT,pch=”.”,cex=3))
```

2 Plots

This code yields the graph given in figure 1, modelling 6 subjects doing 1000 trials per condition, after having done 500 training trials.

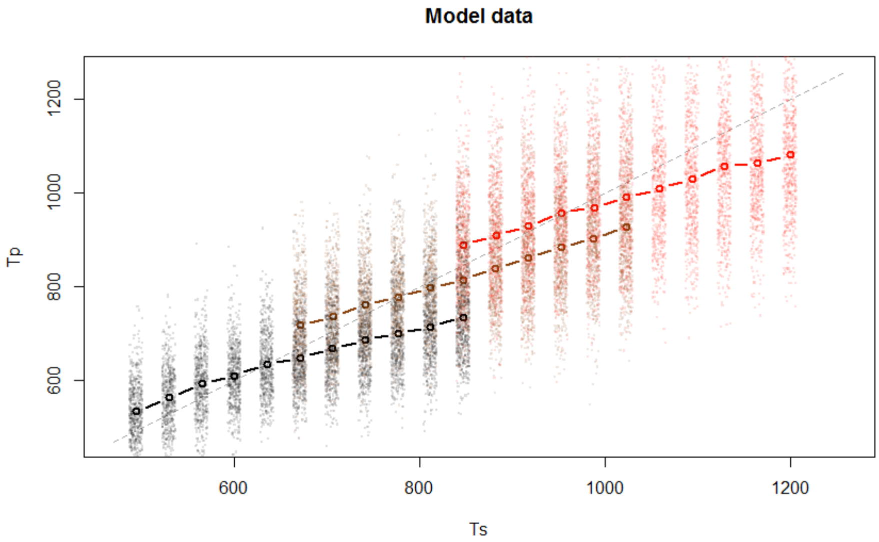

Figure 1: Interval estimation in the readySetGo experiment, using a cognitive model that ran for 6 subjects doing 1000 trials per condition after 500 training trials.

Appendix A Code

Listing 1: time

```
# Time measurement parameters
t_0 = 11             # The starting length of a pulse in ms
a = 1.1              # The growth factor of the pulses
b = 0.015            # Noise influence parameter

# Act-R's noise function
actr.noise <- function(s,n=1) {
  rand <- runif(n,min=0.0001,max=0.9999)
  s * log((1 - rand ) / rand)
}

# Convert ticks to time and return time (not used in the bisection model but added for completeness sake)
ticksToTime = function(ticks) {
  # The starting values, the first pulse length is also subject to noise here
  pulseLength = t_0 + actr.noise(b*a*t_0)
  measuredTime = 0
  # Add the time from every tick and return the total measured time
  for(tick in 1:ticks) {
    pulseLength = a * pulseLength + actr.noise(b*a*pulseLength)
    measuredTime = measuredTime + pulseLength
  }
  measuredTime
}

# Convert time to ticks and return ticks
timeToTicks = function(time) {
  # The starting values, the first pulse length is also subject to noise here
  ticks = 0
  pulseLength = t_0 + actr.noise(b*a*t_0)
  measuredTime = 0
  # Until the targetTime is reached, continue to count ticks and return the final count
  while(measuredTime < time) {
    pulseLength = a * pulseLength + actr.noise(b*a*pulseLength)
    measuredTime = measuredTime + pulseLength
    ticks = ticks + 1
  }
  ticks - 1
}
```

Listing 2: DM-module

```
## List with parameter values:

params <- list()
params$d <- .5

## DM functions

create.dm <- function(chunks,encounters) {
  if (chunks > 52) {
    stop(”Only up to 52 chunks allowed.”)
  }
  DM <- array(NA,c(chunks,encounters))
  row.names(DM) <- c(letters,LETTERS)[1:chunks]
  DM
}

add.encounter <- function(DM,chunk,time) {
  tmp <- DM[chunk,]
  DM[chunk,sum(!is.na(tmp))+1] <- time
  DM
}

get.encounters <- function(DM,chunk) {
  tmp <- DM[chunk,]
  tmp[!is.na(tmp)]
}

## Baselevel activation function:

actr.B <- function(encounters,curtime) {
  if (length(curtime)>1) {
    sapply(curtime,function(X) { actr.B(encounters,X)})
  } else {
    if (curtime < min(encounters)) {
      return(NA)
    } else {
      log(sum((curtime - encounters[encounters<curtime])^-params$d)
    )
    }
  }
}
```