

```

# Retirement simplified model
# Rationals, Imitators, Randoms

# initialize left and right neighbor vars and i initialization -- same for all runs
LN=RN=i=r=0

#Number of runs
RUNS=5

#RunsData: 1-population size, 2-generations total, 3-optimal coordination value
#4-Randoms do not change dummy (1 if they change), 5-p1, 6-p2, 7-p3,
#8-time before equilibrium is reached, 9-end of the run proportion of choosing optimal coordination
RunsData=array(0,dim=c(RUNS,9))

# Starting cycle of runs
for (r in 1:RUNS) {

#dummy var when you stop checking for equilibrium
eq_dummy=0

#the number of a generation when the equilibrium is reached (define as 3 proportions in a row are
equal)
eq_done=0

# population drawn from 200-600 uniform
N=RunsData[r,1]=as.integer(runif(1,200,600))

# number of generations drawn from 20-80 uniform
G=RunsData[r,2]=as.integer(runif(1,20,80))

#payoff matrix
# C1 drawn from 1-10 uniform
C1=RunsData[r,3]=runif(1,1,10)
RET=t(array(c(C1,0,0,1),dim=c(2,2)))

#do randoms change? (1 if they do)
rc_change=RunsData[r,4]=rbinom(1,1,0.5)

# starting probabilities
p1=RunsData[r,5]=runif(1,0,0.3)
p3=RunsData[r,6]=runif(1,0,0.3)
p2=RunsData[r,7]=1-p1-p3

# DONE WITH GENERATION OF RANDOM PARAMETERS
# NOW THE REST OF THE PRE-RUN INITIALIZATION

#population array: fitness, type, behavior
pop=array(0,dim=c(N,3))

# starting fitness
pop[1:N,1]=0

# Lottery initialization
Lottery=array(0,dim=c(N,3))
# starting population given p1, p2, p3

```

```

Lottery=t(rmultinom(N, size = 1, prob=c(p1,p2,p3)))
pop[1:N,2]=max.col(Lottery)

# initial behavior is random
pop[1:N,3]=rbinom(N,1,0.5)+1
pop[pop[1:N,2]==1,3]=1

#data array to record proportions of types and actual behaviors
# percent choosing to Retire
data=array(0,dim=c(G,1))

# NOW START GENERATIONS

#cycle of generations
for (g in 1:G) {

#cycle of individuals in the population
for (x in 1:N) {

# left and right neighbors on a circle
LN=x-1; if (LN==0) LN=N
RN=x+1; if (RN==N+1) RN=1

#if randoms change each generations do this for all randoms
#if (rc_change==1 & pop[x,2]==3) pop[x,3]=rbinom(1,1,0.5)+1

#interaction
pop[x,1]=pop[x,1]+RET[pop[x,3],pop[LN,3]]
pop[LN,1]=pop[LN,1]+RET[pop[LN,3],pop[x,3]]
pop[x,1]=pop[x,1]+RET[pop[x,3],pop[RN,3]]
pop[RN,1]=pop[RN,1]+RET[pop[RN,3],pop[x,3]]
}

#local imitation for type 2 (only!)
for (i in 1:N) {
if (pop[i,2]==2) {
# left and right neighbors on a circle
LN=i-1
if (LN==0) LN=N
RN=i+1
if (RN==N+1) RN=1
if (pop[i,1]<pop[LN,1] | pop[i,1]<pop[RN,1]) {
if (pop[LN,1]>pop[RN,1]) pop[i,3]=pop[LN,3] else pop[i,3]=pop[RN,3]
}
}
}

#reset starting fitness(es)
pop[1:N,1]=0

#NO NEED TO SAVE DATA NOW
#save the data on behavior
data[g,1]=(sum(pop[1:N,3]==1))/N

```

```

if (g>3) {
if (eq_dummy==0) {
if (data[g,1]==data[g-1,1] & data[g,1]==data[g-2,1]) {
eq_dummy=1
eq_done=g
}
}
}

}

# NO PLOTS FOR MASSIVE RUNS
#plot(1:G,data,type="l")

# NEED TO SAVE OUR DEPENDENT VARIABLES FOR THE SIMULATION RUN
#saving the generation number when the population reached equilibrium (given our definition of it)
RunsData[r,8]=eq_done
#saving the proportion of choosing to retire (first row)
RunsData[r,9]=(sum(pop[1:N,3]==1))/N

}

#code to save data
#NOTE! If you use Vista, make sure to run R as ADMINISTRATOR!
#NOTE! The file will be saved in your R folder!
#write.csv(data, file = "filename.csv")

write.csv(RunsData, file="filename.csv")
RunsData
done=1

```