Game Theory of Societal Dynamics

Daniel Lawson

2025-11-17

Game Theory of Societal Dynamics

This is the code that goes with the paper Apparent Strength Conceals Instability in a Model for the Collapse of Historical States by Daniel Lawson and Neeraj Oak, published in PLOS ONE in 2014.

Introduction

The model uses a vector representation of resources, power and conformity for a number of `factions` (societal groups). Each timestep each faction decides whether to `conform` to the state, or `defect`. Resources and power are then updated based on these states.

There are many choices in the dynamics, as well as in the decision making. These include expected resource gain, inertia and random noise. These are stored in a list of parameters called `p` below.

Because the dynamics are inherently discrete, there is always a possibility that some parameters take variables outside of the permitted range. This makes the code more complex than would otherwise be needed.

Parameterisation

This function defines all of the parameter options that are implemented in this code.

Some of these are "disabled" by default, i.e. set to zero or one, but can be enabled to explore extensions to the basic model. The following should aid understanding of the basic parameters of the model. For extensions, refer to the code.

Timescales

The dynamics of decision making choices is on the timescale of iterations. It is therefore recommened to set the power dynamics rate `mu` to some small-enough constant and treat this as a "timestep" parameter for smooth dynamics outside of a defection cascade.

Power dynamics

Power is a zero sum game and adds up to 1, so is on the scale of $1/N$. All factions gain power proportional to their resource share, which is then renormalised.

There are two mechanisms to redistribute power, a multiplicative one `w` that describes the defectors relative gain for a specific resource, and an additive one `rho`. The additive choice is needed for the spatial model, so its our default (unlike in Model 1 from the paper which uses both parameters.)

Resource dynamics

Any resource distribution can be allocated but it is simplest to allocate it summing to 1 by default. This keeps it on the same scale as power. However, it is not a zero sum game.

The resource dynamics are based on the idea that conformers share resources proportional to their power, while defectors retain their own resources but pay a penalty determined by the parameter `pwar`, which should therefore be on the scale $1/N$. For example, in the default parameters `rpen=0.02` which and $N = 11$ so defectors receive around 20% less resource than the conformers on average.

```
default_parameters<-function(N){
  ## Default parameters for the model, as used for Figure 1 of the paper (Model 1 in S1)
  list(mu=0.01, # multiplicative power gain rate for conformers, and overall rate
       rho=0.2, # additive power gain during defection which should be O(1/N)
       rpen=0.02, # additive resource penalty for defection
## Only one of rho>0, w>1 is needed for defection to be potentially advantageous

## Extensions
       w=1, # multiplicative power gain rate for defectors, relative to mu
       sigma=0, # noise in the decision function
       epsilon=1, # multiplicative resource penalty for defection
       alphap=1, # power gain exponent for conformers
       alphaw=1, # power gain exponent for defectors
       kchoice=0.0, # bias towards conformity in the decision function
       inertia=0.0, # intertia in the decision function
       ## spatial parameters
   #    rho = 0.0, # scaling of political gain from war as a function of distance. Set to 0 to make all
       wardecay=0.0, # decay rate of war effects with distance. Set to 0 to disable the spatial model
       ## per-faction parameters
       res0=rep(1/N,N) # resource level available to faction i. Called R^0_i in the paper
       )
}
```

Power dynamics

Let P_i be the power of faction i, R_i its resources, and C_i its conformity state (1=conform, 0=defect).

In the basic model, power follows the dynamics:

$$\Delta P_i = \mu R_i \quad \text{(conform)}$$
$$= \mu w R_i + \mu \rho \quad \text{(defect)}$$

There are mechanisms for a spatial model, which decreases ρ as we move from the capital, or a non-linear model using `alphap` where power gain is non-linear in resources.

```
newpow<-function(rest,powt,conformt,p){
  ## Update power levels
  ## Initialisation: sanity checks
  powt[powt<0]<-0 ## No negative power
  powt[powt>0]<-powt[powt>0]/sum(powt[powt>0]) ## Normalize power

  ## Compute the power change
  deltapow<-rep(0,length(powt)) #define the result (content not used)
  deltapow[conformt==1]<-p$mu * (rest[conformt==1])^p$alphap
  deltapow[conformt==0]<-p$mu * p$w *
    ( (rest[conformt==0])^p$alphaw +  p$rho *
    ## Spatial model adjustment: power gain from war decreases with distance, scaled by rho
    ifelse(p$wardecay>0,
           getpwareff(powt,p)[conformt==0], # spatial model if used
           1)  # 1 otherwise
```

```
  )

  ## prevent negative power
  whichgoneg<-(powt + deltapow-mean(deltapow)<0)
  newpowt<-rep(0,length(powt))
  newpowt[!whichgoneg]<-powt[!whichgoneg] + deltapow[!whichgoneg]-mean(deltapow[!whichgoneg])
  return(newpowt)
}
```

Resource dynamics

In the basic model, if the total state (i.e. conformer) power is $P^* = \sum_{i:C_i=1} P_i$ and the total state resource is $R^* = \sum_{i:C_i=1} R_i$ then resources follow the dynamics:

$$\Delta R_i = R^* P_i / P^* \quad \text{(conform)}$$
$$= \epsilon P_i - r_{pen} \quad \text{(defect)}$$

i.e. Conformer resources is redistributed according to power, defectors receive a penalty, either multiplicative or additive.

Again the spatial model modifies the penalty for defection to decrease with distance from the capital, and a nonlinear model using `sumresc` is implemented.

```
newres<-function(rest,powt,conformt,p){
  ## Update resource levels

  ## Calculate the resource and power available to the conformers
  sumresc<-sum(p$res0[conformt==1])
  sumpowc<-sum(powt[conformt==1])

  ## Conformers share resources proportional to power
  ret<-rest
  if(sumpowc>0) { ## Regular case
    ret[conformt==1]<-(powt[conformt==1]/sumpowc) * sumresc
    ## Equal share if somehow the state has no power
  }else ret[conformt==1]<-(1.0/max(1,sum(conformt==1))) * sumresc

  ## Defectors retain their resource, but with a penalty
  ret[conformt==0]<-p$epsilon * p$res0[conformt==0] - p$rpen *
    ## Choice of resource penalty for defection.
    ## penalty for war decreases with distance (scaled by rpen)
    ifelse(p$wardecay>0,
          getpwareff(powt,p)[conformt==0], # penalty under spatial model
          1)# penalty under the no-spatial model
  ## Prevent negative resources
  ret[ret<0]<-0
  ret
}
```

Decision making

In the simple model, each faction simply compares the resource level it would receive if it conformed or defected, and chooses the option with the higher expected resource level. This is modified by a bias towards conformity `kchoice`, inertia `inertia` and random noise `sigma`.

3

```r
chooseconform<-function(rest,powt,conformt,p){
  ## Choose whether to conform or not
  conformnew<-conformt # assume other actors follow their previous decisions
  for(i in 1:length(rest)){ ## For every faction:
    conformtest<-conformt
    conformtest[i]<-1
    ## Evaluate reward if we conform
    peaceres<-newres(rest,powt,conformtest,p)[i]
    conformtest[i]<-0
    ## Evaluate reward if we defect
    warres<-newres(rest,powt,conformtest,p)[i]
    if(peaceres - warres + p$kchoice + p$inertia*(2*conformt[i]-1) + rnorm(1,sd=p$sigma)<0) { ## Decisi
      conformnew[i]<-0
    }else {conformnew[i]<-1}
  }
  return(conformnew)
}
```

Iterating the model

Running the dynamics of the model is very simple, just iterating the three steps above. A simulation is just repeating this for a number of timesteps anre recording the history.

```r
oneiteration<-function(curstate,p){
  # One iteration of the model: update conform choices, resource and power
  curstate$conformt<-chooseconform(curstate$rest,curstate$powt,curstate$conformt,p)
  curstate$rest<-newres(curstate$rest,curstate$powt,curstate$conformt,p)
  curstate$powt<-newpow(curstate$rest,curstate$powt,curstate$conformt,p)
  return(curstate)
}


dosim<-function(curstate,timesteps,p){
  ## Run the simulation for a number of timesteps
  ## First create a history object to store what happened
  thistory<-data.frame(t(unlist(curstate)))
  history<-as.data.frame(matrix(0,nrow=1+timesteps,ncol=1+dim(thistory)[2]))
  names(history)<-c("Time",names(thistory))
  history[1,]<-cbind(0,thistory)
  ## Iterate and store history
  for(i in 1:timesteps){
    curstate<-oneiteration(curstate,p)
    history[i+1,]<-cbind(i,data.frame(t(unlist(curstate))))
  }
  ## Return the history and the final state
  list(curstate=curstate,history=history)
}
```

Extensions

The spatial model requires two helper functions to compute distances from the capital, and adjust the war penalty accordingly.

```r
getdists<-function(capital,N) {
  ## Distances of each from from the capital. Used in the spatial model only
  dists<-matrix(0,nrow=N,ncol=3)
  dists[,1]<-abs((1:N)-capital)
  dists[,2]<-abs((1:N)-(N+capital))
  dists[,3]<-abs((1:N)-(-N+capital))
  apply(dists,1,min)
}

getpwareff<-function(powt,p){
  ## Get the war penalty adjusted for distance from capital
  capital<-which(powt==max(powt))
  if(length(capital)>1) capital<-sample(capital,1)
  tdists<-getdists(capital,length(powt))
  twdists<-exp(-tdists*p$wardecay)
  twdists/mean(twdists)
}
```

Helper Functions

Helper functions for some reasonable choices of initialising distributions.

```r
## Choices for initial resource distributions
## These can be used to initialise power, and create fixed resource distributions

initialise_brokenstick<-function(N){
  ## "Broken stick" mean distribution, i.e. power law
  (1/2)^(1:(N))/sum((1/2)^(1:(N)))
}

initialise_inverseN<-function(N){
  ## 1/N distribution
  res0<-1/(1:N)
  res0<-res0 / sum(res0)
  res0
}

initialise_normal<-function(N,y,x0,sd){
  ## Normal distribution of resources
  ## y = total resources
  ## x0 = centre
  ## sd = standard deviation
  yrange<- seq(-N+1,2*N,1)
  tmp<-dnorm(yrange,x0,sd)
  tmp<-tmp[1:N] + tmp[(2*N+1):(3*N)] + tmp[(N+1):(2*N)]
  y*tmp/sum(tmp)
}
```

Helper functions to summarise history in terms of means and s.d. of resources, power and conformity rate.

```r
summarizehistory<-function(historypoint){
  ## summarises a row of the history matrix
  historypoint<-as.vector(unlist(historypoint))
  numx<-(length(historypoint)-1)/3
  warlevel<-mean(historypoint[1+2*numx+1:numx])
```

```r
  sdres<-sd(historypoint[1+0*numx+1:numx])
  sdpow<-sd(historypoint[1+1*numx+1:numx])
  return(c(conformrate=warlevel,sdres=sdres,sdpow=sdpow))
}
```

Plotting functions

Some useful visualisations. Feel free to modify these as needed.

```r
plot_history<-function(history,skip=1,...){
  ## Plot the resource and power history
  par(mar=c(4,4,2,4))
  history[,-1] <- history[,-1]  + 0.000001
  numx<-(dim(history)[2]-1)/3
  myylim<-range(history[,2:(1+numx)])
  myylim2<-range(history[,(2+numx):(1+2*numx)])
  plot(history$Time,history$Time,ylim=myylim,type="n",xlab="Time",ylab="",...)

  for(faction in 1:numx){
    lines(history$Time,history[,1+faction],col="red",lty=faction)
  }
  for(faction in 1:numx){
    lines(history$Time,(history[,1+numx+faction]-myylim2[1])*(myylim[2]-myylim[1])/(myylim2[2]-myylim2[
  }
  axis(4,seq(myylim[1],myylim[2],by=(myylim[2]-myylim[1])/5),format(seq(myylim2[1],myylim2[2],by=(myyli
  mtext("red=resources",2,2)
  mtext("green=power",4,2)
}

plot_history_summary<-function(history,skip=1,...){
  ## Plot summary statistics from the history
  tdat<-t(apply(history,1,summarizehistory))
  plot(history$Time,history$Time,ylim=c(0,max(tdat[,-1])),type="n",xlab="Time",ylab="",...)
  lines(history$Time,tdat[,1]*max(tdat[,-1]),col="black")
  lines(history$Time,tdat[,2],col="red")
  lines(history$Time,tdat[,3],col="green")
  axis(4,seq(0,1,by=0.2)*max(tdat[,-1]),seq(0,1,by=0.2))
  mtext("red=sdres,green=sdpower",2,2)
  mtext("black=conform",4,2)
}

image_history<-function(history,skip=1,...){
  ## Visualisation where each actor is a row
  opar<-par(mar=par()$mar*c(1,1.8,1,1.8))
  conformnames<-grep("conformt",names(history))
  powernames<-grep("powt",names(history))
  maxname<-apply(history[,powernames],1,function(x){which(x==max(x))})
  for(i in 1:dim(history)[1]) history[i,conformnames[maxname[i]]]<-2
  image(as.matrix(history[,conformnames]),axes=FALSE,xlab="Time",ylab="Space",col=c("red","white","blac
  axis(1,seq(0,1,length.out=5),seq(min(history$Time)-1,max(history$Time),length.out=5))
  axis(2,seq(0,1,length.out=length(conformnames)),seq(1,length(conformnames),length.out=length(conformna
}

visualise_history<-function(history,skip=1,...){
```

```
  par(mfrow=c(3,1))
  plot_history(history,skip,...)
  plot_history_summary(history,skip,...)
  image_history(history,skip,...)
}
```

Example usage

The first example is for the default parameters with no spatial effects.
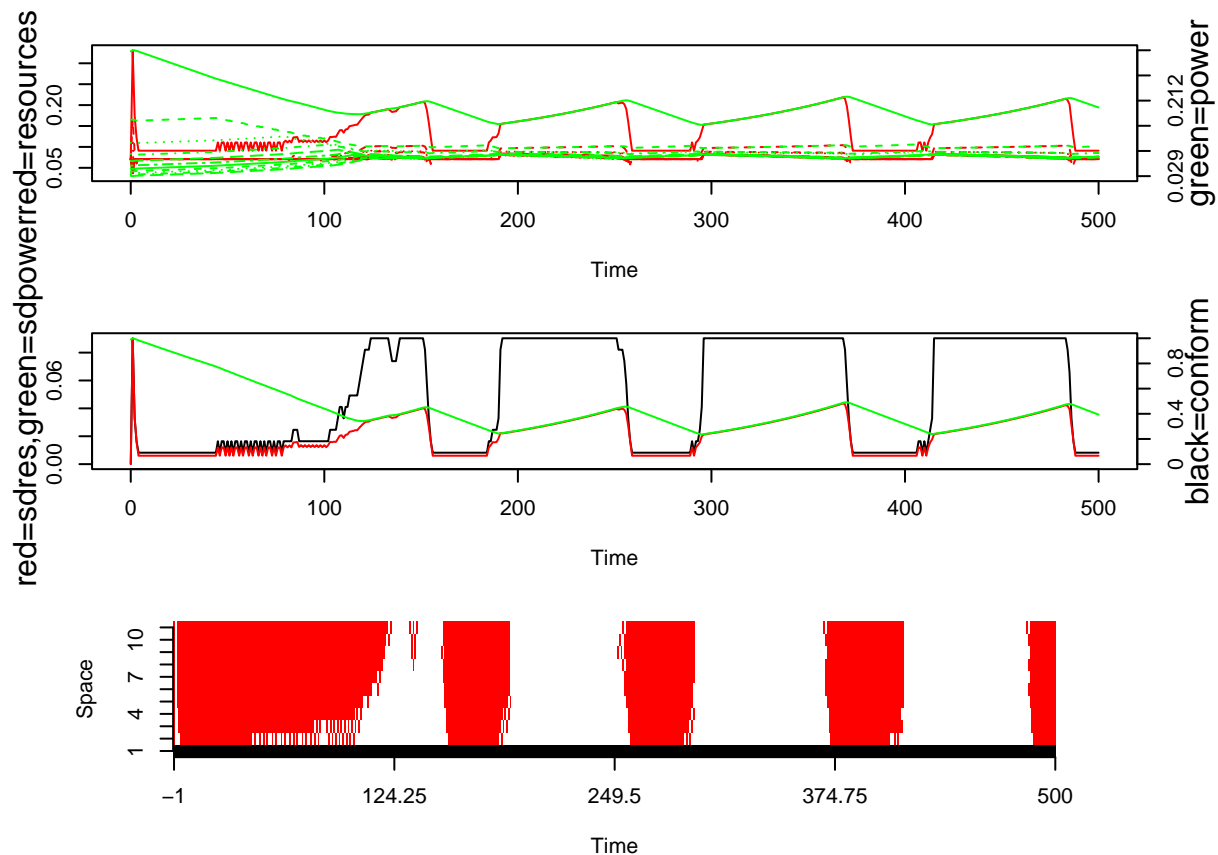
```
N<-11
p<-default_parameters(N)
## Options for defining the penalty for defecting, swapping between multiplicative and additive with no
#p$w=1; p$rho=0.2;
#p$rho=0; p$w=4

## Initial state
conformI<-rep(0,N) # No initial conformers
powI<-initialise_inverseN(N) # Initial power distribution
resI<-newres(p$res0,powI,conformI,p) # Compute resource allocation based on is
curstate<-list(rest=resI,powt=powI,conformt=conformI)

## Run the simulation
res_simple<-dosim(curstate,500,p)

#png(file="spatialModelNoSpatialEffects.png",height=1024,width=1280)
visualise_history(res_simple$history)
```

```
#dev.off()
```

An example of the spatial model will be put here.

```
ps<-default_parameters(N)
curstate<-list(rest=resI,powt=powI,conformt=conformI)
ps$wardecay=0.15
## incomplete
```