

CALIBRATION OF ZULF'S MARKOV MORAL THEORY FOR HUMAN RACE

ZULFIKAR MOINUDDIN AHMED

Our model is a 55 parameter model for Human Moral Values. The lower left triangular values of a 10 by 10 transition Probability Matrix are the parameters to be fit.

Here we provide *working* code.

```
# load data
wvs7<-readRDS("wvs7.rds")
vars<-c("Q290", "Q46", "Q57", "Q177", "Q178", "Q179", "Q180", "Q181", "Q182", "Q183", "Q184", "Q185", "Q186", "Q187", "Q188", "Q189", "Q190", "Q191", "Q192", "Q193", "Q194", "Q195", "Q196", "Q197", "Q198", "Q199", "Q200", "Q201", "Q202", "Q203", "Q204", "Q205", "Q206", "Q207", "Q208", "Q209", "Q210", "Q211", "Q212", "Q213", "Q214", "Q215", "Q216", "Q217", "Q218", "Q219", "Q220", "Q221", "Q222", "Q223", "Q224", "Q225", "Q226", "Q227", "Q228", "Q229", "Q230", "Q231", "Q232", "Q233", "Q234", "Q235", "Q236", "Q237", "Q238", "Q239", "Q240", "Q241", "Q242", "Q243", "Q244", "Q245", "Q246", "Q247", "Q248", "Q249", "Q250", "Q251", "Q252", "Q253", "Q254", "Q255", "Q256", "Q257", "Q258", "Q259", "Q260", "Q261", "Q262", "Q263", "Q264", "Q265", "Q266", "Q267", "Q268", "Q269", "Q270", "Q271", "Q272", "Q273", "Q274", "Q275", "Q276", "Q277", "Q278", "Q279", "Q280", "Q281", "Q282", "Q283", "Q284", "Q285", "Q286", "Q287", "Q288", "Q289", "Q290", "Q291", "Q292", "Q293", "Q294", "Q295", "Q296", "Q297", "Q298", "Q299", "Q300", "Q301", "Q302", "Q303", "Q304", "Q305", "Q306", "Q307", "Q308", "Q309", "Q310", "Q311", "Q312", "Q313", "Q314", "Q315", "Q316", "Q317", "Q318", "Q319", "Q320", "Q321", "Q322", "Q323", "Q324", "Q325", "Q326", "Q327", "Q328", "Q329", "Q330", "Q331", "Q332", "Q333", "Q334", "Q335", "Q336", "Q337", "Q338", "Q339", "Q340", "Q341", "Q342", "Q343", "Q344", "Q345", "Q346", "Q347", "Q348", "Q349", "Q350", "Q351", "Q352", "Q353", "Q354", "Q355", "Q356", "Q357", "Q358", "Q359", "Q360", "Q361", "Q362", "Q363", "Q364", "Q365", "Q366", "Q367", "Q368", "Q369", "Q370", "Q371", "Q372", "Q373", "Q374", "Q375", "Q376", "Q377", "Q378", "Q379", "Q380", "Q381", "Q382", "Q383", "Q384", "Q385", "Q386", "Q387", "Q388", "Q389", "Q390", "Q391", "Q392", "Q393", "Q394", "Q395", "Q396", "Q397", "Q398", "Q399", "Q400", "Q401", "Q402", "Q403", "Q404", "Q405", "Q406", "Q407", "Q408", "Q409", "Q410", "Q411", "Q412", "Q413", "Q414", "Q415", "Q416", "Q417", "Q418", "Q419", "Q420", "Q421", "Q422", "Q423", "Q424", "Q425", "Q426", "Q427", "Q428", "Q429", "Q430", "Q431", "Q432", "Q433", "Q434", "Q435", "Q436", "Q437", "Q438", "Q439", "Q440", "Q441", "Q442", "Q443", "Q444", "Q445", "Q446", "Q447", "Q448", "Q449", "Q450", "Q451", "Q452", "Q453", "Q454", "Q455", "Q456", "Q457", "Q458", "Q459", "Q460", "Q461", "Q462", "Q463", "Q464", "Q465", "Q466", "Q467", "Q468", "Q469", "Q470", "Q471", "Q472", "Q473", "Q474", "Q475", "Q476", "Q477", "Q478", "Q479", "Q480", "Q481", "Q482", "Q483", "Q484", "Q485", "Q486", "Q487", "Q488", "Q489", "Q490", "Q491", "Q492", "Q493", "Q494", "Q495", "Q496", "Q497", "Q498", "Q499", "Q500", "Q501", "Q502", "Q503", "Q504", "Q505", "Q506", "Q507", "Q508", "Q509", "Q510", "Q511", "Q512", "Q513", "Q514", "Q515", "Q516", "Q517", "Q518", "Q519", "Q520", "Q521", "Q522", "Q523", "Q524", "Q525", "Q526", "Q527", "Q528", "Q529", "Q530", "Q531", "Q532", "Q533", "Q534", "Q535", "Q536", "Q537", "Q538", "Q539", "Q540", "Q541", "Q542", "Q543", "Q544", "Q545", "Q546", "Q547", "Q548", "Q549", "Q550", "Q551", "Q552", "Q553", "Q554", "Q555", "Q556", "Q557", "Q558", "Q559", "Q560", "Q561", "Q562", "Q563", "Q564", "Q565", "Q566", "Q567", "Q568", "Q569", "Q570", "Q571", "Q572", "Q573", "Q574", "Q575", "Q576", "Q577", "Q578", "Q579", "Q580", "Q581", "Q582", "Q583", "Q584", "Q585", "Q586", "Q587", "Q588", "Q589", "Q590", "Q591", "Q592", "Q593", "Q594", "Q595", "Q596", "Q597", "Q598", "Q599", "Q600", "Q601", "Q602", "Q603", "Q604", "Q605", "Q606", "Q607", "Q608", "Q609", "Q610", "Q611", "Q612", "Q613", "Q614", "Q615", "Q616", "Q617", "Q618", "Q619", "Q620", "Q621", "Q622", "Q623", "Q624", "Q625", "Q626", "Q627", "Q628", "Q629", "Q630", "Q631", "Q632", "Q633", "Q634", "Q635", "Q636", "Q637", "Q638", "Q639", "Q640", "Q641", "Q642", "Q643", "Q644", "Q645", "Q646", "Q647", "Q648", "Q649", "Q650", "Q651", "Q652", "Q653", "Q654", "Q655", "Q656", "Q657", "Q658", "Q659", "Q660", "Q661", "Q662", "Q663", "Q664", "Q665", "Q666", "Q667", "Q668", "Q669", "Q670", "Q671", "Q672", "Q673", "Q674", "Q675", "Q676", "Q677", "Q678", "Q679", "Q680", "Q681", "Q682", "Q683", "Q684", "Q685", "Q686", "Q687", "Q688", "Q689", "Q690", "Q691", "Q692", "Q693", "Q694", "Q695", "Q696", "Q697", "Q698", "Q699", "Q700", "Q701", "Q702", "Q703", "Q704", "Q705", "Q706", "Q707", "Q708", "Q709", "Q710", "Q711", "Q712", "Q713", "Q714", "Q715", "Q716", "Q717", "Q718", "Q719", "Q720", "Q721", "Q722", "Q723", "Q724", "Q725", "Q726", "Q727", "Q728", "Q729", "Q730", "Q731", "Q732", "Q733", "Q734", "Q735", "Q736", "Q737", "Q738", "Q739", "Q740", "Q741", "Q742", "Q743", "Q744", "Q745", "Q746", "Q747", "Q748", "Q749", "Q750", "Q751", "Q752", "Q753", "Q754", "Q755", "Q756", "Q757", "Q758", "Q759", "Q760", "Q761", "Q762", "Q763", "Q764", "Q765", "Q766", "Q767", "Q768", "Q769", "Q770", "Q771", "Q772", "Q773", "Q774", "Q775", "Q776", "Q777", "Q778", "Q779", "Q780", "Q781", "Q782", "Q783", "Q784", "Q785", "Q786", "Q787", "Q788", "Q789", "Q790", "Q791", "Q792", "Q793", "Q794", "Q795", "Q796", "Q797", "Q798", "Q799", "Q800", "Q801", "Q802", "Q803", "Q804", "Q805", "Q806", "Q807", "Q808", "Q809", "Q810", "Q811", "Q812", "Q813", "Q814", "Q815", "Q816", "Q817", "Q818", "Q819", "Q820", "Q821", "Q822", "Q823", "Q824", "Q825", "Q826", "Q827", "Q828", "Q829", "Q830", "Q831", "Q832", "Q833", "Q834", "Q835", "Q836", "Q837", "Q838", "Q839", "Q840", "Q841", "Q842", "Q843", "Q844", "Q845", "Q846", "Q847", "Q848", "Q849", "Q850", "Q851", "Q852", "Q853", "Q854", "Q855", "Q856", "Q857", "Q858", "Q859", "Q860", "Q861", "Q862", "Q863", "Q864", "Q865", "Q866", "Q867", "Q868", "Q869", "Q870", "Q871", "Q872", "Q873", "Q874", "Q875", "Q876", "Q877", "Q878", "Q879", "Q880", "Q881", "Q882", "Q883", "Q884", "Q885", "Q886", "Q887", "Q888", "Q889", "Q890", "Q891", "Q892", "Q893", "Q894", "Q895", "Q896", "Q897", "Q898", "Q899", "Q900", "Q901", "Q902", "Q903", "Q904", "Q905", "Q906", "Q907", "Q908", "Q909", "Q910", "Q911", "Q912", "Q913", "Q914", "Q915", "Q916", "Q917", "Q918", "Q919", "Q920", "Q921", "Q922", "Q923", "Q924", "Q925", "Q926", "Q927", "Q928", "Q929", "Q930", "Q931", "Q932", "Q933", "Q934", "Q935", "Q936", "Q937", "Q938", "Q939", "Q940", "Q941", "Q942", "Q943", "Q944", "Q945", "Q946", "Q947", "Q948", "Q949", "Q950", "Q951", "Q952", "Q953", "Q954", "Q955", "Q956", "Q957", "Q958", "Q959", "Q960", "Q961", "Q962", "Q963", "Q964", "Q965", "Q966", "Q967", "Q968", "Q969", "Q970", "Q971", "Q972", "Q973", "Q974", "Q975", "Q976", "Q977", "Q978", "Q979", "Q980", "Q981", "Q982", "Q983", "Q984", "Q985", "Q986", "Q987", "Q988", "Q989", "Q990", "Q991", "Q992", "Q993", "Q994", "Q995", "Q996", "Q997", "Q998", "Q999", "Q1000")

# Create ethnicity table
# by mapping various detailed
# country-based ethnicities to
# broader groups
library(haven)
# Problem is WVS Q290 has too many
# gradations when we want to deal
# with a few classes that can allow
# us to overcome ethnic prejudices
ethnicities<-unique(as.character(as_factor(wvs7$Q290)))

reduced_ethnicities<-function(){
  mapeth<-rep("Other",length(ethnicities))
  mapeth[c(1,7,15,33,38,43,65,107,154,210,214)]<-"White"
  mapeth[c(3,223,222,51,52,53,54,55,56,45,39,36,11)]<-"Black"
  mapeth[c(6,13,19,20,21,22,67,70)]<-"Indian"
  mapeth[c(4,17,48,97,98,99,100)]<-"Arab"
  mapeth[c(5,14,16,62,63,64,68,72,73,74,75,76,77,
           78,79,80,81,82,83)]<-"East Asian"
  data.frame(key=ethnicities,val=mapeth)
}

ethnicity_map<-function( v ){
  emap<-reduced_ethnicities()
  w<-sapply(as.character(as_factor(v)),function(x) { out<-emap[which(emap$key==x),]$val; if(i
  #print(head(w))
  #w<-append(w,"Other")
  as.factor(unlist(w))
}
```

Date: May 5, 2021.

```

polv<-na.omit(wvs7[,vars])
polv$eth<-ethnicity_map(as_factor(polv$Q290))

# Fit P to distributions to Q177-Q195 in WVS 7
library(markovchain)

t10 <-1:10
nrm<-function(v)v/sum(v)

transitionMatrixFromPars<-function(x){
  P<-matrix(0,nrow=10,ncol=10)
  P[1,1]<-x[1]
  P[1:2,2]<-x[2:3]
  P[1:3,3]<-x[4:6]
  P[1:4,4]<-x[7:10]
  P[1:5,5]<-x[11:15]
  P[1:6,6]<-x[16:21]
  P[1:7,7]<-x[22:28]
  P[1:8,8]<-x[29:36]
  P[1:9,9]<-x[37:45]
  P[1:10,10]<-x[46:55]
  #symmetrize
  for (k in 2:10){
    P[k,1:k]<-P[1:k,k]
  }
  for (k in 1:10){
    P[k,]<-nrm(P[k,])
  }
  P
}

parsFromTransitionMatrix<-function(P){
  x<-c()
  for (k in 1:10){
    x<-append(x, P[1,1:k])
  }
  x
}

# We let x be the parameters of the subdiagonal
# of P in form appropriate for an optimiser
moral_markov_obj<-function( x ){

  P <- transitionMatrixFromPars( x )
  states <- as.character(1:10)

```

```

mcVals = new("markovchain",
             states = states,
             transitionMatrix = P,
             name = "Vals")
print('initialised markov chain')
Imtx <- matrix( 0, nrow=18,ncol=10)
Jmtx <- matrix( 0, nrow=18, ncol=10)
I1<-nrm(table(polv[, "Q177"]))
Imtx[1,] <- I1
print('set init distribution for Y1')
M<-500
Y1<-sample(1:10,M,replace=TRUE,prob=I1)
print('sampled Y1')
all.Y<-matrix(0,nrow=M,ncol=18)
all.Y[,1]<-Y1
for (kk in 1:M) {
  #print(kk)
  outs <- rmarkovchain(n = 17, object = mcVals, what = "list")
  all.Y[kk,2:18]<-outs
}

for ( r in 2:18 ){
  iv<-nrm(table(polv[,vars[r]]))
  jv<-nrm(table(all.Y[,r]))
  ivn<-rep(0,10)
  jvn<-rep(0,10)
  for ( kc in 1:10 ){
    a<-iv[as.character(kc)]
    b<-jv[as.character(kc)]
    if (!is.null(a)){
      ivn[kc]<-a
    }
    if (!is.null(b)){
      jvn[kc]<-b
    }
  }

  Imtx[r,]<-ivn
  Jmtx[r,]<-jvn
}

print('calculating l2 distance')
l2.dist <- 0
hits<-0
for (r in 1:18){
  d1<- norm(abs(Imtx[r,]),type="2")
  d2<- norm(abs(Jmtx[r,]),type="2")
  d <- norm( abs(Imtx[r,] - Jmtx[r,] ), type="2")
}

```

```

    #print(paste("r=",r,"d1=",d1,"d2=",d2,"d=",d))
    if (!is.na(d)){
      hits<-hits+1
      l2.dist <- l2.dist + d^2
    }
  }
  l2.dist<-sqrt(l2.dist)
  print(paste('hits=',hits))
  print(l2.dist)
  l2.dist
}

```

```

library(nloptr)
# Get an init value
lambda <- 0.52
p11 <- 0.6
P <- matrix(0,nrow = 10, ncol=10)
P[1,1] = p11
for (k in 2:10){
  for (r in 1:k){
    P[k,r] <- exp(-lambda*(k+r))*p11
    P[r,k] <- P[k,r]
  }
}
for ( r in 1:10){
  P[r,]<-P[r,]/sum(P[r,])
}
P0<-P

```

```

x0<-parsFromTransitionMatrix(P0)
xlen <- length(x0)
l0 <- rep(0,xlen)
u0 <- rep(1,xlen)

```

```

eval_g0<-function( x ){
  P1<-transitionMatrixFromPars(x)
  out<-0
  for (k in 1:10){
    out <- out + (sum(P1[k,]))^2-1.0
  }
  out
}

```

```

# Solve using NLOPT_LN_COBYLA without gradient information
res1 <- nloptr( x0=x0,

```

```

        eval_f=moral_markov_obj,
        lb = 10,
        ub = u0,
        eval_g_ineq = eval_g0,
        opts = list("algorithm"="NLOPT_LN_COBYLA",
                    "xtol_rel"=1.0e-8,"print_level"=1))
print( res1 )

```

1. SOME PROVISIONAL RESULTS

```
> res1
```

Call:

```

nloptr(x0 = x0, eval_f = moral_markov_obj, lb = 10, ub = u0,
      eval_g_ineq = eval_g0, opts = list(algorithm = "NLOPT_LN_COBYLA",
      xtol_rel = 1e-08, print_level = 1))

```

Minimization using NLOpt version 2.4.2

NLOpt solver status: 5 (NLOPT_MAXEVAL_REACHED: Optimization stopped because maxeval (above) was reached.)

Number of Iterations....: 100

Termination conditions: xtol_rel: 1e-08

Number of inequality constraints: 1

Number of equality constraints: 0

Current value of objective function: 0.944810719116277

Current value of controls: 0.998691 0.6017886 0.1230545 0.6382924 0.1523181 0.06756478
0.6521802 0.1243201 0.07645729 0.03616285 0.6788139 0.1106364
0.05256054 0.04208854 0.02963721 0.6604343 0.1345903
0.07503765 0.03138417 0.04702849 0.01516008 0.9375384
0.1186496 0.05600062 0.03639242 0.02682043 0.01248168
0.01596603 0.6248498 0.1129891 0.0487273 0.04372557
0.03241158 0.02317385 0.01029844 0.00726488 0.6085975
0.09877426 0.06007426 0.0613663 0.02351311 0.01476094
0.00509009 0.007064568 0.001769594 0.6950101 0.1412725
0.08852981 0.03363069 0.03376865 0.01791395 0.01243466
0.007366109 0.003654178 0.002536741