

Incumbency simulation in R

inputs
G = number of simulations to use
gamma
sigma2.theta

incumbency.function <- function(G=100000, gamma=0, sigma2.theta=1, 
sigma2.epsilon=1, sigma2.xstar=2, sigma2.rw=0) {

  # first level simulation to calculate the normal vote
  thetal <- rnorm(G, mean=0, sd=sqrt(sigma2.theta))
  epsilonL <- rnorm(G, mean=0, sd=sqrt(sigma2.epsilon))
  epsilonR <- rnorm(G, mean=0, sd=sqrt(sigma2.epsilon))
  etal <- rnorm(G, mean=gamma, sd=sqrt(sigma2.xstar))
  lambdaL1 <- rep(sigma2.theta / (sigma2.theta + sigma2.epsilon), G)
  SL1 <- thetal + epsilonL
  SR1 <- thetal + epsilonR
  mL1 <- lambdaL1 * SL1
  mR1 <- lambdaL1 * SR1

  # The normal vote is
  reelect1 <- sum(mL1 - mR1 >= etal) / (sum(mL1 - mR1 >= etal) + sum(mL1 - mR1 < etal))

  # yank off the first case keepig only those who won and run a second election
  # The incumbent's ability given that they won stays the same
  thetal2 <- thetal[mL1 - mR1 >= etal]
  epsilonL2 <- rnorm(length(thetal2), mean=0, sd=sqrt(sigma2.theta))
  epsilonR2 <- rnorm(length(thetal2), mean=0, sd=sqrt(sigma2.epsilon))
  eta2 <- rnorm(length(thetal2), mean=gamma, sd=sqrt(sigma2.xstar))
  lambdaL2 <- lambdaL1[mL1 - mR1 >= etal]
  lambdaL2 <- (lambdaL2 * sigma2.epsilon + sigma2.rw) / (lambdaL2 * sigma2.theta + sigma2.rw + sigma2.epsilon)
  lambdaR2 <- rep(sigma2.theta / (sigma2.theta + sigma2.epsilon), length(thetal2))

  # The signals
  SL2 <- thetal2 + epsilonL2
  SR2 <- thetal2 + epsilonR2

  # The posteriors
  mL2 <- lambdaL2 * SL2 + (1 - lambdaL2) * mL1[mL1 - mR1 >= etal]
  mR2 <- lambdaR2 * SR2

  # Probability incumbent wins
  reelect2 <- sum(mL2 - mR2 >= eta2) / sum(mL2 - mR2 >= eta2)

  # Find the left-wing incumbency advantage by comparing probability incumbent wins to the normal vote
  iaL <- reelect2 - reelect1

  sigma <- sqrt(((2*(sigma2.theta^2))/(sigma2.theta + sigma2.epsilon)) + sigma2.xstar)
  nv <- 1 - pnorm((gamma / sigma))

  cat("iaL ", iaL, ", nv", "nv", nv, "\n")
  return(c(iaL, nv))
}
ruler <- seq(-4, 4, .25)
storage.matrix <- matrix(NA, length(ruler), 2)

# put the output into a storage matrix
for(i in ruler) {
  storage.matrix[count, 1:2] <- incumbency.function(G=100000, gamma=i)
  count <- count + 1
}

# create the final matrix
storage.matrix2 <- matrix(NA, length(seq(ruler)), 4)

# now put back in the original stuff (lw incumbency advantage and normal vote)
storage.matrix2[, 1:2] <- storage.matrix

# now add the right wing incumbency advantage which is the mirror image of the left-wing incumbency advantage
for(j in seq(1, length(ruler))) storage.matrix2[j, 3] <- storage.matrix2[34-j, 1]

# now add the average per district incumbency advantage, which is a weighted average of the two incumbent
# weighted by the nv
for(k in seq(1, length(ruler))) storage.matrix2[k, 4] <-
  storage.matrix2[k, 2] * storage.matrix2[k, 1] + (1-storage.matrix2[k, 2]) *
  storage.matrix2[k, 3]

## Due to bad programming, change the parameter values and labels by hand to create the figures
## epsilon figure
postscript(file = "c:\latex\incumbency\figuresR\epsilon/epsilonnew35.eps", horizontal = FALSE, paper = "letter")
par(cex=2)
plot(1-storage.matrix2[, 2], storage.matrix2[, 4],
  type="l", xlab="right-wing normal vote", ylab="average incumbency advantage",
  ylim=c(0, 10), main=expression(paste((sigma^2) [theta] == 1, "", ",",
  (sigma^2) [epsilon] == 1)))
dev.off()

## eta figure
postscript(file = "c:\latex\incumbency\figuresR\eta/etanew2.eps", horizontal = FALSE, paper = "letter")
par(cex=2)
plot(1-storage.matrix2[, 2], storage.matrix2[, 4],
  type="l", xlab="right-wing normal vote", ylab="average incumbency advantage",
  ylim=c(0, 10), main=expression(paste((sigma^2) [theta] == 1, "", ",",
  (sigma^2) [epsilon] == 2)))
dev.off()