```
ethans.little.function <- function(G=50000, gamma=0, kappa=0.001, sigma2.theta=3,
 sigma2.epsilon=1,  sigma2.rw=0, i) {


    #Define the first value of sigma2.xstar
    sigma2.xstar=0.5
   # first level simulation
   thetaL <- rnorm(G, mean=0, sd=sqrt(sigma2.theta))
   thetaR <- 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))
   eta1 <- rnorm(G, mean=gamma, sd=sqrt(sigma2.xstar))
   lambda1 <- rep(sigma2.theta / (sigma2.theta + sigma2.epsilon), G)
   SL1 <- thetaL + epsilonL
   SR1 <- thetaR + epsilonR
   mL1 <- lambda1 * SL1
   mR1 <- lambda1 * SR1

reelect1 <- sum(mL1 - mR1 >= eta1) / (sum(mL1 - mR1 >= eta1) + sum(mL1 - mR1 < eta1))

   # yank off the first case
   thetaL12 <- thetaL[mL1 - mR1 >= eta1]
   thetaR12 <- rnorm(length(thetaL12), mean=0, sd=sqrt(sigma2.theta))
   epsilonL12 <- rnorm(length(thetaL12), mean=0, sd=sqrt(sigma2.epsilon))
   epsilonR12 <- rnorm(length(thetaL12), mean=0, sd=sqrt(sigma2.epsilon))
   eta12 <- rnorm(length(thetaL12), mean=gamma, sd=sqrt(sigma2.xstar))
   lambdaL112 <- lambda1[mL1 - mR1 >= eta1]
   lambdaL12 <- (lambdaL112 * sigma2.epsilon + sigma2.rw) / (lambdaL112 * sigma2.epsilon
       + sigma2.rw + sigma2.epsilon)
   lambdaR12 <- rep(sigma2.theta / (sigma2.theta + sigma2.epsilon), length(thetaL12))
   sigma <- rep(sqrt(sigma2.epsilon*((((sigma2.theta / (sigma2.theta +
   sigma2.epsilon))*sigma2.epsilon)/(((sigma2.theta / (sigma2.theta +
   sigma2.epsilon))*sigma2.epsilon)+sigma2.epsilon))^2)+sigma2.xstar + (sigma2.theta /
   (sigma2.theta + sigma2.epsilon))*sigma2.theta), length(thetaL12))
   #sigma <- (lambdaL2)^2*sigma2.epsilon + sigma2.xstar+ (sigma2.theta / (sigma2.theta +
   sigma2.epsilon))*sigma2.theta)
   SL12 <- thetaL12 + epsilonL12
   SR12 <- thetaR12 + epsilonR12
   mL12 <- lambdaL12 * SL12 + (1 - lambdaL12) * mL1[mL1 - mR1 >= eta1]
#Assign mR12 to be the real value if there is a challenge and a value that loses for sure if
#there is not a real challenge
   mR12 <- ifelse(1-pnorm(mL1[mL1 - mR1 >= eta1]/sigma) >= kappa, lambdaR12 * SR12, mL12 - eta12 -
   1)


#Probability of left wing guy achieving reelection, conditioning on incumbency

      reelect2 <- sum(mL12 - mR12 >= eta12) / sum(mL1 - mR1 >= eta1)



#Now calculate the increased probability of winning given incumbency

iaL <- reelect2 - reelect1



#######
#Now do this all again for the next value of sigma2.xstar

    #Define the next value of sigma2.xstar
    sigma2.xstar2=1
   # first level simulation
   thetaL2 <- rnorm(G, mean=0, sd=sqrt(sigma2.theta))
   thetaR2 <- rnorm(G, mean=0, sd=sqrt(sigma2.theta))
   epsilonL2 <- rnorm(G, mean=0, sd=sqrt(sigma2.epsilon))
```

```
   epsilonR2 <- rnorm(G, mean=0, sd=sqrt(sigma2.epsilon))
   eta12 <- rnorm(G, mean=gamma, sd=sqrt(sigma2.xstar2))
   lambda12 <- rep(sigma2.theta / (sigma2.theta + sigma2.epsilon), G)
   SL12 <- thetaL2 + epsilonL2
   SR12 <- thetaR2 + epsilonR2
   mL12 <- lambda12 * SL12
   mR12 <- lambda12 * SR12

reelect12 <- sum(mL12 - mR12 >= eta12) / (sum(mL12 - mR12 >= eta12) + sum(mL12 - mR12 < eta12))

   # yank off the first case
   thetaL22 <- thetaL2[mL12 - mR12 >= eta12]
   thetaR22 <- rnorm(length(thetaL22), mean=0, sd=sqrt(sigma2.theta))
   epsilonL22 <- rnorm(length(thetaL22), mean=0, sd=sqrt(sigma2.epsilon))
   epsilonR22 <- rnorm(length(thetaL22), mean=0, sd=sqrt(sigma2.epsilon))
   eta22 <- rnorm(length(thetaL22), mean=gamma, sd=sqrt(sigma2.xstar2))
   lambdaL122 <- lambda12[mL12 - mR12 >= eta12]
   lambdaL22 <- (lambdaL122 * sigma2.epsilon + sigma2.rw) / (lambdaL122 * sigma2.epsilon
       + sigma2.rw + sigma2.epsilon)
   lambdaR22 <- rep(sigma2.theta / (sigma2.theta + sigma2.epsilon), length(thetaL22))
  sigma2 <- rep(sqrt(sigma2.epsilon*((((sigma2.theta / (sigma2.theta +
  sigma2.epsilon))*sigma2.epsilon)/(((sigma2.theta / (sigma2.theta +
  sigma2.epsilon))*sigma2.epsilon)+sigma2.epsilon))^2)+sigma2.xstar2 + (sigma2.theta /
  (sigma2.theta + sigma2.epsilon))*sigma2.theta), length(thetaL22))
  # sigma2 <- (lambdaL22)^2*sigma2.epsilon + sigma2.xstar2+ (sigma2.theta / (sigma2.theta +
  sigma2.epsilon))*sigma2.theta)
   SL22 <- thetaL22 + epsilonL22
   SR22 <- thetaR22 + epsilonR22
   mL22 <- lambdaL22 * SL22 + (1 - lambdaL22) * mL12[mL12 - mR12 >= eta12]
#Assign mR22 to be the real value if there is a challenge and a value that loses for sure
#if there is not a real challenge
   mR22 <- ifelse(1-pnorm(mL12[mL12 - mR12 >= eta12]/sigma2) >= kappa, lambdaR22 * SR22, mL22 -
   eta22 - 1)


#Probability of left wing guy achieving reelection, conditioning on incumbency

      reelect22 <- sum(mL22 - mR22 >= eta22) / sum(mL12 - mR12 >= eta12)



#Now calculate the increased probability of winning given incumbency

iaL2 <- reelect22 - reelect12





#################AGAIN
#Now do this all again for the next value of sigma2.xstar

    #Define the next value of sigma2.xstar
    sigma2.xstar3=1.5
   # first level simulation
   thetaL3 <- rnorm(G, mean=0, sd=sqrt(sigma2.theta))
   thetaR3 <- rnorm(G, mean=0, sd=sqrt(sigma2.theta))
   epsilonL3 <- rnorm(G, mean=0, sd=sqrt(sigma2.epsilon))
   epsilonR3 <- rnorm(G, mean=0, sd=sqrt(sigma2.epsilon))
   eta13 <- rnorm(G, mean=gamma, sd=sqrt(sigma2.xstar3))
   lambda13 <- rep(sigma2.theta / (sigma2.theta + sigma2.epsilon), G)
   SL13 <- thetaL3 + epsilonL3
   SR13 <- thetaR3 + epsilonR3
   mL13 <- lambda13 * SL13
   mR13 <- lambda13 * SR13

reelect13 <- sum(mL13 - mR13 >= eta13) / (sum(mL13 - mR13 >= eta13) + sum(mL13 - mR13 < eta13))
```

```
   # yank off the first case
   thetaL23 <- thetaL3[mL13 - mR13 >= eta13]
   thetaR23 <- rnorm(length(thetaL23), mean=0, sd=sqrt(sigma2.theta))
   epsilonL23 <- rnorm(length(thetaL23), mean=0, sd=sqrt(sigma2.epsilon))
   epsilonR23 <- rnorm(length(thetaL23), mean=0, sd=sqrt(sigma2.epsilon))
   eta23 <- rnorm(length(thetaL23), mean=gamma, sd=sqrt(sigma2.xstar3))
   lambdaL123 <- lambda13[mL13 - mR13 >= eta13]
   lambdaL23 <- (lambdaL123 * sigma2.epsilon + sigma2.rw) / (lambdaL123 * sigma2.epsilon
       + sigma2.rw + sigma2.epsilon)
   lambdaR23 <- rep(sigma2.theta / (sigma2.theta + sigma2.epsilon), length(thetaL23))
  sigma3 <- rep(sqrt(sigma2.epsilon*((((sigma2.theta / (sigma2.theta +
  sigma2.epsilon))*sigma2.epsilon)/(((sigma2.theta / (sigma2.theta +
  sigma2.epsilon))*sigma2.epsilon)+sigma2.epsilon))^2)+sigma2.xstar3 + (sigma2.theta /
  (sigma2.theta + sigma2.epsilon))*sigma2.theta), length(thetaL23))
  # sigma3 <- (lambdaL23)^2*sigma2.epsilon + sigma2.xstar3+ (sigma2.theta / (sigma2.theta +
  sigma2.epsilon))*sigma2.theta)
   SL23 <- thetaL23 + epsilonL23
   SR23 <- thetaR23 + epsilonR23
   mL23 <- lambdaL23 * SL23 + (1 - lambdaL23) * mL13[mL13 - mR13 >= eta13]
#Assign mR23 to be the real value if there is a challenge and a value that loses for sure if there
is not a real challenge
   mR23 <- ifelse(1-pnorm(mL13[mL13 - mR13 >= eta13]/sigma3) >= kappa, lambdaR23 * SR23, mL23 -
   eta23 - 1)


#Probability of left wing guy achieving reelection, conditioning on incumbency

      reelect23 <- sum(mL23 - mR23 >= eta23) / sum(mL13 - mR13 >= eta13)



#Now calculate the increased probability of winning given incumbency

iaL3 <- reelect23 - reelect13



############AGAIN

#Now do this all again for the next value of sigma2.xstar

    #Define the next value of sigma2.xstar
    sigma2.xstar4=2
   # first level simulation
   thetaL4 <- rnorm(G, mean=0, sd=sqrt(sigma2.theta))
   thetaR4 <- rnorm(G, mean=0, sd=sqrt(sigma2.theta))
   epsilonL4 <- rnorm(G, mean=0, sd=sqrt(sigma2.epsilon))
   epsilonR4 <- rnorm(G, mean=0, sd=sqrt(sigma2.epsilon))
   eta14  <- rnorm(G, mean=gamma, sd=sqrt(sigma2.xstar4))
   lambda14 <- rep(sigma2.theta / (sigma2.theta + sigma2.epsilon), G)
   SL14 <- thetaL4 + epsilonL4
   SR14 <- thetaR4 + epsilonR4
   mL14 <- lambda14 * SL14
   mR14 <- lambda14 * SR14

reelect14 <- sum(mL14 - mR14 >= eta14) / (sum(mL14 - mR14 >= eta14) + sum(mL14 - mR14 < eta14))

   # yank off the first case
   thetaL24 <- thetaL4[mL14 - mR14 >= eta14]
   thetaR24 <- rnorm(length(thetaL24), mean=0, sd=sqrt(sigma2.theta))
   epsilonL24 <- rnorm(length(thetaL24), mean=0, sd=sqrt(sigma2.epsilon))
   epsilonR24 <- rnorm(length(thetaL24), mean=0, sd=sqrt(sigma2.epsilon))
   eta24 <- rnorm(length(thetaL24), mean=gamma, sd=sqrt(sigma2.xstar4))
   lambdaL124 <- lambda14[mL14 - mR14 >= eta14]
   lambdaL24 <- (lambdaL124 * sigma2.epsilon + sigma2.rw) / (lambdaL124 * sigma2.epsilon
```

```
        + sigma2.rw + sigma2.epsilon)
    lambdaR24 <- rep(sigma2.theta / (sigma2.theta + sigma2.epsilon), length(thetaL24))
   sigma4 <- rep(sqrt(sigma2.epsilon*((((sigma2.theta / (sigma2.theta +
   sigma2.epsilon))*sigma2.epsilon)/(((sigma2.theta / (sigma2.theta +
   sigma2.epsilon))*sigma2.epsilon)+sigma2.epsilon))^2)+sigma2.xstar4 + (sigma2.theta /
   (sigma2.theta + sigma2.epsilon))*sigma2.theta), length(thetaL24))
   # sigma4 <- (lambdaL24)^2*sigma2.epsilon + sigma2.xstar4+ (sigma2.theta / (sigma2.theta +
   sigma2.epsilon))*sigma2.theta)
    SL24 <- thetaL24 + epsilonL24
    SR24 <- thetaR24 + epsilonR24
    mL24 <- lambdaL24 * SL24 + (1 - lambdaL24) * mL14[mL14 - mR14 >= eta14]
#Assign mR24 to be the real value if there is a challenge and a value that loses for sure if there
is not a real challenge
    mR24 <- ifelse(1-pnorm(mL14[mL14 - mR14 >= eta14]/sigma4) >= kappa, lambdaR24 * SR24, mL24 -
    eta24 - 1)


#Probability of left wing guy achieving reelection, conditioning on incumbency

      reelect24 <- sum(mL24 - mR24 >= eta24) / sum(mL14 - mR14 >= eta14)



#Now calculate the increased probability of winning given incumbency

iaL4 <- reelect24 - reelect14

#Now do this all again for the next value of sigma2.xstar

     #Define the next value of sigma2.xstar
     sigma2.xstar5=2.5
    # first level simulation
    thetaL5 <- rnorm(G, mean=0, sd=sqrt(sigma2.theta))
    thetaR5 <- rnorm(G, mean=0, sd=sqrt(sigma2.theta))
    epsilonL5 <- rnorm(G, mean=0, sd=sqrt(sigma2.epsilon))
    epsilonR5 <- rnorm(G, mean=0, sd=sqrt(sigma2.epsilon))
    eta15  <- rnorm(G, mean=gamma, sd=sqrt(sigma2.xstar5))
    lambda15 <- rep(sigma2.theta / (sigma2.theta + sigma2.epsilon), G)
    SL15 <- thetaL5 + epsilonL5
    SR15 <- thetaR5 + epsilonR5
    mL15 <- lambda15 * SL15
    mR15 <- lambda15 * SR15

reelect15 <- sum(mL15 - mR15 >= eta15) / (sum(mL15 - mR15 >= eta15) + sum(mL15 - mR15 < eta15))

    # yank off the first case
    thetaL25 <- thetaL5[mL15 - mR15 >= eta15]
    thetaR25 <- rnorm(length(thetaL25), mean=0, sd=sqrt(sigma2.theta))
    epsilonL25 <- rnorm(length(thetaL25), mean=0, sd=sqrt(sigma2.epsilon))
    epsilonR25 <- rnorm(length(thetaL25), mean=0, sd=sqrt(sigma2.epsilon))
    eta25 <- rnorm(length(thetaL25), mean=gamma, sd=sqrt(sigma2.xstar5))
    lambdaL125 <- lambda15[mL15 - mR15 >= eta15]
    lambdaL25 <- (lambdaL125 * sigma2.epsilon + sigma2.rw) / (lambdaL125 * sigma2.epsilon
        + sigma2.rw + sigma2.epsilon)
    lambdaR25 <- rep(sigma2.theta / (sigma2.theta + sigma2.epsilon), length(thetaL25))
   sigma5 <- rep(sqrt(sigma2.epsilon*((((sigma2.theta / (sigma2.theta +
   sigma2.epsilon))*sigma2.epsilon)/(((sigma2.theta / (sigma2.theta +
   sigma2.epsilon))*sigma2.epsilon)+sigma2.epsilon))^2)+sigma2.xstar5 + (sigma2.theta /
   (sigma2.theta + sigma2.epsilon))*sigma2.theta), length(thetaL25))
   # sigma5 <- (lambdaL25)^2*sigma2.epsilon + sigma2.xstar5+ (sigma2.theta / (sigma2.theta +
   sigma2.epsilon))*sigma2.theta)
    SL25 <- thetaL25 + epsilonL25
    SR25 <- thetaR25 + epsilonR25
    mL25 <- lambdaL25 * SL25 + (1 - lambdaL25) * mL15[mL15 - mR15 >= eta15]
#Assign mR25 to be the real value if there is a challenge and a value that loses for sure if there
is not a real challenge
```

```
   mR25 <- ifelse(1-pnorm(mL15[mL15 - mR15 >= eta15]/sigma5) >= kappa, lambdaR25 * SR25, mL25 -
   eta25 - 1)


#Probability of left wing guy achieving reelection, conditioning on incumbency

      reelect25 <- sum(mL25 - mR25 >= eta25) / sum(mL15 - mR15 >= eta15)



#Now calculate the increased probability of winning given incumbency

iaL5 <- reelect25 - reelect15


#Now do this all again for the next value of sigma2.xstar

    #Define the next value of sigma2.xstar
    sigma2.xstar6=3
   # first level simulation
   thetaL6 <- rnorm(G, mean=0, sd=sqrt(sigma2.theta))
   thetaR6 <- rnorm(G, mean=0, sd=sqrt(sigma2.theta))
   epsilonL6 <- rnorm(G, mean=0, sd=sqrt(sigma2.epsilon))
   epsilonR6 <- rnorm(G, mean=0, sd=sqrt(sigma2.epsilon))
   eta16  <- rnorm(G, mean=gamma, sd=sqrt(sigma2.xstar6))
   lambda16 <- rep(sigma2.theta / (sigma2.theta + sigma2.epsilon), G)
   SL16 <- thetaL6 + epsilonL6
   SR16 <- thetaR6 + epsilonR6
   mL16 <- lambda16 * SL16
   mR16 <- lambda16 * SR16

reelect16 <- sum(mL16 - mR16 >= eta16) / (sum(mL16 - mR16 >= eta16) + sum(mL16 - mR16 < eta16))

   # yank off the first case
   thetaL26 <- thetaL6[mL16 - mR16 >= eta16]
   thetaR26 <- rnorm(length(thetaL26), mean=0, sd=sqrt(sigma2.theta))
   epsilonL26 <- rnorm(length(thetaL26), mean=0, sd=sqrt(sigma2.epsilon))
   epsilonR26 <- rnorm(length(thetaL26), mean=0, sd=sqrt(sigma2.epsilon))
   eta26 <- rnorm(length(thetaL26), mean=gamma, sd=sqrt(sigma2.xstar6))
   lambdaL126 <- lambda16[mL16 - mR16 >= eta16]
   lambdaL26 <- (lambdaL126 * sigma2.epsilon + sigma2.rw) / (lambdaL126 * sigma2.epsilon
       + sigma2.rw + sigma2.epsilon)
   lambdaR26 <- rep(sigma2.theta / (sigma2.theta + sigma2.epsilon), length(thetaL26))
  sigma6 <- rep(sqrt(sigma2.epsilon*((((sigma2.theta / (sigma2.theta +
  sigma2.epsilon))*sigma2.epsilon)/(((sigma2.theta / (sigma2.theta +
  sigma2.epsilon))*sigma2.epsilon)+sigma2.epsilon))^2)+sigma2.xstar6 + (sigma2.theta /
  (sigma2.theta + sigma2.epsilon))*sigma2.theta), length(thetaL26))
  # sigma6 <- (lambdaL26)^2*sigma2.epsilon + sigma2.xstar6+ (sigma2.theta / (sigma2.theta +
  sigma2.epsilon))*sigma2.theta)
   SL26 <- thetaL26 + epsilonL26
   SR26 <- thetaR26 + epsilonR26
   mL26 <- lambdaL26 * SL26 + (1 - lambdaL26) * mL16[mL16 - mR16 >= eta16]
#Assign mR26 to be the real value if there is a challenge and a value that loses for sure if there
is not a real challenge
   mR26 <- ifelse(1-pnorm(mL16[mL16 - mR16 >= eta16]/sigma6) >= kappa, lambdaR26 * SR26, mL26 -
   eta26 - 1)


#Probability of left wing guy achieving reelection, conditioning on incumbency

      reelect26 <- sum(mL26 - mR26 >= eta26) / sum(mL16 - mR16 >= eta16)



#Now calculate the increased probability of winning given incumbency
```

```
iaL6 <- reelect26 - reelect16




   cat("iaL ", iaL, "\n", "iaL2 ", iaL2, "\n", "iaL3 ", iaL3, "\n","iaL4 ", iaL4, "\n","iaL5 ",
   iaL5, "\n","iaL6 ", iaL6, "\n")
   return(c(iaL, iaL2, iaL3, iaL4, iaL5, iaL6))
   }

#create the final incumbency advantage matrix

ruler<- seq(0,.5,.05)
storage.matrix <- matrix(NA, length(ruler), 7)

#Put the incumbency advantage for each value of sigma2.xstar
#into the storage matrix and repeat with a higher value of kappa
count <- 1
for(i in ruler) {
  storage.matrix[count,1:6] <- ethans.little.function(G=50000, kappa=i)
  count <- count + 1
}

#Put increments of kappa into the storage matrix

storage.matrix[,7] <- seq(0,0.5,0.05)

#Plot the values of kappa on the x-axis against the incumbency advantage under each type of
#sigma2.xstar on the y-axis

postscript(file = "c:/latex/incumbency/figuresR/kappa/kappa3.eps", horizontal = FALSE, paper =
"letter")
par(cex=2)

plot(storage.matrix[,7], storage.matrix[,1], type="l",ylim=c(0,.35), xlab="Recruitment Cost",
ylab="Incumbency Advantage",
 main=expression(paste({sigma^2} [theta] == 3, ", ",
     {sigma^2} [epsilon] == 1)))

lines(storage.matrix[,7], storage.matrix[,2])
lines(storage.matrix[,7], storage.matrix[,3])
lines(storage.matrix[,7], storage.matrix[,4])
lines(storage.matrix[,7], storage.matrix[,5])
lines(storage.matrix[,7], storage.matrix[,6])
dev.off()
```