Monty Hall in R

numsim = 100000

doors = 1:3

opendoor = function(x) {

if (x[1]==x[2])

return(sample(doors[-c(x[1])], 1))

else return(doors[-c(x[1],x[2])])

}

swapdoor = function(x) { return(doors[-c(x[1], x[2])]) }

winner = sample(doors, numsim, replace=TRUE)

choice = sample(doors, numsim, replace=TRUE)

open = apply(cbind(winner, choice), 1, opendoor)

newchoice = apply(cbind(open, choice), 1, swapdoor)

#

cat("without switching, won ",

round(sum(winner==choice)/numsim\*100,1),"

percent of the time.\n", sep="")

cat("with switching, won ",

round(sum(winner==newchoice)/numsim\*100,1),"

percent of the time.\n",

Monty Hall in Python

*#!/usr/bin/python*

**import** **sys**,**random**

rounds = 10000

wins = 0

random.seed()

*# The "-n" commandline option makes us run without ever switching.*

**if** len(sys.argv) > 1 **and** sys.argv[1] == "-n":

swap = False

**else**:

swap = True

**for** i **in** xrange(rounds) :

*# Generate random door contents*

doors = ["goat", "goat", "car"]

random.shuffle(doors)

*# Pick a door*

door\_choice = random.randrange(3)

**print** "Selecting door", door\_choice

*# Show a door with a goat*

**for** j, contents **in** enumerate(doors):

**if** j != door\_choice **and** contents == "goat":

goat\_door = j

**print** "The host reveals a goat behind door", goat\_door

**break**

**if** swap:

*# Swap to the other door*

**for** j, contents **in** enumerate(doors):

**if** j != door\_choice **and** j != goat\_door:

swap\_to = j

**print** "Swapping to door", swap\_to

**else**:

swap\_to = door\_choice

**if** doors[swap\_to] == "car":

wins += 1

**print** "You won the car!!"

**else**:

**print** "Sorry, but you're stuck with a goat"

**print** "You played", rounds, "rounds, and won", wins, "of them!"