The Anatomy of a Simple Monte Carlo Simulation Program

Based on a grand loop (as usually written in a low-level language like Fortran, C etc)

Source: Don Edwards, University of South Carolina

This consists of iterating, many times, a Basic Experiment (B.E.) which has random inputs. Each time we do the B.E., we make certain measurements on the outcome(s) and/or check for occurrence or non-occurrence of key events.

1. Create / read in simulation parameters
   For example: overall simulation size (Msim), parameters of distributions of random inputs, data structures that will be constant throughout all the Msim iterations of the B.E.

2. Initialize (usually to 0 or NA) observational variables for the simulation
   For example: counters for occurrences of key events, holding structures for numeric (non-0-1) measured outcomes of the B.E.

3. Begin the grand loop
   In R, for example: `for (i in 1:Msim) {...`

4. Generate a single iteration of the Basic Experiment
   For example: generate a value for each of the random inputs, and do all manipulations / calculations necessary to produce the observational variables' values for this iteration.

5. Measure observational variables and update counters and/or holding structures for measured outcomes.
   For example: add 1 to counters of key events that occurred on this iteration of the B.E.; store values of numeric outcomes measured on this iteration

6. End the grand loop.
   In R, for example: `}  # end grand loop`

7. Calculate final summaries and create graphical descriptives
   For example: estimates of probabilities of key events = final counter values / Msim. Make (e.g.) histograms and compute (e.g.) means and standard deviations of measured numeric outcomes over the Msim iterations. Possibly calculate confidence intervals for true probabilities of key events and (e.g.) long-run averages of numeric outcomes. Be creative in making graphical summaries of the results.

8. End the simulation program
   If the work has been done by a subroutine or R / Splus function, send back to the main program all information to be saved. In an R function this usually involves forming a data frame or list including all the summaries that are to be saved to the workspace after the function ends, and in the last function statement simply naming this object.

Example: Let’s play cards ☺

```
#Uncommented Program
poker.loop<-function(Msim=10000,n=5)
{
   denom<-rep(1:13,4)
   suit<-rep(c("S","H","D","C"),each=13)
   carddeck<-data.frame(denom,suit)

   count.flush<-0

   for(i in 1:Msim)
   {
         select<-sample(nrow(carddeck),n)
         hand<-carddeck[select,]
         selectsuits <- unique(hand[,2])
         numsuits <- length(selectsuits)
         if (numsuits==1) {count.flush<-count.flush+1}
   } # close grand loop

   p.flush<-count.flush/Msim
   out<-list(Msim,p.flush)
   names(out)<-c("Msim","p.flush")
   out
}
```

Notes on Flushes:

There are 5,148 possible flushes, of which 40 are also straight flushes; the probability of being dealt a flush that is not also a straight flush, in a five-card hand is

$$\frac{4 \cdot C_{13}^{5} - 40}{2{,}598{,}960} = \frac{4 \cdot 1{,}287 - 40}{2{,}598{,}960} = \frac{5{,}108}{2{,}598{,}960} \approx 0.20\%$$

Source: Wikipedia

```
#Commented Program
poker.loop<-function(Msim=10000,n=5)
{

   # a function to simulate Poker draws from
   # a standard card deck
   # this is the Grand Loop approach

   # make the card deck
   denom<-rep(1:13,4)
   suit<-rep(c("S","H","D","C"),each=13)
   carddeck<-data.frame(denom,suit)

   # initialize flush counter and storage
   count.flush<-0

   # begin the grand loop
   for(i in 1:Msim)
   {
         # determine card numbers for this hand
```

```
        select<-sample(nrow(carddeck),n)

        # select rows from the card deck for this hand
        hand<-carddeck[select,]

        # determine suit of cards in hand
        selectsuits <- unique(hand[,2])

        # count number of suits in hand
        numsuits <- length(selectsuits)

        # increment counter if there is only one suit (a flush) is in our hand
        if (numsuits==1) {count.flush<-count.flush+1}
  } # close grand loop to repeat drawing poker hands Msim times

  # final flush probability calculation
  p.flush<-count.flush/Msim

  # output results
  out<-list(Msim,p.flush)
  names(out)<-c("Msim","p.flush")
  out
}
```