61.09 Probabilidad y Estadística
Trabajo Práctico de Simulación

13 de mayo de 2012

Grupo 1
Loiza, Samanta (91935)
Aguilera, Juan Martín (92483)
Rossi, Federico Martín (92086)

1. Introducción

Con el fin de fortalecer las nociones fundamentales de la probabilidad es que este trabajo busca aplicar las bases de los conceptos ya estudiados en el curso de manera tal de adoptar una mejor comprensión de la inferencia estadística en hechos cotidianos. En este caso, nuestro trabajo será explorar las propiedades estádisticas de cierto tipo de secuencias, las cuales serán detalladas más adelante.

En cada análisis realizado se mostrará su resolución analítica, como así también algoritmos que simulen dichas situaciones, a fin de poder mostrar una aproximación a la constatación empírica de los resultados. Estos últimos se han hecho en el lenguaje de programación *GNU Octave*[1].

Todos los archivos y códigos fuente aquí mencionados, así como también el presente informe, pueden ser descargados de la sección *Downloads* del repositorio del grupo (*http://code.google.com/p/simulacion-proba2012*).

2. Actividades previas

Tal como lo sugiere el título de este apartado, se mostrarán a continuación una serie de actividades previas necesarias para dominar las técnicas básicas de simulación de números aleatorios.

2.1. Una primera simulación

A continuación, en el *Código 1*, se muestra una rutina que, a partir de un generador de números pseudo-aleatorios, permite simular los valores de un dado. Como se puede ver, se ha utilizado la función nativa ***rand***, la cual devuelve una matriz (que es utilizada como vector) con valores aleatrios entre 0 y 1. Para ajustarla a nuestras necesidades, o sea, obtener valores aleatorios entre 1 y 6 hemos multiplicado dicho vector por 6. De esta manera pasa a contener valores aleatorios entre 0 y 6. Luego, mediante otra funcion nativa llamada ***ceil***, se redondea a los valores enteros próximos superiores de cada uno (ceiling), obteniendo como resultado un vector con valores enteros entre 1 y 6.

Código 1: *randomDado.m*

```
# Funcion que simula la tirada de n tiradas de un dado.
# PRE: 'n' es la cantidad de veces que se desea lanzar el dado.
# POST: se devuelve un vector con los valores obtenidos en los lanzamientos.
function listaDeValores = randomDado(n)

  listaDeValores = ceil(rand(1,n)*6);

endfunction
```

[1] "*GNU Octave* es un lenguaje interpretado de alto nivel, pensado principalmente para cálculos numéricos. Para más información dirijase a *http://www.gnu.org/software/octave* "

2.2. Puesta a prueba

Pondremos a prueba ahora lo visto en el apartado anterior estimando las probabilidades de cada uno de los 6 valores posibles utilizando los resultados obtenidos en 1000 ejecuciones de la rutina. [Colocar contenido aquí]

2.3. Predicados

A veces se necesita estimar la probabilidad de un evento definido a partir de variables aleatorias. [Colocar contenido aquí]

Código 2: *probDobleDelOtro.m*

```
# Funcion que estima la probabilidad de que al tirar dos dados, uno tenga
# el doble de valor que el otro.
# PRE: Debe pasarse por parametro 'n', que es la cantidad de tiradas a simular, utilizando la funcion
#randomDado()
# POST: Se devuelve la probabilidad.
function y = probDobleDelOtro(n)

  tiradasDado1 = randomDado(n);
  tiradasDado2 = randomDado(n);

  salioDobleDelOtro = 0;

  for i= 1:n
    if( tiradasDado1(i) == 2*tiradasDado2(i))
      salioDobleDelOtro = salioDobleDelOtro + 1;
    endif
  endfor

  y = salioDobleDelOtro/n;

endfunction
```

2.4. Enigma final

Con lo hecho y aprendido hasta ahora podemos plantearnos la misma pregunta que se hizo a si mismo el Caballero de Méré en el siglo XVII:

> *"¿Cómo puede ser que cuando apuesto a que voy a obtener al menos un doble as en 24 tiradas de dos dados suelo perder, siendo que suelo ganar cuando apuesto a que voy a obtener al menos un as en 4 tiradas?"*

[Colocar contenido aquí]

Código 3: *probDobleAsEn24Tiradas.m*

```
function y = probDobleAsEn24Tiradas(n)
  cantExperienciasEnLasQueSalioAs = 0;
  for i = 1:n
    salioDobleAs = false;
    d1 = randomDado(24);
    d2 = randomDado(24);
    for j = 1:24
      if(d1(j)== 1 && d2(j) == 1)
        salioDobleAs = true;
      endif
    endfor
    if (salioDobleAs)
      cantExperienciasEnLasQueSalioAs = cantExperienciasEnLasQueSalioAs + 1;
    endif
  endfor
  y = cantExperienciasEnLasQueSalioAs/n;
endfunction
```

Código 4: *probDobleAs.m*

```
function y = probDobleAs(n)
  tiradasDado1= randomDado(n);
  tiradasDado2 = randomDado(n);
  cantDobleAs = 0;
  for i= 1:length(tiradasDado1)
    if( tiradasDado1(i) == 1 && tiradasDado2(i) == 1)
      cantDobleAs = cantDobleAs + 1;
    endif
  endfor
  y = cantDobleAs/length(tiradasDado1);
endfunction
```

3. Actividad principal

Primeramente veamos la situación a analizar. Luego de haber escrito una pieza de software, se desea someterlo a una prueba que garantice su funcionamiento para todas las entradas posibles (finitas). Dado que cada ejecución puede ser afectada por los resultados de las anteriores, se descartan los métodos triviales que consistan en probarlas siguiendo un patrón preestablecido. Atendiendo a lo anterior, se propone elegir al azar una entrada con la cual realizar la ejecución y repetir el procedimiento hasta que todas hayan sido elegidas al menos una vez. Si por ejemplo, existen 3 entradas posibles, representadas por los dígitos 1, 2 y 3, las secuencias que se muestran a continuación serían válidas:

- 3 2 2 2 3 3 2 2 3 2 2 2 3 2 3 1
- 1 1 2 3
- 1 3 2
- 2 1 2 2 2 2 3

Es importante remarcar que en cada paso se elige cualquiera sin importar si ya había sido elegida e independientemente de los valores obtenidos anteriormente. Como se puede apreciar, las longitudes de estas secuencias (y por lo tanto, el tiempo y la memoria que consumen) son aleatorias.

Habiendonos situado en lo recién expuesto, pasaremos a explorar y analizar las propiedades estadísticas de las secuencias generadas.