03/12/2013

Informatica per la Finanza
Informatica per le Scienze Economiche
Informatica e Programmazione per il Turismo

Numeri (pseudo)casuali

- Sono utili in diverse applicazioni
  - In realtà si dovrebbero chiamare pseudocasuali in quanto sono generati con un algoritmo deterministico
    - Consideriamo un generatore congruenziale lineare (LCG):

      $x_{n+1} = (a\ x_n + c) \bmod m \quad n >= 0$

      con a, c ed m scelti opportunamente
    - Si può notare:
      - » Hanno un periodo (essendo il risultato del resto di una divisione)
      - » Scegliendo opportunamente i parametri il periodo è m-1
    - Hanno bisogno di un valore iniziale (seme, seed)
    - Generano numeri distribuiti secondo una v.c. uniforme

Esempio di generatore

- Dati i parametri
  - a=13 c=0 m=31
  - a=7 c=7 m=31
- Provare a generare 33 valori per ciascuna terna usando la funzione a lato

```
function x = GCL (a, c, m, t, s)
 %generatore congruenziale lineare
 % genera un vettore di numeri
 % casuali di dimensione t
 % partendo dal seme s

    x = zeros(1,t);
    x(1) = s;
    for i=1:(t-1)
        x(i+1) = mod(a*x(i)+c , m);
    end
```


Generatori di numeri casuali

- Normalmente i numeri generati sono compresi fra 0 ed 1 (1 escluso) poiché i valori ottenuti si dividono per il periodo del generatore
- E' possibile ottenere dei generatori anche secondo altre v.c.
  - In generale a partire da un generatore per una v.c uniforme è possibile creare un generatore per qualunque altra distribuzione nota la distribuzione della funzione cumulata

Generatori di MatLab

- In MatLab i numeri casuali si generano con la funzione **`rand`** (genera numeri compresi fra 0 e 1,1 escluso) :
  - **`rand(n)`**
    - genera una matrice n x n di
  - **`rand(n,m)`** o **`rand([n,m])`**
    - generano una matrice nxm di numeri casuali
  - **`rand`**
    - genera un numero casuale

Generatori di MatLab

- Il generatore di default di Matlab ha periodo 2^{19936}-1
  - Con ***`help RandStream`*** si possono ottenere ulteriori informazioni
  - Altre funzioni per generare numeri casuali:
    - **`randi(iMax)`**: genera un numero intero fra (0,iMax); non è implementato nelle vecchie versioni di Matlab e in Octave
    - **`randn(n)`**: genera n numeri casuali distribuiti secondo la distribuzione normale

Generatori di Matlab

Inizializzare il generatore

- Per far partire/ripartire il generatore con la stessa sequenza occorre inizializzare il generatore
- Nelle nuove versioni di Matlab si usa la funzione **`rng()`**.
  - **`s=rng`** **`s`** contiene il valore del seme usato
  - **`rng(v)`** inizializza il generatore con **`v`**
  - **`rng('shuffle')`** inizializza il generatore con un valore casuale basato sull'orologio di sistema
  - **`rng('default')`** riporta i generatore al valore di default
  - **`rng(vt,generator)`** forza il generatore con il valore **`vt`** ed utilizza il generatore indicato da **`generator`**:
    - **`'twister'`** Mersenne Twister
    - **`'combRecursive'`** Combined Multiple Recursive
    - **`'multFibonacci'`** Multiplicative Lagged Fibonacci
    - `'v5uniform'`, `'v5normal'`, `'v4'` vecchi generatori Matlab

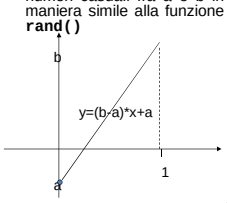
Generatori di Matlab

```
>> rng(8);
>> q=rand(1,100);
>> t=rand(1,100);
>> rng(8);
>> z=rand(1,100);
>> sum(q==t)
  ans = 0
>> sum(q==z)
  ans = 100
```

q e z contengono gli stessi valori

Generazione numeri casuali

- Scrivere una funzione che ha come argomento un intervallo a,b e genera numeri casuali fra a e b in maniera simile alla funzione **`rand()`**




```
function  r = GCLab(a, b, nR, nC)
if (nargin == 4)
  r = (rand(nR,nC) .* (b - a)) + a;
else
  if (nargin == 3)
    r = (rand(nR) .* (b-a)) +a;
  else
    if (nargin ==2)
      r = rand *(b-a) +a;
    else
      error('numero argomenti errati')
    end
  end
end

end
```


Esempio

- Simulare il lancio di una moneta equa (p(T)=p(C)=1/2) per N lanci attraverso una funzione che ha come parametro N e restituisce il numero di teste in N lanci e la frequenza relativa

Esempio

```
function [mediaFa] = moneta(N, M)
   r=[];
   for i=1:M
       r=[r monetaL(N)];
   end

   mediaFa = mean(r);
end

function [fA, fR] = monetaL(N)
   lanci = rand(1,N);
   fA=sum(lanci < 0.5);
   fR = fA / N;
end
```

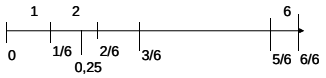

Numeri casuali secondo una distribuzione data

- Supponiamo di voler generare dei numeri casuali che simulino le uscite di un dado equo a sei facce
  - Ogni faccia ha probabilità 1/6 quindi basta dividere l'intervallo di uscita dei numeri della funzione **`rand`** (da 0 ad 1 escluso) in 6 parti:
    - Faccia 1 valori <1/6
    - Faccia 2 valori >=1/6 e <2/6
    - :
    - Faccia 6 valori >=5/6 e < 1

La generazione secondo una distribuzione data si basa sulla funzione **cumulata** e non sulla funzione di densità

Numeri casuali secondo una distribuzione data

```
function num=dado(r,c)
  if (nargin==0)
      r=1; c=1;
  end
  if (nargin==1)
      c=r;
  end
  x = rand(r,c);
  num=zeros(r,c);
  for i=1:r
      for j=1:c
          if x(i,j) < 1/6
              num(i,j) = 1;
          elseif x(i,j) < 2/6
              num(i,j) = 2;
          elseif x(i,j) < 3/6
              num(i,j) = 3;
          elseif x(i,j) < 4/6
              num(i,j) = 4;
          elseif x(i,j) < 5/6
              num(i,j) = 5;
          else
              num(i,j) = 6;
          end
      end
  end
end
```



Numeri casuali secondo una distribuzione data

- La soluzione porta a due osservazioni:
  1. Si è utilizzata la funzione cumulata per risolvere il problema
  2. Usare una catena di if è poco utile
- Un miglioramento prevede
  - uso di un vettore con il limite destro degli intervalli della cumulata
  - usare un'istruzione ciclica che termina quando il valore estratto è minore del limite
  - si può estendere ad intervalli non limitati superiormente ed inferiormente

Numeri casuali secondo una distribuzione data

```
function num=numeri(cumul,r,c)
  if (nargin==1)
      r=1; c=1;
  elseif (nargin==2)
      c=r;
  elseif (nargin ~= 3)
      disp('numero argomenti errato');
      return;
  end
  x = rand(r,c);
  num=zeros(r,c);
  for i=1:r
    for j=1:c
      for k=1:length(cumul)
        if x(i,j) < cumul(k)
          num(i,j) = k;
          break;
        end
      end
    end
  end
end
```

cumul =[1/6 2/6 3/6 4/6 5/6 6/6]

Per creare il vettore delle cumulate si può usare la funzione **cumsum()**:

```
>> x=ones(1,6)*1/6
x =
  0.1667 0.1667 0.1667 0.1667 0.1667 0.1667
>> xCum=cumsum(x)
xCum =
  0.1667 0.3333 0.5000 0.6667 0.8333 1.0000
```

Quindi la funzione si può richiamare con:
numeri(cumsum(ones(1,6)*1/6))

Esempio

- Simulare un generatore di facce di dadi non equo.
  - Il generatore ha in ingresso il vettore con le cumulate
  - verificare anche che i valori di vC sono tutti diversi:
    - usare **diff()**

Esempio

```
function F =generEventi(vC)
  if any(sort(vC) ~= vC)
      error('non  ordinati')
  end
   if ~all(diff(vC))
      error('valori uguali')
  end
  %vC contiene solo valori fra 0 ed 1
  if (sum(vC >= 0) ~= sum(vC <= 1.0))
      error('A valori non fra 0 ed 1');
  else
      if (sum(vC >= 0) ~= 6) %e' un dado!
       error('non e un dado!');
      end
  end

  if vC(6) ~= 1
      error('ultimo valore diverso da 1');
  end

  x = rand;
  %verifico dove sono i valori in vC maggiori di x
  pos=find(vC>x);
  F = vC(pos(1));
end
```

vC=[0.20 0.50 0.75 0.8 0.9 1.0]

Esempio

- Gli eventi A B C D E F hanno una probabilità data dal vettore delle cumulate vC. Fare una estrazione casuale secondo questa cumulata restituendo il nome dell'evento

```
vC=[0.20  0.50  0.75  0.8  0.9  1.0]
     1     2     3     4    5    6
vN=['A'   'B'   'C'   'D'  'E' 'F']
```

Esempio

```
function F =generEventi(vC, vN)
  if any(sort(vC) ~= vC)
      error('non  ordinati')
  end
    if ~all(diff(vC))
      error('valori uguali')
  end
  if (sum(vC >= 0) ~= sum(vC <= 1.0))
      error('A valori non fra 0 ed 1');
  end
  if vC(length(vC)) ~= 1
      error('ultimo valore diverso da 1');
  end
  if length(vC) ~= length(vN)
      error('dimensioni diverse')
  end

  x = rand;
  pos=find(vC>x);
  F = vN(pos(1));
end
```


Generatore VBA

- Si usa la funzione **Rnd(*numero*)**
  - Dove ***numero*** ha i seguenti significati:
    - Minore di zero
      - » Genera lo stesso valore (***numero*** è usato per inizializzare il generatore).
      - » Usato per ripartire la sequenza dallo stesso numero
    - Maggiore di zero o non specificato
      - » Il successivo numero casuale nella sequenza.
    - Uguale a zero
      - » L'ultimo valore generato.
- Si inizializza il generatore con **Randomizie(*numero*)**
  - ***numero***
    - Se presente è usato per inizializzare il generatore
    - Se omesso usa l'orologio di sistema
  - Richiamare **Rnd(-numero)** o **Randomize(numero)** non generala stessa sequenza

Generazione in VBA

```
Option Explicit
Sub testgen()
  Dim i As Integer

  Rnd (-65) 'inizializza
  For i = 1 To 10
    Cells(i, 1) = Rnd()
  Next
  Rnd (-65) 'rinizializza
  For i = 1 To 10
    Cells(i, 2) = Rnd(i * 5)
  Next

  Randomize (65) 'inizializza con Randomize
  For i = 1 To 10
    Cells(i, 3) = Rnd()
  Next

  For i = 1 To 10
    Cells(i, 4) = Rnd(0)
  Next
End Sub
```

#	A	B	C	D
1	0,485921	0,485921	0,93571	0,300203
2	0,322737	0,322737	0,754741	0,300203
3	0,382784	0,382784	0,867654	0,300203
4	0,678655	0,678655	0,506393	0,300203
5	0,800504	0,800504	0,595243	0,300203
6	0,094075	0,094075	0,500996	0,300203
7	0,173604	0,173604	0,704328	0,300203
8	0,242285	0,242285	0,020053	0,300203
9	0,55224	0,55224	0,233219	0,300203
10	0,760987	0,760987	0,300203	0,300203
11				

Stessa sequenza

Sequenza diversa

Test Chi-quadrato

- Il Test Chi-quadrato appartiene alla categoria dei test di verifica di ipotesi e vuole verificare la correttezza di una ipotesi H0
  - Lo applicheremo per verificare se le sequenze generate dalla funzione **rand** sono distribuite secondo una distribuzione uniforme

Test Chi-quadrato

- Il test nel nostro caso procede dividendo tutte le possibili uscite in k=10 classi disgiunte di uguale ampiezza (ovvero nei decili):
  1. Si effettuano N osservazioni
  2. Si calcola la frequenza assoluta (n_s) di oggetti in ogni classe
  3. Si ricava la statistica X^2:

$$X^2 = \sum_{1 \le s \le k} \frac{(n_s - N \cdot p_s)^2}{N \cdot p_s} = \cdots = \frac{1}{N} \sum_{1 \le s \le k} \left(\frac{n_s^2}{p_s} \right) - N$$

  4. Dove p_s indica la probabilità assegnata alla classe dal modello nell'ipotesi H0 (nel nostro caso sempre 1/10)

Test Chi-quadrato

5. Si confronta il valore ottenuto per X^2 con quello di una distribuzione $\chi2(1-\alpha,k-1)$. Se il valore della statistica X^2 è inferiore a quello di $\chi2(1-\alpha,k-1)$ il test si ritiene superato (di solito α=0,05).

Si ripetono i punti da 1 a 5 per più volte

- Per ricavare i valore della cumulata inversa di $\chi2$ si usa la funzione Matlab

chi2inv(1-α,g)

  - Dove **g** indica i gradi di libertà (k-1=9 nel nostro caso)

Test Chi-quadrato

Procediamo per passi nella risoluzione

- Ricavare N=10000 numeri casuali distribuiti secondo una distribuzione uniforme:
  - Inizializzare il generatore

```
rng('shuffle');
```

  - Ricavare le N uscite in un vettore 1 x N

```
x=rand(1,10000);
```


Test Chi-quadrato

- Calcolare la frequenza assoluta degli oggetti in ogni classe

i->

<1/10	<2/10	<3/10	.....	<8/10	<9/10	<1
1	2	3		8	9	10

vettDecili

```
x=[0.15  0.16  0.45  0.36 0.89]
                j->
```

```
freqAss = zeros(1,10);
for j=1:length(x)
     for i=1:10
       if (x(j) < i/10)
          break;
       end
     end
   freqAss(i) = freqAss(i) + 1;
end
```

```
pos =find(x(j) < vettDecili)
freqAss(pos(1)) =freqAss(pos(1))+1;
```

```
freqAss= [0 2 0 1 1 0 0 0 1 0]
```


Test Chi-quadrato

- Calcolare X^2

```
X2 = sum(freqAss.*freqAss)/10000-10000
```

- Verificare se il test è stato superato

```
ok = X2 <  chi2inv(0.95,9);
```

– Il codice è stato riscritto sottoforma di funzione e ripete il test per t volte

Test Chi-quadrato

```
function [totOk, Ok]=testUniforme(N,t)

  if (margin == 1)
      t=15;
  end
  if (margin == 0)
      N=10000;   t=15;
  end

  Ok =zeros(1,t);
  for q=1:t  %le diverse prove
    rng('shuffle'); %inizializza generatore
    x=rand(1,N);    %N numeri casuali
    freqAss = zeros(1,10);
    for j=1:length(x)
      for i=1:10
        if (x(j) < i/10)
          break;
        end
      end
      freqAss(i) = freqAss(i) + 1;
    end

    Ok(q) = (sum(freqAss.*freqAss)/(N*(1/10) - N)) <  chi2inv(0.95,9);
  end

  totOk = sum(Ok);
  disp (['test superato ' , num2str(totOk),  ' volte su '  ,num2str(t)])
end
```


Confronto di velocità

```
x=rand(1,10000);
freqAss1 = zeros(1,10);
freqAss2 = zeros(1,10);

tic
vettDecili = 1/10:1/10:1
for j=1:length(x)
  pos = find( x(j) < vettDecili);
  freqAss1(pos(1)) = freqAss1(pos(1))+1;
end
tvt=toc

tic
for j=1:length(x)
  for i=1:10
    if (x(j) < i/10)
       break;
    end
  end
  freqAss2(i) = freqAss2(i) + 1;
end
tit = toc

sum(freqAss2 == freqAss1)
tvt
tit
```


Test generatore VBA

- Testare il generatore di numeri casuali di VBA
  - Procurarsi i valori
  - Importarli in Matlab (import Wizard)
  - Eseguire il test
    - Modificando il codice precedente

Test generatore VBA

– Procurarsi i valori

```
Option Explicit
Sub genera()
  Dim i As Integer
  Open ThisWorkbook.Path & "\numeriCasuali.txt“ _
      For Output As 1

  For i = 1 To 10000
    Write #1, Rnd
  Next

  Close (1)
End Sub
```


Test generatore VBA

- Generalizzare il codice Matlab in modo che:
  - Accetti in ingresso
    - Uno scalare N o una matrice N
      - Se N è una matrice ogni riga contiene i valori su cui si deve verificare l'ipotesi di uniformità
    - Uno scalare t
      - Indica il numero di prove
      - Esiste/usato solo se N è uno scalare

Test generatore VBA

```
function [totOk, Ok]=testUniformeGener(N,t)
  if (nargin == 0)
      N=10000;   t=15;
  end
  if sum(size(N)) == 2 %scalare
      if (nargin == 1)
          t=15;
      end
      X=zeros(t,N);
      for q=1:t
          rng('shuffle');
          X(q,:)=rand(1,N);
      end
  else
      t = size(N,1);
      X = N;
      N = size(N,2);
  end
  Ok =zeros(1,t);
  for q=1:t  %le diverse prove
    freqAss=prova(X(q,:));
    Ok(q)=(sum(freqAss.*freqAss)/(N*(1/10)-N)) < chiZinv(0.95,9);
  end

  totOk = sum(Ok);
  disp ([‘superato ',num2str(totOk),' volte su ',num2str(t)])
end
```

```
function [freqAss] = prova(seq)
    freqAss = zeros(1,10);
    for j=1:length(seq)
        for i=1:10
            if (seq(j) < i/10)
                break;
            end
        end
        freqAss(i) = freqAss(i) + 1;
    end
end
```


Test generatore VBA

```
Option Explicit
Sub genera()
  Dim i As Integer,_
      j As Integer
  For j = 1 To 20
    Randomize
    For i = 1 To 10000
      Cells(i, j) = Rnd
    Next
  Next
End Sub
```

```
>> q=xlsread('prove.xlsm','Foglio3');
>> size(q)
ans =
        10000          20

>> testUniformeGener(q')
test superato 20 volte su 20
ans =
    20
```