Ðàñïðåäåëåíèå Ïóàññîíà

Îáîçíà÷åíèå	$P(x \mid \lambda)$
Îáëàñòü çíà÷åíèé	x – öåëîå, $0 \le x < \infty$
Ïàðàìåòðû	Ïàðàìåòð ïîëîæåíèÿ $\lambda > 0$
Ïëîòíîñòü (ôóíêöèÿ âåðîÿòíîñòè)	$\lambda^x \exp(-\lambda)/x!$
Ìàòåìàòè÷åñêîå îæèäàíèå	λ
Äèñïåðñèÿ	λ
Ôóíêöèÿ ðàñïðåäåëåíèÿ	$\sum_{i=0}^{x} \frac{\lambda^i e^{-\lambda}}{i!}$

Ñâÿçü ñ äðóãèìè ðàñïðåäåëåíèÿìè

Âåðîÿòíîñòü òîãî, ÷òî ïóàññîíîâà ñëó÷àéíàÿ âåëè÷èíà ζ íå ïðåâîñõîäèò x, ðàâíà âåðîÿòíî-ñòè òîãî, ÷òî ñëó÷àéíàÿ âåëè÷èíà, èìåþùàÿ ðàñïðåäåëåíèå õè-êâàäðàò ñ $2 \times (x+1)$ ñòåïåíÿ-ìè ñâîáîäû, áîëüøå 2λ: $P(x \mid \lambda) = 1 - \chi^2(2\lambda \mid 2(x+1))$. Ïîñêîëüêó õè-êâàäðàò, â ñâîþ î÷åðåäü, ÿâëÿåòñÿ ÷àñòíûì ñëó÷àåì ãàììà-ðàñïðåäåëåíèÿ, ïîëó÷àåì $P(x \mid \lambda) = \Gamma(x+1 \mid \lambda)$. Êîíå÷íî, ê ýòîìó çàêëþ÷åíèþ ìîæíî áûëî ïðèäòè è íåïîñðåäñòâåííî.

Ñóììà n íåçàâèñèìûõ ñëó÷àéíûõ âåëè÷èí, ζ_i, i=1..n, ãäå ζ_i ïîä÷èíÿþòñÿ ðàñïðåäåëåíèþ Ïóàññîíà ñ ïàðàìåòðîì λ_i, èìååò òàêæå ðàñïðåäåëåíèå Ïóàññîíà ñ ïàðàìåòðîì $\lambda = \lambda_i + \ldots + \lambda_n$.

Ðàñïðåäåëåíèå Ïóàññîíà ÿâëÿåòñÿ ïðåäåëüíîé ôîðìîé áèíîìèàëüíîãî ðàñïðåäåëåíèÿ $B(x \mid n, p)$ ïðè $n \to \infty$, $p \to \infty$, $np \to \lambda$.

Ïðè $\lambda > 9$ ðàñïðåäåëåíèå Ïóàññîíà ìîæíî àïïðîêñèìèðîâàòü íîðìàëüíûì ðàñïðåäåëåíèåì ñî ñðåäíèì è äèñïåðñèåé, ðàâíûìè λ.

Ãåíåðàöèÿ ñëó÷àéíûõ ÷èñåë

Ïðîñòîé ñïîñîá ìíå íå èçâåñòåí. Âîò òðóäîåìêèé:

Âû÷èñëÿåì ôóíêöèþ ðàñïðåäåëåíèÿ $P(x|\lambda)$, x=0..N, ãäå N ïðîèçâîëüíî, íî äîñòàòî÷íî âåëèêî (âåëè÷èíà ýòîãî "âåëèêî" çàâèñèò îò âåëè÷èíû λ). Ïîëîæèì $\zeta = x$, åñëè $P(x|\lambda) \le r < P(x+1|\lambda)$, ãäå r ïîä÷èíÿåòñÿ ðàâíîìåðíîìó íà [0,1] ðàñïðåäåëåíèþ.

Ïîñêîëüêó âû÷èñëÿòü ôóíêöèþ ðàñïðåäåëåíèÿ áûâàåò íàêëàäíî, ïðè ìàëûõ λ ìîæíî ïðèìåíÿòü ñëåäóþùèé ìåòîä: $\zeta = x$, åñëè $r_1 \ge e^{-\lambda}$, $r_1 r_2 \ge e^{-\lambda}$, …, $r_1 r_2 \ldots r_x \ge e^{-\lambda}$, $r_1 r_2 \ldots r_x r_{x+1} \ge e^{-\lambda}$.

Âû÷èñëåíèå ôóíêöèè ðàñïðåäåëåíèÿ è åå êâàíòèëåé

Êîíå÷íî, ïðè âû÷èñëåíèè êóìóëÿòèâíîé ôóíêöèè ðàñïðåäåëåíèÿ ñëåäóåò âîñïîëüçîâàòüñÿ óïîìÿíóòîé ñâÿçüþ ïóàññîíîâñêîãî è ãàììà-ðàñïðåäåëåíèÿ (ñì. ôóíêöèþ poissonDF). Ýòîò ñïîñîá çàâåäîìî ëó÷øå íåïîñðåäñòâåííîãî ñóììèðîâàíèÿ óæå ïðè n = 10.

Êàê âñåãäà, êîãäà ìû èìååì äåëî ñ äèñêðåòíîé ôóíêöèåé ðàñïðåäåëåíèÿ, âû÷èñëåíèå êâàíòèëåé íåîäíîçíà÷íî; ïðè ïðîâåðêå ñòàòèñòè÷åñêèõ êðèòåðèåâ ïðåäëàãàåòñÿ ñðàâíèâàòü ñ çàäàííûì ïîðîãîì íàáëþäàåìûå çíà÷èìîñòè.

Âçàìåí ïðåäëàãàåòñÿ ôóíêöèÿ rev_poissonDF, êîòîðàÿ ïî èçâåñòíîé âåðîÿòíîñòè q òîãî, ÷òî ñëó÷àéíàÿ âåëè÷èíà, ïîä÷èíåííàÿ ðàñïðåäåëåíèþ Ïóàññîíà, íå ïðåâîñõîäèò n, íàõîäèò ïàðàìåòð λ ýòîãî ðàñïðåäåëåíèÿ. Äðóãèìè ñëîâàìè, ýòà ôóíêöèÿ ïðèìåíÿåòñÿ äëÿ ðåøåíèÿ ïî λ óðàâíåíèÿ $P(x|\lambda) = q$.

Îíà ïîëåçíà, íàïðèìåð, äëÿ îïðåäåëåíèÿ ëåâîé è ïðàâîé ãðàíèö äîâåðèòåëüíîãî èíòåðâàëà.

Ôàéë poissonDF.h

```
#ifndef __POISSON_H__                // Prevents redefinition

/***************************************************/
/*                Ðàñïðåäåëåíèå Ïóàññîíà           */
/***************************************************/

double poissonDF(double lambda, double n);
/* Âû÷èñëÿåò âåðîÿòíîñòü òîãî, ÷òî ñëó÷àéíàÿ âåëè÷èíà,
 * ïîä÷èíÿþùàÿñÿ ðàñïðåäåëåíèþ Ïóàññîíà ñ ñðåäíèì lambda,
 * íå ïðåâîñõîäèò n.
 */

double rev_poissonDF(double n, double q);
/* Ïî èçâåñòíîé âåðîÿòíîñòè q òîãî, ÷òî ñëó÷àéíàÿ âåëè÷èíà,
 * ïîä÷èíÿþùàÿñÿ ðàñïðåäåëåíèþ Ïóàññîíà, íå ïðåâîñõîäèò n,
 * íàõîäèò ñðåäíåå lambda, äëÿ êîòîðîãî poissonDF(lambda, n)
 * ðàâíî q.
 */

double rightCI_Poisson(double n, double level);
/* Ïî èçâåñòíîìó êîëè÷åñòâó "óñïåõîâ" âû÷èñëÿåòñÿ
 * ïðàâàÿ ãðàíèöà äîâåðèòåëüíîãî èíòåðâàëà ñ
 * äîâåðèòåëüíûì óðîâíåì level.
 */
```

```
double leftCI_Poisson(double n, double level);
/* Ïî èçâåñòíîìó êîëè÷åñòâó "óñïåõîâ" âû÷èñëÿåòñÿ
 * ëåâàÿ ãðàíèöà äîâåðèòåëüíîãî èíòåðâàëà ñ
 * äîâåðèòåëüíûì óðîâíåì level.
 */

#define __POISSON_H__        // To prevent redefinition
#endif                       // Ends "#ifndef __POISSON_H__
```

Ôàéë poissonDF.cpp

```
/*************************************************/
/*                 Ðàñïðåäåëåíèå Ïóàññîíà           */
/*************************************************/
#include <assert.h>

#include "gammaDF.h"

ENTRY double
poissonDF(double lambda, double n)
/* Âû÷èñëÿåò âåðîÿòíîñòü òîãî, ÷òî ñëó÷àéíàÿ âåëè÷èíà,
 * ïîä÷èíÿþùàÿñÿ ðàñïðåäåëåíèþ Ïóàññîíà ñ ñðåäíèì lambda,
 * íå ïðåâîñõîäèò n.
 */
{
   assert(lambda > 0);
   return (n < 0) ? 0 : 1-GammaDF(n+1).value(lambda);
}/*poissonDF*/

ENTRY double
rev_poissonDF(double n, double q)
/* Ïî èçâåñòíîé âåðîÿòíîñòè q òîãî, ÷òî ñëó÷àéíàÿ âåëè÷èíà,
 * ïîä÷èíÿþùàÿñÿ ðàñïðåäåëåíèþ Ïóàññîíà, íå ïðåâîñõîäèò n,
 * íàõîäèò ñðåäíåå lambda, äëÿ êîòîðîãî poissonDF(lambda, n)
 * ðàâíî q.
 */
{
   assert((q >= 0) && (q <= 1));
   return (q == 0) ? 0 : ((q == 1) ? 1 : 1-GammaDF(n+1).inv(q));
}/*rev_poissonDF*/

ENTRY
double rightCI_Poisson(double n, double level)
{
   assert((level >= 0.5) && (level < 1));
   return rev_poissonDF(n, (1+y)/2);
}/*rightCI_Poisson*/

ENTRY
double leftCI_Poisson(double n, double level)
{
assert((level >= 0.5) && (level < 1));
   return rev_poissonDF(n, (1-y)/2);
}/*leftCI_Poisson*/
```