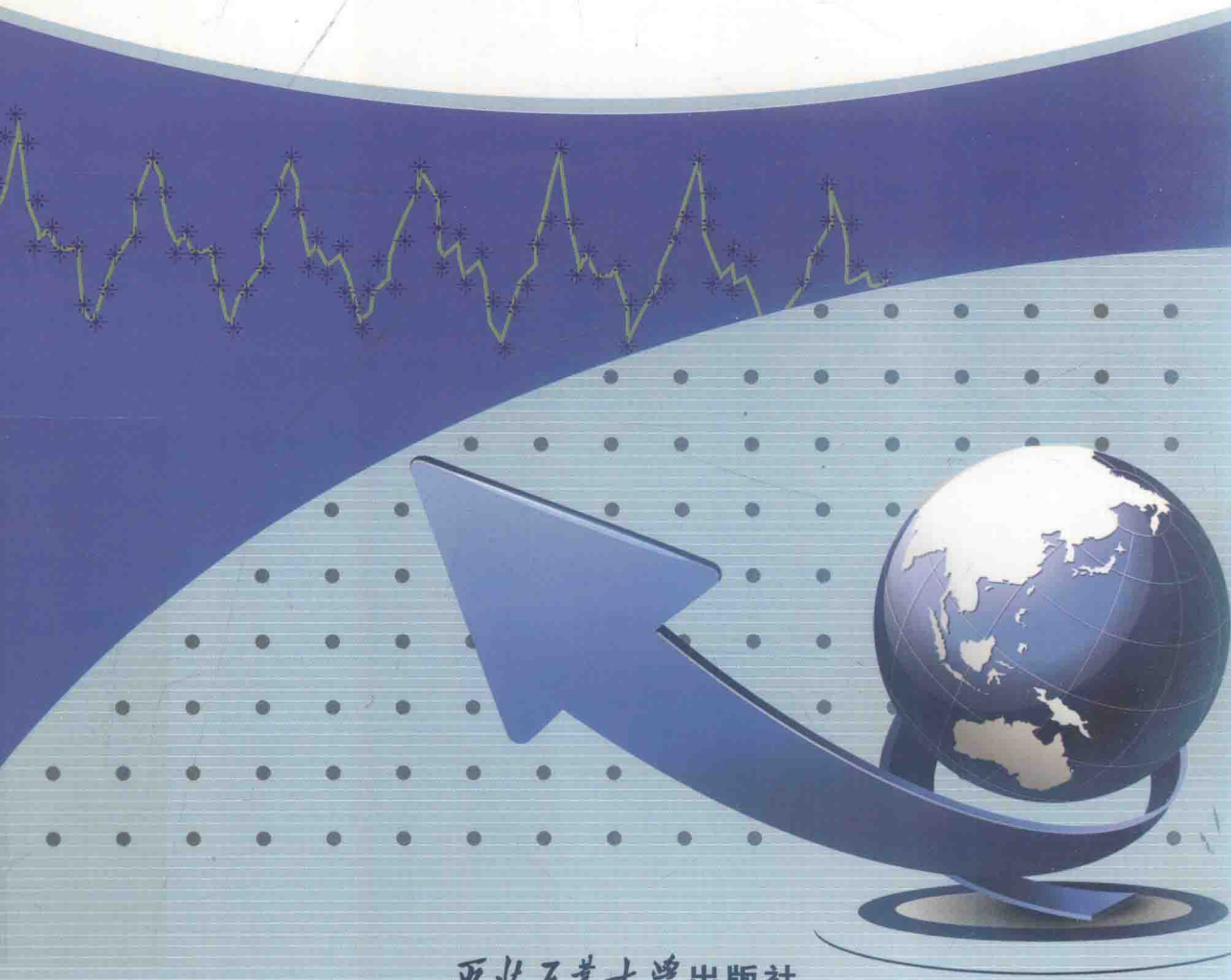
高等学校“十三五”规划教材

统计计算与软件应用

（第2版）

肖华勇◎主编



西北工业大学出版社

高等学校“十三五”规划教材

TONGJI JISUAN YU RUANJIAN YINGYONG

统计计算与软件应用

（第2版）

肖华勇　主编

西北工业大学出版社

西安

【内容简介】 本书是在《统计计算与软件应用》首版的基础上修订而成的。内容包括常用概率分布的分布函数、分位数的算法及程序实现，常用随机变量产生的算法与程序实现，概率中典型问题的计算机模拟实验，参数区间估计的软件实现，以及统计中假设检验、回归分析、方差分析、方差分析应用、聚类分析、判别分析、主成分分析、因子分析、典型相关分析、时间序列分析的基本方法和软件实现等。

本书通过实际案例引导读者利用统计软件来解决实际问题，加深读者对概率知识的理解，提高对统计知识的应用能力，达到利用统计知识，结合统计软件，解决实际问题的能力。

本书可作为高等学校统计专业高年级学生学习统计软件的教材，也可供需要用统计软件解决实际问题的读者阅读、参考。

图书在版编目(CIP)数据

统计计算与软件应用/肖华勇主编. —2 版. —西安：
西北工业大学出版社，2018.3
ISBN 978-7-5612-5884-2

Ⅰ.①统… Ⅱ.①肖… Ⅲ.①统计核算—高等学校—教材
②统计分析—应用软件—高等学校—教材 Ⅳ.①C81

中国版本图书馆 CIP 数据核字(2018)第 041210 号

策划编辑：杨 军
责任编辑：杨 军

出版发行：西北工业大学出版社
通信地址：西安市友谊西路 127 号 邮编：710072
电 话：(029)88493844 88491757
网 址：www.nwpup.com
印 刷 者：兴平市博闻印务有限公司
开 本：787 mm×1 092 mm 1/16
印 张：14.5
字 数：349 千字
版 次：2018 年 3 月第 2 版 2018 年 3 月第 1 次印刷
定 价：46.00 元

第 2 版前言

普通高等院校统计专业的学生到四年级已经学习了概率论、数理统计、时间序列分析、多元统计分析等专业课程。这些课程既有理论，又有实践应用，而对各种随机变量的产生，各种分布函数和分位数的计算，并没有进行介绍，但是这部分内容对实际处理问题却十分重要。本书给出了这些算法，并给出 C 语言实现程序，便于学生学习和使用。对概率中的古典概型、几何概型和大数定律等，以及统计中的区间估计和排队论模型等，只有通过计算机模拟，才能更好地弄清概念和解决问题，同时掌握利用随机试验验证理论计算的方法。本书对这些问题给出了模拟程序与方法。对各种算法只有通过自己编程实现，才能更好地掌握各种算法，深刻理解算法原理。本书给出的程序为读者提供了一种非常好的参考。

时间序列分析与多元统计分析都是应用性极强的学科，除学习有关理论外，需要通过利用专业的统计软件如 SAS，SPSS 进行相当多的实际数据处理，才能掌握专业软件的使用技巧，学会利用专业软件解决实际问题，也才能更好地理解专业书中介绍的各种处理方法，如时间序列中的 AR 模型、ARMA 模型、ARIMA 模型，多元统计中的回归分析、方差分析、主成分分析、因子分析、典型相关分析等。这些方法与理论，只有通过对实际数据的处理，对实际问题的解决，才能获得切身的感受，才能更好地学好本课程，将来在实际工作中也才能真正懂得如何解决问题。本书针对这些统计方法并结合具体实例，通过统计软件进行求解，给出详细操作步骤，同时，对结果给出详细解释，这样使读者更容易上手，对结果能真正弄懂意义。

本书在第 1 版的使用过程中，受到许多读者的喜欢。本次修订，主要增加了一些随机模拟的实例，还增加了以下内容第 4 章参数区间估计的软件实现，6.4 节线性回归的 MATLAB 实现，第 8 章方差分析应用——葡萄酒评价问题，14.5 节时间序列的典型分解模型，14.6 时间序列的灰色模型模型及预测。这样不但增强了实际应用，而且增加了程序实现，更便于读者学习和使用。

本书的特点是读者容易上手，可以帮助读者掌握统计软件和统计方法的使用。本书既可以作为统计实验课教材，也可供需要用统计软件解决实际问题的读者使用和参考，还可以供学习数学建模和参加建模竞赛的学生学习统计的方法和统计软件时阅读和参考。

本书在编写过程中，参阅了相关文献资料。在此，谨向其作者深表谢意。

由于笔者水平有限，书中不足之处在所难免，恳请广大读者指正。

肖华勇

2017 年 11 月

目　录

第 1 章　常用分布函数、分位数算法与程序

在进行统计中的假设检验、区间估计时，经常会涉及各种分布函数和分位数的计算，然而通常教科书中并没有相关算法的介绍与实现。针对这一问题，本章介绍了常用分布函数、分位数的算法，并给出了 C 语言程序和验证结果。

1.1　标准正态分布的分布函数和分位数的计算

1. 标准正态分布函数 $\Phi(x)$ 的计算

因为 $\Phi(x)$ 是对称函数，所以只须给出 $x>0$ 时 $\Phi(x)$ 的计算方法。当 $x<0$ 时，利用 $\Phi(x)=1-\Phi(-x)$ 计算。

$\Phi(x)$ 的连分式计算法：

$$\Phi(x)\approx\begin{cases}\dfrac{1}{2}+\dfrac{f(x)x}{1}+\dfrac{-x^2}{3}+\dfrac{2x^2}{5}+\cdots+\dfrac{(-1)^n nx^2}{(2n+1)}, & 0\leqslant x\leqslant 3\\ 1-\dfrac{f(x)}{x}+\dfrac{1}{x}+\dfrac{2}{x}+\cdots+\dfrac{n}{x}, & x>3\end{cases}$$

其中，$f(x)=\dfrac{1}{\sqrt{2\pi}}e^{-\frac{x^2}{2}}$ 为标准正态分布的密度函数。

采用递推式表达，当 $0\leqslant x\leqslant 3$ 时，令

$$\begin{cases}a_{n+1}=0\\ a_k=\dfrac{(-1)^k kx^2}{(2k+1)+a_{k+1}}, & k=n,n-1,\cdots,1\end{cases}$$

则

$$\Phi(x)=\frac{1}{2}+\frac{f(x)x}{1+a_1}$$

当 $x>3$ 时，令

$$\begin{cases}a_{n+1}=0\\ a_k=\dfrac{k}{x+a_{k+1}}, & k=n,n-1,\cdots,1\end{cases}$$

则

$$\Phi(x)=1-\frac{f(x)}{x+a_1}$$

当 $n=28$ 时，计算精度达到 10^{-12}。

2. 标准正态分布分位数 u_α 的计算

采用 Toda 近似公式：

$$u_\alpha\approx\left(y\sum_{i=0}^{10}b_i y^i\right)^{\frac{1}{2}}$$

其中

$$y=-\ln[4\alpha(1-\alpha)]$$

$b_0=0.157\ 079\ 628\ 8\times10$，　$b_1=0.370\ 698\ 790\ 6\times10^{-1}$

$b_2=-0.836\ 435\ 358\ 9\times10^{-3}$，$b_3=-0.225\ 094\ 717\ 6\times10^{-3}$

$b_4=0.684\ 121\ 829\ 9\times10^{-5}$，　$b_5=0.582\ 423\ 851\ 5\times10^{-5}$

$b_6=-0.104\ 527\ 497\ 0\times10^{-5}$，$b_7=0.836\ 093\ 701\ 7\times10^{-7}$

$b_8=-0.323\ 108\ 127\ 7\times10^{-8}$，$b_9=0.365\ 776\ 303\ 6\times10^{-10}$

$b_{10}=0.693\ 693\ 398\ 2\times10^{-12}$

该公式的最大相对误差为 1.2×10^{-8}。

C语言实现程序：

```
#include <stdlib.h>
#include <stdio.h>
#include <time.h>
#include <math.h>
 #define Pi 3.1415926535
  double GaossUa(double af);
  double GaossPx(double x);
  double GaossFx(double x);
  double GaossFx(double x);
 void main()
 {
  double af;
  double x;
  af=0.05;
printf("标准正态分布上侧分位数 af=%6.3f,u=%9.6f\n",af,GaossUa(af));
  af=0.85;
printf("标准正态分布上侧分位数 af=%6.3f,u=%9.6f\n",af,GaossUa(af));
  x=2;
printf("标准正态分布函数 F(%5.2f)=%9.6f\n",x,GaossFx(x));
  x=-1.5;
printf("标准正态分布函数 F(%5.2f)=%9.6f\n",x,GaossFx(x));
 }
//计算标准正态分布的上侧分位数
//采用 Toda 近似公式计算(1967 年)
// af ---输入的上侧概率分位数
// ua ---返回上侧分位数点
//精度达到 1.2*e-8
  double GaossUa(double af)
 {
 double b[11]={0.1570796288*10,0.3706987906*1.0e-1,
               -0.8364353589*1.0e-3,-0.2250947176*1.0e-3,
               0.6841218299*1.0e-5,0.5824238515*1.0e-5,
               -0.1045274970*1.0e-5,0.8360937017*1.0e-7,
               -0.3231081277*1.0e-8,0.3657763036*1.0e-10,
```

```
                0.6936233982 * 1.0e-12};
double y,ua;
int i;
y=-log(4 * af * (1.0-af));
ua=0.0;
  for(i=0;i<=10;i++)
  ua+=b[i] * pow(y,i);
  ua=sqrt(ua * y); //计算公式
if(af>0.5)  ua=-ua;
return ua;
}
 //标准正态分布的密度函数
 double GaossPx(double x)
{
 double f;
 f=1.0/sqrt(2.0 * Pi) * exp(-x * x/2.0);
 return f;
}
//计算标准正态分布的分布函数
// x- - - -输入 x
// prob- - - -返回 P{X<=x}分布函数值
//对 n=28,精度达到 1.0e-12
 double GaossFx(double x)
{
  double prob,t,temp;
  int i,n,symbol;
    temp=x;
  if(x<0)
     x=-x;
    n=28;  //连分式展开的阶数
    if(x>=0&&x<=3.0)
                  {  //当 0<=x<=3.0 时
      t=0.0;
                 for(i=n;i>=1;i- -)
                 {
                  if(i%2==1) symbol=-1;
                   else symbol=1;
                   t=symbol * i * x * x/(2.0 * i+1.0+t);
                 }
                 prob=0.5+GaossPx(x) * x/(1.0+t);
                  }
 else if(x>3.0)
 {  //当 x>3 时
```

```
                t=0.0;
            for(i=n;i>=1;i--)
        t=1.0*i/(x+t);
            prob=1-GaossPx(x)/(x+t);
    }
    x=temp;
    if(x<0)
            prob=1.0-prob;
     return prob;
}
```

该程序测试结果为

```
标准正态分布上侧分位数 af= 0.050,u= 1.644854
标准正态分布上侧分位数 af= 0.850,u=-1.036433
标准正态分布函数 F( 2.00)= 0.977250
标准正态分布函数 F(-1.50)= 0.066807
```

1.2 Beta分布、T分布、F分布和二项分布的分布函数

T分布、F分布、二项分布等分布的分布函数和分位数的计算都可以利用Beta分布给出。因此这里给出Beta分布的分布函数 $I_x(a,b)$ 和分位数 $\beta_a(a,b)$ 的算法。

1. 分布函数计算

(1)Beta分布的分布函数 $I_x(a,b)$ 的递推算法。Beta分布的分布函数 $I_x(a,b)$ 有以下递推公式：

$$\begin{cases} I_x(a+1,b)=I_x(a,b)-\dfrac{1}{a}U_x(a,b) \\ I_x(a,b+1)=I_x(a,b)+\dfrac{1}{b}U_x(a,b) \\ U_x(a+1,b)=\dfrac{a+b}{a}xU_x(a,b) \\ U_x(a,b+1)=\dfrac{a+b}{b}(1-x)U_x(a,b) \end{cases}$$

其中

$$U_x(a,b)=\frac{1}{B(a,b)}x^a(1-x)^b$$

利用Beta分布的分布函数计算 T 分布、F 分布、二项分布时，参数 a,b 的值或者是正整数，或者是 $\frac{1}{2}$ 的倍数，故这里只考虑参数是正整数或者是 $\frac{1}{2}$ 的倍数的情况下 $I_x(a,b)$ 的计算问题。这时递推公式的初值的选取只有以下4种情况：

1) 当 $a=\frac{1}{2},b=\frac{1}{2}$ 时，有

$$U_x(\frac{1}{2},\frac{1}{2})=\frac{1}{\pi}\sqrt{x(1-x)}$$

$$I_x(\frac{1}{2},\frac{1}{2})=1-\frac{2}{\pi}\arctan\sqrt{\frac{1-x}{x}}$$

2) 当 $a=\frac{1}{2}, b=1$ 时，有

$$U_x(\frac{1}{2},1)=\frac{1}{2}\sqrt{x}(1-x), \quad I_x(\frac{1}{2},1)=\sqrt{x}$$

3) 当 $a=1, b=\frac{1}{2}$ 时，有

$$U_x(1,\frac{1}{2})=\frac{1}{2}x\sqrt{1-x}, \quad I_x(1,\frac{1}{2})=1-\sqrt{1-x}$$

4) 当 $a=1, b=1$ 时，有

$$U_x(1,1)=x(1-x), \quad I_x(1,1)=x$$

(2)T 分布的分布函数计算：

$$T(t \mid n)=\begin{cases}1-\frac{1}{2}I_x(\frac{n}{2},\frac{1}{2}), & t>0\\ \frac{1}{2}I_x(\frac{n}{2},\frac{1}{2}), & t\leqslant 0\end{cases}$$

其中

$$x=\frac{n}{n+t^2}$$

(3)F 分布的分布函数计算：

$$F(x \mid m,n)=I_y(\frac{m}{2},\frac{n}{2})$$

其中

$$y=\frac{mx}{n+mx}$$

(4) 二项分布的分布函数计算。

1) 直接计算法。由二项分布的分布函数的定义计算，有

$$B(x \mid n,p)=\begin{cases}0, & x<0\\ \sum\limits_{k=0}^{[x]}\binom{n}{k}p^k(1-p)^{n-k}, & 0\leqslant x<n\\ 1, & x\geqslant n\end{cases}$$

2) 利用 Beta 分布的分布函数 $I_x(a,b)$ 计算，有

$$B(x \mid n,p)=I_{1-p}(n-[x],[x]+1), \quad 0\leqslant x<n$$

2. 分位数计算

(1)Beta 分布的分位数 $\beta_\alpha(a,b)$ 的求法。方程迭代求根的二分法很适用于求 Beta 分布的分位数。因 $\beta_\alpha(a,b)\in(0,1)$，故(0,1) 就是方程 $f(x)=I_x(a,b)-(1-\alpha)=0$ 的有根区间。把有根区间逐次二等分，最终总可以得出所要求精度的 α 分位数的近似值。此算法简单，且不必给出初始值。

(2)T 分布分位数计算。由 T 分布的分布函数与 $I_x(a,b)$ 的关系式可知：

1) 当 $0<\alpha<\frac{1}{2}$ 时，$t_\alpha>0$，且 t_α 满足：

$$T(t_\alpha \mid n)=1-\frac{1}{2}I_x(\frac{n}{2},\frac{1}{2})=1-\alpha$$

其中

$$x=\frac{n}{n+t_\alpha^2}$$

由此得 $I_x(\frac{n}{2},\frac{1}{2})$，则

$$\beta_{1-2\alpha}\left(\frac{n}{2},\frac{1}{2}\right)=\frac{n}{n+t_p^2}$$

$$t_\alpha(n)=\sqrt{\frac{n}{\beta_{1-2\alpha}\left(\frac{n}{2},\frac{1}{2}\right)}-n}$$

2) 当 $\alpha>\frac{1}{2}$ 时，$t_\alpha<0$，可由 $t_\alpha(n)=-t_{1-\alpha}(n)$ 计算。

(3)F 分布分位数计算。F 分布的 α 分位数记为 $F_\alpha(m,n)$，满足：$F(F_\alpha \mid m,n)=1-\alpha$。由 F 分布函数与 Beta 分布函数的关系，可知 F_α 满足：

$$F(F_\alpha \mid m,n)=I_y\left(\frac{m}{2},\frac{n}{2}\right)=1-\alpha$$

其中

$$y=\frac{mF_\alpha}{n+mF_\alpha}$$

则有

$$\frac{mF_\alpha}{n+mF_\alpha}=\beta_\alpha\left(\frac{m}{2},\frac{n}{2}\right)$$

故

$$F_\alpha(m,n)=\frac{n\beta_\alpha\left(\frac{m}{2},\frac{n}{2}\right)}{m\left(1-\beta_\alpha\left(\frac{m}{2},\frac{n}{2}\right)\right)}$$

C 语言实现程序：

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <time.h>
 #define Pi 3.1415926535
 #define MaxTime   500    //定义最大迭代步数
 #define eps   1.0e-10     //定义精度

double BetaFx(double x,double a,double  b);
double TdistFx(double x,int Freedom);
  double FdistFx(double x,int Freedom_m,int Freedom_n);
  double BetaUa(double af,double a,double b);
  double TdistUa(double af,int Freedom);
  double FdistUa(double af,int Freedom_m,int Freedom_n);
 double BinominalFx(double x,double p,int n);
void main()
{
 double x,p,af,F;
 int m,n;
 m=10;
 n=15;
 x=2.5;
```

```
  F=FdistFx(x,m,n);
  printf("F 分布 F(%4.2f,%2d,%2d)=%6.4f\n",x,m,n,F);
    x=1.4;
    n=25;
  F=TdistFx(x,n);
  printf("T 分布 T(%4.2f,%3d)=%6.4f\n",x,n,F);
   p=0.8;
   n=120;
   x=95;
 F=BinominalFx(x,p,n);
  printf("二项分布 B(%4.2f,%4.2f,%3d)=%6.4f\n",x,p,n,F);
   m=10; n=15;
   af=0.05;
   x=FdistUa(af,m,n);
  printf("F 分布上侧分位数 F(%4.2f,%2d,%2d)=%6.4f\n",af,m,n,x);
   n=15;
   af=0.05;
   x=TdistUa(af,n);
  printf("T 分布上侧分位数 T(%4.2f,%2d)=%6.4f\n",af,n,x);
}
 double BetaFx(double x,double a,double   b)
  {
    int m,n;
            double I,U;
            double ta,tb;
            m=(int)(2*a);
            n=(int)(2*b);
    //以下给定初始值
            if(m%2==1&&n%2==1)
            {
                ta=0.5;
                tb=0.5;
      U=sqrt(x*(1.0-x))/Pi;
             I=1.0-2.0/Pi*atan(sqrt((1.0-x)/x));
            }
             else if(m%2==1&&n%2==0)
             {
                 ta=0.5;
                 tb=1;
        U=0.5*sqrt(x)*(1.0-x);
              I=sqrt(x);
             }
             else if(m%2==0&&n%2==1)
```

```
            {
                ta=1;
                tb=0.5;
        U=0.5*x*sqrt(1.0-x);
                I=1.0-sqrt(1.0-x);
            }
        else if(m%2==0&&n%2==0)
            {
                ta=1;
                tb=1;
                U=x*(1.0-x);
                I=x;
            }
            while(ta<a)
            {
        I=I-U/ta;
            U=(ta+tb)/ta*x*U;
            ta+=1;
            }
            while(tb<b)
            {
            I=I+U/tb;
            U=(ta+tb)/tb*(1.0-x)*U;
            tb+=1;
            }
    return I;
    }
    //计算 T 分布 t(n)在 x 处分布函数值
    //Freedom    ------自由度
     double TdistFx(double x,int Freedom)
     {
            double t,prob;
            t=Freedom/(Freedom+x*x);
             if(x>0)
            prob=1.0-0.5*BetaFx(t,Freedom/2.0,0.5);
             else
             prob=0.5*BetaFx(t,Freedom/2.0,0.5);
             return prob;
     }
    //计算 F 分布 F(m,n)在 x 处分布函数值
    //Freedom_m    -----第一自由度
    //Freedom_n    -----第二自由度
```

```
double FdistFx(double x,int Freedom_m,int Freedom_n)
{
        double y,prob;
        if(x<=0) return 0.0;
         y=Freedom_m*x/(Freedom_n+Freedom_m*x);
         prob=BetaFx(y,Freedom_m/2.0,Freedom_n/2.0);
         return prob;
}
//计算二项分布的分布函数
//p- - - - 二项分布参数
//n - - - - 二项分布
//返回 P{X<=x}
 double BinominalFx(double x,double p,int n)
{
 double prob;
 if(x<0) prob=0.0;
 else if(x>=n) prob=1.0;
 else   prob=BetaFx(1.0-p,n-int(x),int(x)+1);
 return prob;
}
//计算 Beta 分布参数为 a,b(要求为 1/2 的整数倍),上侧分位概率为 af 的上侧分位数
//a 为第一参数
//b 为第二参数
//af - - - -上侧分位概率
//返回上侧分位数

  double BetaUa(double af,double a,double b)
  {
        int times=0;
        double x1,x2,xn;
        double f1,f2,fn,ua;
        x1=0.0;
        x2=1.0;
        f1=BetaFx(x1,a,b)-(1.0-af);
        f2=BetaFx(x2,a,b)-(1.0-af);
         while(fabs((x2-x1)/2.0)>eps)
         {
        xn=(x1+x2)/2.0;
        fn=BetaFx(xn,a,b)-(1.0-af);
        if(f1*fn<0)          x2=xn;
        else if(fn*f2<0)             x1=xn;
        f1=BetaFx(x1,a,b)-(1.0-af);
        f2=BetaFx(x2,a,b)-(1.0-af);
```

```
            times++;
            if(times>MaxTime) break;
             }
 //   printf("times=%5d\n",times);
   ua=xn;
          return ua;

}
//计算 T 分布 t(n)的上侧分位数
//Freedom   ------自由度
//af ---输入的上侧概率分位数
//返回上侧分位数 ua
double TdistUa(double af,int Freedom)
{
            double ua,tbp;
            if(af<=0.5)
            {
            tbp=BetaUa(1-2*af,Freedom/2.0,0.5);
            ua=sqrt(Freedom/tbp-Freedom);
            }
            else if(af>0.5)
            {
   tbp=BetaUa(1-2*(1.0-af),Freedom/2.0,0.5);
            ua=-sqrt(Freedom/tbp-Freedom);
            }
            return ua;
}
//计算 F 分布 F(m,n)的上侧分位数
//Freedom_m   -------第一自由度
//Freedom_n   -------第二自由度
//af ---输入的上侧概率分位数
//返回上侧分位数 ua
double FdistUa(double af,int Freedom_m,int Freedom_n)
{
            double ua,tbp;
            tbp=BetaUa(af,Freedom_m/2.0,Freedom_n/2.0);
            ua=Freedom_n*tbp/(Freedom_m*(1.0-tbp));
            return ua;
}
```

程序测试结果：

F 分布 F(2.50,10,15)=0.9469

T 分布 T(1.40, 25)=0.9131

二项分布 B(95.00,0.80,120)=0.4457