

Statistical Distributions in JAVA and Internet

Miloslav Nosal and Eric M. Nosal

*Department of Mathematics and Statistics and Department of Computer Science,
University of Calgary, 2500 University Drive, Calgary, Alberta, Canada, T2N 1N4
and STATSCON, Statistical Consulting
nosal@ucalgary.ca*

Abstract. *STATISTICAL RESOURCE CENTER (SRC)* is a collection of fully interactive, animated, graphical JAVA applets, developed for teaching basic statistical methods. At this time, in addition to regression and probability applets, its main contribution is 12 applets exploring fundamental statistical distributions. The SRC center is in a development stage and its current Internet URLs are

<http://www.acs.ucalgary.ca/~nosal/src> and <http://www.acs.ucalgary.ca/~enosal/src>

SRC statistical distribution applets allow a graphical interactive exploration of the shape and location of the Gaussian, Student, Chi-squared and Fisher distributions. In addition, these applets also allow remote computations of the distribution critical points and tail probabilities over the Internet. All the numerical computations are supplemented with interactive animated graphical presentations, which demonstrate the meaning of the computed values.

1. Introduction

STATISTICAL RESOURCE CENTER (SRC) has been developed using JAVA 2 in order to provide easy interactive Internet access as well as complete portability and computer platform independence. JAVA 2 has many classes of objects, interfaces and packages facilitating GUI building, graphics and animations, runtime event handling (drag and drop), multimedia including audio, and video, communications and networking, access to databases, multi-language facilities, network security etc. Unfortunately, JAVA language was not developed to perform numerical calculations. The original JAVA standard contains neither numerical functions, which would allow computations associated with fundamental statistical distributions, nor special mathematical functions such as Gamma and Beta functions. This fact makes distributed interactive computation over the Internet using JAVA rather difficult.

JAVA 2 contains SHORT, INTEGER, FLOAT and DOUBLE data types allowing simple arithmetic operations and elementary functions (trigonometric, exponential, and logarithmic). Constants include π , e , $-\infty$ and $+\infty$. *Media* package includes some “hidden” vector and matrix operations disguised as 2D and 3D image transformations needed for graphics development and animation. There are no standard libraries with advanced mathematical support available. *VISUAL NUMERICS* (VN), corporation providing International Mathematical and Statistical Libraries (IMSL) for Fortran, C etc. has developed *JAVA Numerical Library* (JNL). JNL is very incomplete: e.g. incomplete Gamma, Beta and other functions are missing. JNL has serious numerical problems, first discovered during the present development of the SRC and formally recognized by VN. VN has made an executive decision not to support further development JNL in the near future. However, numerical evaluation of fundamental statistical distribution functions and their inverses requires the above mentioned special functions. For this purpose we have used special numerical algorithms for computations of special mathematical functions and

implemented them in JAVA. These JAVA implemented algorithms were then used to develop the SRC. Some of these algorithms are discussed in the next section.

2. Special Numerical Algorithms

A majority of numerical procedures used in the SRC uses standard JAVA 2 functions together with JNL functions developed by VN. The Section 3 contains details of all JAVA algorithms together with detailed listing of all used functions and their sources. It can be seen there that the Normal, Student and Fisher distributions can be quite easily evaluated using standard JAVA functions together with JNL. However, the Chi-Squared distribution critical points and tail probabilities required development of special numerical algorithms. These have been coded in JAVA, and form a part of a new library, called StatsconLib. Details of this library are given in Section 4 below. Numerical details of the corresponding numerical algorithms are given in Section 2 below.

2.1 Algorithm for Computation of Chi-Squared Tail Probabilities

Computation of the Chi-Squared Cumulative Distribution Function (CDF) and corresponding tail probabilities requires the use of the Incomplete Gamma function. In this section, we follow methods and notation of Press et al. (1989), Section 6.2. Chi-Squared CDF with ν degrees of freedom is defined as

$$F(\chi^2, \nu) = P\left(\frac{\nu}{2}, \frac{\chi^2}{2}\right)$$

where $P(a, x)$ is the Incomplete Gamma Function defined by the following formula

$$P(a, x) \equiv \frac{\gamma(a, x)}{\Gamma(a)} \equiv \frac{1}{\Gamma(a)} \int_0^x e^{-t} t^{a-1} dt; \quad (a > 0)$$

where $\Gamma(a)$ is defined as

$$\Gamma(a) = \int_0^{\infty} e^{-t} t^{a-1} dt$$

The function $\gamma(a, x)$ has an infinite series development given as follows:

$$\gamma(a, x) = e^{-x} x^a \sum_{n=0}^{\infty} \frac{\Gamma(a)}{\Gamma(a+1+n)} x^n \quad (1)$$

Sometimes it is more convenient to compute the function $\Gamma(a, x)$, defined as follows:

$$\Gamma(a, x) \equiv \int_x^{\infty} e^{-t} t^{a-1} dt; \quad (a > 0)$$

This function has a continued fraction development as follows:

$$\Gamma(a, x) = e^{-x} x^a \left(\frac{1}{x+1} \frac{1-a}{1+} \frac{1}{x+} \frac{2-a}{x+} \frac{2}{x+} \dots \right); \quad (x > 0) \quad (2)$$

It is the case that (1) converges fast for $x < a + 1$, while (2) converges fast for $x > a + 1$. Thus (1) and (2) together allow for fast and efficient computation of the Incomplete Gamma Function. This algorithm has been used in the StatsconLib, described in Section 4 below.

2.2 Algorithm for Computation of Chi-Squared Critical Points

Computation of critical points of the Chi-Squared distribution requires an inversion of its CDF defined in section 2.1 above, which in turn requires a numerical inversion of the Incomplete Gamma Function. Furthermore, a corresponding numerical algorithm must be accurate, fast, but simple in order to run in real time over the Internet. After extensive testing, a two step algorithm was used in the StatsconLib, described in Sections 3 and 4 below.

Step 1: *Wilson – Hilferty* approximation (Kennedy (1980), p. 118, Wilson and Hilferty (1931)) is given as follows. For a given value $0 < \alpha < 1$, let z_α be a critical point of the Normal distribution, i.e.

$$\frac{1}{\sqrt{2\pi}} \int_{-\infty}^{z_\alpha} e^{-\frac{x^2}{2}} dx = \alpha$$

For given degrees of freedom ν , Wilson – Hilferty approximation of the corresponding Chi-Squared critical point is given as

$$\chi_{\alpha,\nu}^2 \cong W(\alpha,\nu) = \nu \left\{ 1 - \frac{2}{9\nu} + z_\alpha \sqrt{\frac{2}{9\nu}} \right\}^3$$

Step 2: *Gentleman – Jenkins* refinement (Thisted (1988), p.322, Gentleman (1968) is given as

$$\chi_{\alpha,\nu}^2 \cong W(2\alpha - F_{\chi^2}(W(\alpha,\nu)))$$

This 2-step algorithm is very efficient, fast and quite accurate. For practically all useful values $.05 \leq \alpha \leq .95$, the relative accuracy is better than 0.02% for $\nu > 5$ and better than 0.1% for $\nu > 3$. For $\nu \leq 3$ the absolute accuracy is better than 0.0015.

3. JAVA Algorithms used in the SRC

Normal Distribution:

Probability Density Function (PDF): Used the formula of the normal distribution PDF.

```
denom = stdev * Math.sqrt(2*3.1415926539);
y      = 1/denom * Math.exp(-0.5) * Math.pow(((x-mean)/stdev), 2);
```

where $y = P(X=x)$ given a standard deviation (stdev) and a mean (mean).

Critical Points: Used Visual Numeric's JNL.

```
n-crit = mean - stdev * Statistics.inverseNormalCdf(tailProb);
```

Where n-crit is the critical point and area is the entered tail probability and stdev, mean as above.

Tail Probability: Used Visual Numeric's JNL.

```
area = Statistics.normalCdf(mean-(n-crit*stdev));
```

Where area is the tail probability and n-crit, mean, and stdev are as above.

Student-T Distribution:

Probability Density Function: Used the formula of the Student-T distribution and the gamma function from JNL.

```
y = Sfun.gamma((df1+1)/2)/  
    (Sfun.gamma(df1/2)*Sfun.gamma(0.5)*Math.sqrt(df1));
```

where $y = P(X=x)$ given degrees of freedom 1 (df1).

Critical Points: Used Visual Numeric's JNL.

```
t-crit = Statistics.inverseTCdf(1-area, df1);
```

Where t-crit is the critical point and area is the entered tail probability and df1 is as above.

Tail Probability: Used Visual Numeric's JNL.

```
area = Statistics.tCdf(t-crit, df1);
```

Where area is the tail probability and t-crit, df1 are as above.

Tail Probability: Used Visual Numeric's JNL.

```
area = Statistics.normalCdf(mean-(n-crit*stdev));
```

Where area is the tail probability and n-crit, mean, and stdev are as above.

Fisher Distribution:

Probability Density Function: Used the formula of the Fisher distribution PDF and the gamma function from JNL.

```
numer = (Sfun.gamma((df1+df2)/2) * Math.pow(df1/df2, df1/2) *  
        Math.pow(x, (df1/2)-1));
```

```
denom = (Sfun.gamma(df1/2) * Sfun.gamma(df2/2) *  
        (Math.pow((df1/df2)*x+1, (df1+df2)/2)));
```

```
y = numer/denom;
```

where $y = P(X=x)$ given degrees of freedom 1 (df1) and 2 (df2).

Critical Points: Used Visual Numeric's JNL.

```
f-crit = Statistics.inverseFCdf(1-area, df1, df2);
```

Where f-crit is the critical point and area is the entered tail probability and df1, df2 as above.

Tail Probability: Used Visual Numeric's JNL.

```
area = Statistics.FCdf(f-crit, df1, df2);
```

Where area is the tail probability and f-crit, df1, df2 are as above.

Chi-Squared Distribution: (See Section 4 below)

Probability Density Function: Used the fomula of the Chi-Squared PDF with gamma function from JNL.

```
numer = (Math.pow(x, (df1/2)-1)*Math.exp(-x/2));
denom = (Math.pow(2, df1/2)*Sfun.gamma(df1/2));
y = numer/denom;
```

where $y = P(X=x)$ given a degree of freedom (df1).

Critical Points: Used StatsconLib for Chi-Squared CDF and JNL for inverse Normal CDF.

Step 1:

```
private double W(double alpha, double df1) {
    double s;
    s = Statistics.inverseNormalCdf(1-alpha);

    return df1*Math.pow((1(2/(9*df1))+s*Math.sqrt(2/(9*df1))), 3);
}
```

...
...

Step 2:

```
double alpha = chi-crit;
double Fx2 = 1-stats.chiSquaredCDF(df1, W(alpha, df1));
area = W(2*alpha - Fx2, df1);
```

Where chi-crit is the critical point and area is the entered tail probability and df1 is as above.

Tail Probability: Used StatsconLib.

```
area = 1 - stats.chiSquaredCDF(df1, chi-crit);
```

Where area is the tail probability and chi-crit, df1 are as above.

4. JAVA Code for StatsconLib.java:

```
import java.lang.Math;

public class StatsconLib {

    public boolean error;
    public String whatError;

    /**
     * Function gammaln: returns the value of ln(gamma(xx)) for xx > 0
     */
    public double gammaln(double xx) {
        double x, tmp, ser;
        double cof[] = {76.18009173, -86.50532033, 24.01409822, -
1.231739516, 0.120858003e-2, -0.536382e-5};
        int j;

        x = xx - 1.0;
        tmp = x + 5.5;
        tmp -= (x+0.5)*Math.log(tmp);
        ser = 1.0;
```

```

    for(j = 0; j <= 5; j++) {
        x += 1.0;
        ser+= cof[j]/x;
    }
    return -tmp+Math.log(2.50662827465*ser);
}

//*****
//Function gammaIncomplete: calculates the incomplete gamma fuction.*
//                               P(a,x)                               *
//*****
    public double gammaIncomplete(double a, double x) {
        double gamser, gammcf, gln;
        if(x < 0.0 || a <= 0.0) {
            error = true;
            whatError = "Invalid use of arguments in function
gammaIncpomplete";
            return -1;
        }

        if(x < (a+1.0)) {
            {
                int n;
                double sum, del, ap;

//Compute infinite series (6.2.5) from Numerical Recipes in C,
//page 171

                gln = gammaln(a);
                if(x <= 0.0) {
                    if(x < 0.0) {
                        error = true;
                        whatError = "x can not be less than 0";
                        return -1;
                    }
                    gamser = 0.0;
                    return gamser;
                }
                else {
                    ap = a;
                    del = sum = 1.0/a;
                    for(n = 1; n <= 100; n++) {
                        ap += 1.0;
                        del *= x/ap;
                        sum += del;
                        if( Math.abs(del) < Math.abs(sum)*3.0e-7) {
                            gamser = sum*Math.exp(-x+a*Math.log(x)-gln);
                            return gamser;
                        }
                    }
                }
                error = true;
                whatError = "a too large, 100 iterations is too small
for series sum";
                return -1;
            }
        }
    }

    else {
        int n;
        double gold, g, fac, b1, b0, anf, ana, an ,a1, a0;

```

```

gold = b0 = 0.0;
fac = b1 = a0 = 1.0;

//Compute the continued fraction (6.2.6) from Numerical Recipes in C,
//page 171

gln = gammaln(a);
a1 = x;
for(n = 1; n <= 100; n++) {
    an = (double) n;
    ana = an-a;
    a0=(a1+a0*ana)*fac;
    b0=(b1+b0*ana)*fac;
    anf=an*fac;
    a1=x*a0+anf*a1;
    b1=x*b0+anf*b1;
    if(a1!=0.0) {
        fac = 1.0/a1;
        g=b1*fac;
        if(Math.abs((g-gold)/g) < 3.0e-7) {
            gammcf = Math.exp(-x+a*Math.log(x)-gln) * g;
            return 1.0-gammcf;
        }
        gold = g;
    }
}
error = true;
whatError = "a too large, 100 iterations for continuous
fraction";
return -1;
}
}

public double chiSquaredCDF(double df, double chi-crit) {
    return gammaIncomplete(df/2.0, chi-crit/2.0);
}
}

```

gammaln(double xx) computes $\ln(\text{gamma}(xx))$ for $xx > 0$. From this point, $\text{gamma}(xx)$ is calculated using either continued fractions of the series expansion of $\text{gamma}(xx)$. For values of $x < a+1$, the series expansion was used, otherwise continued fractions were used. For more details, see Press (1989), pp 168-177.

5. Conclusions

Statistical Resource Center has been used by hundreds of students at the University of Calgary registered in various statistical courses. General student response has been very positive. Students appreciated mainly connection between the numerical computing and graphical presentation of numerical concepts as well as possibility to interactively explore various ideas.

6. References

- Gentleman, W.M. and Jenkins, M.A. (1968): An approximation for Student's t-distribution, *Biometrika* 55, pp. 571-572
- Kennedy, W.J. and Gentle, J.E. (1980): *Statistical Computing*, Marcell Dekker, Inc., New York and Basel, p. 118

Press, W.H., Flannery, B. P., Teukolsky, S.A., Vetterling, W.T., (1989): Numerical Recipes in C, Cambridge University Press, Cambridge, pp. 171 – 177

Thisted R, . A. (1988): Elements of Statistical Computing, Chapman and Hall, New York and London

Wilson, E.B. and Hilferty, M.M. (1931): The Distribution of Chi-Square, Proc. Acad. Nat. Sci., 17, pp. 684 - 688