Article

probFunCal2: A web-based calculator for calculating probability distribution functions as binomial, Poisson, exponential, gamma, beta, uniform, Weibull, lognormal, Cauchy, geometric, and negative binomial distributions

WenJun Zhang
School of Life Sciences, Sun Yat-sen University, Guangzhou 510275, China
E-mail: zhwj@mail.sysu.edu.cn, wjzhang@iaees.org

Received 8 Augustl 2025; *Accepted 26 September 2025*; *Published online 1 October 2025*; *Published 1 December 2026*

Abstract
In this study, I developed probFunCal2, an advanced web-based calculator for probability distribution functions that complements the original probFunCal by providing comprehensive support for additional probability distributions. The calculator includes binomial, Poisson, exponential, gamma, beta, uniform, Weibull, lognormal, Cauchy, geometric, and negative binomial distributions. For each distribution, probFunCal2 calculates probability density/mass functions (PDF/PMF), cumulative distribution functions (CDF), and inverse cumulative distribution functions (quantiles). The implementation uses sophisticated numerical methods including Stirling's approximation for factorials, continued fraction methods for regularized incomplete beta and gamma functions, and various quantile approximations. The calculator is fully browser-based, supporting both online and offline usage across multiple platforms and devices. This tool provides researchers, students, and practitioners with a robust, accessible platform for statistical computations essential for probability theory, statistical inference, and data analysis.

Keywords calculator; web application; JavaScript; probability distribution; binomial distribution; Poisson distribution; exponential distribution; gamma distribution; beta distribution; uniform distribution; Weibull distribution; lognormal distribution; Cauchy distribution; geometric distribution; negative binomial distribution.

Selforganizology
ISSN 2410-0080
URL: http://www.iaees.org/publications/journals/selforganizology/online-version.asp
RSS: http://www.iaees.org/publications/journals/selforganizology /rss.xml
E-mail: selforganizology@iaees.org
Editor-in-Chief: WenJun Zhang
Publisher: International Academy of Ecology and Environmental Sciences

1 Introduction

Probability distributions form the foundation of statistical theory and practice, enabling the modeling of random phenomena across diverse scientific disciplines. In my earlier study, the Matlab calculator, probTable, was developed (Zhang, 2025). In probTable, probability distributions as normal distribution, *t* distribution, *F*

distribution, χ^2 distribution, etc., were available and probability density function, cumulative distribution function, and inverse cumulative distribution function of these probability distributions can be calculated. In another study, a calculator, probDistriCal, was developed for probability distributions (Zhang and Qi, 2025). However, it can only calculate the probability for a known probability distribution. While basic distributions such as the normal, *t*, χ^2, and *F* distributions are well-established and widely implemented, many other important distributions play crucial roles in specialized applications.

The original probFunCal calculator (Zhang, 2026) provided essential functionality for these fundamental distributions, but significant gaps remained for other commonly used distributions. In this study, I developed probFunCal2, an advanced web-based calculator for probability distribution functions that complements the original probFunCal by providing comprehensive support for additional probability distributions. The calculator includes binomial, Poisson, exponential, gamma, beta, uniform, Weibull, lognormal, Cauchy, geometric, and negative binomial distributions.

The development of probFunCal2 was motivated by several practical needs: (1) the requirement for accessible, platform-independent tools for probability calculations; (2) the need for accurate numerical implementations of complex distribution functions; (3) the demand for educational tools that demonstrate both theoretical and computational aspects of probability distributions; and (4) the necessity of offline-capable applications for fieldwork and environments with limited internet connectivity.

2 Probability Distributions

Consider a random variable *X* with probability density function (PDF) $f(x)$ or probability mass function $f(k)$ for discrete cases. The cumulative distribution function (CDF) is defined as $F(x) = P(X<x) = \int_{-\infty}^{x} f(t)\,dt$ for continuous distributions, and $F(k) = P(X \le k) = \sum_{i=0}^{k} f(i)$ for discrete distributions. The quantile function is the inverse $F^{-1}(p)$ such that $P(X \le F^{-1}(p)) = p$.

2.1 Binomial Distribution (Zhang, 2007; Liu and Zhang, 2011; Zhang and Qi, 2025)

$$P_r = C_r^n p^r q^{n-r}$$
$$r>0$$

where P_r is the the probability that event A occurs *r* times in *n* trials, *p* is the probability of event A occurring in each trial.

2.2 Poission Distribution (Zhang, 2007; Liu and Zhang, 2011; Zhang and Qi, 2025)

$$P_r = e^{-\lambda}\lambda^r/r!$$
$$r>0$$

where P_r is the the probability that event A occurs *r* times, and λ is the mean, $\lambda>0$.

2.3 Exponential Distribution (Zhang, 2025; Zhang and Qi, 2025)

$$f(x)=\lambda e^{-\lambda x},\ x\ge 0$$
$$f(x)=0,\ x<0$$

where $\lambda>0$. The mean is $1/\lambda$ and variance is $1/\lambda^2$.

2.4 Gamma Distribution (Zhang, 2025)

The gamma distribution $\Gamma(\alpha, \beta)$ generalizes the exponential and χ^2 distributions, with shape parameter $\alpha > 0$ and scale parameter $\beta > 0$:

$$f(x)=\frac{1}{\beta^{\alpha+1}\boldsymbol{\Gamma}(\alpha+1)}x^{\alpha}e^{-\frac{x}{\beta}},\ x\geq 0$$

$f(x)=0,\ x<0$

where $\beta>0$, $c>-1$. The mean is $\alpha\beta$ and variance is $\alpha\beta^2$. When $\alpha = k/2$ and $\beta = 2$, this becomes the χ^2 distribution with k degrees of freedom.

2.5 Beta Distribution

The beta distribution $B(\alpha, \beta)$ models proportions or probabilities, defined on [0, 1]:

$$f(x|\alpha, \beta) = [\Gamma(\alpha+\beta)/(\Gamma(\alpha)\Gamma(\beta))]\ x^{\alpha-1}(1-x)^{\beta-1},\ 0 < x < 1$$

The mean is $\alpha/(\alpha+\beta)$ and variance is $\alpha\beta/[(\alpha+\beta)^2(\alpha+\beta+1)]$. Special cases include uniform ($\alpha = \beta = 1$) and arcsine ($\alpha = \beta = 1/2$) distributions.

2.6 Uniform Distribution

The uniform distribution $U(a, b)$ has constant density on interval $[a, b]$:

$$f(x|a, b) = 1/(b-a),\ a \leq x \leq b$$

The mean is $(a+b)/2$. This serves as a foundational distribution for Monte Carlo methods.

2.7 Weibull Distribution (Zhang, 2025; Zhang and Qi, 2025)

The Weibull distribution $W(m, v, x_0)$ models lifetime data and reliability analysis:

$$f(x)=\frac{m}{x_0}(x-v)^{m-1}e^{-\frac{(x-v)^m}{x_0}},\ x\geq v$$

$f(x)=0,\ x<v$

where, $m>0$, $x_0>0$, and v are constants.

2.8 Lognormal Distribution

The lognormal distribution $LN(\mu, \sigma)$ has a normally distributed logarithm:

$$f(x|\mu, \sigma) = \frac{1}{x\sigma\sqrt{(2\pi)}}\ e^{-[\ln(x)-\mu]^2/(2\sigma^2)},\ x > 0$$

The mean is $e^{\mu+\sigma^2/2}$ and variance is $[e^{\sigma^2}-1]e^{2\mu+\sigma^2}$. This distribution models multiplicative processes and financial returns.

2.9 Cauchy Distribution

The Cauchy distribution $C(x_0, \gamma)$ has heavy tails and no defined moments:

$$f(x|x_0, \gamma) = \frac{1}{\pi\gamma}\ [1 + (\frac{x-x_0}{\gamma})^2]^{-1},\ -\infty < x < \infty$$

The location parameter is x_0 and scale is $\gamma > 0$. The median equals the location parameter.

2.10 Geometric Distribution

The geometric distribution $G(p)$ models the number of failures before the first success in Bernoulli trials with success probability p:

$f(k|p) = (1-p)^k p$, $k = 0, 1, 2, ...$

The mean is $(1-p)/p$ and variance is $(1-p)/p^2$.

2.11 Negative Binomial Distribution

The negative binomial distribution $NB(r, p)$ models the number of failures before r successes:

$f(k|r, p) = C(k+r-1, k)\, p^r\, (1-p)^k$, $k = 0, 1, 2, ...$

The mean is $r(1-p)/p$ and variance is $r(1-p)/p^2$. This generalizes the geometric distribution ($r = 1$).

3 Numerical Implementation

3.1 Binomial Distribution

The binomial PDF uses log-factorials to avoid overflow: $\log[f(k)] = \log[\Gamma(n+1)] - \log[\Gamma(k+1)] - \log[\Gamma(n-k+1)] + k \log(p) + (n-k) \log(1-p)$. The CDF is computed via direct summation for small n, with normal approximation for large n. The quantile function employs binary search for small n and normal approximation for large n.

3.2 Poisson Distribution

The Poisson PMF uses the recursive relation $f(k+1) = f(k) \times \lambda/(k+1)$ to improve numerical stability. The CDF sums terms until convergence. For quantiles, normal approximation $N(\lambda, \sqrt{\lambda})$ provides good accuracy for $\lambda > 10$.

3.3 Exponential Distribution

The exponential distribution admits closed-form solutions: PDF = $\lambda e^{-\lambda x}$, CDF = $1- e^{-\lambda x}$, quantile = $-\ln(1-p)/\lambda$. These are implemented directly with careful handling of numerical underflow.

3.4 Gamma Distribution

The gamma PDF requires the gamma function $\Gamma(\alpha)$, implemented using Lanczos approximation for accuracy. The CDF uses the regularized incomplete gamma function via continued fraction expansion. Quantiles employ the Wilson-Hilferty transformation, which approximates the gamma quantile via normal quantiles.

3.5 Beta Distribution

The beta PDF uses the beta function $B(\alpha, \beta) = \Gamma(\alpha)\Gamma(\beta)/\Gamma(\alpha+\beta)$. The CDF implements the regularized incomplete beta function using continued fractions, with the choice between direct and complementary forms based on $x < (\alpha+1)/(\alpha+\beta+2)$. Quantiles use Newton-Raphson iteration on the inverse regularized beta function.

3.6 Other Distributions

The uniform, Weibull, lognormal, and Cauchy distributions have closed-form expressions that are implemented directly. The geometric distribution uses the geometric series sum for the CDF. The negative binomial follows the binomial implementation pattern but with modified parameters.

3.7 Numerical Stability

Several techniques ensure numerical stability: (1) logarithmic computations for products and factorials; (2) recursive relations to avoid repeated gamma function calls; (3) careful handling of boundary cases (p=0, p=1,

x=0, x=∞); (4) approximation methods for large parameter values; and (5) convergence criteria for iterative methods.

4 Web Implementation

probFunCal2 is implemented as a single HTML file with embedded JavaScript, ensuring maximum portability. The user interface follows the design pattern established in probFunCal, with separate forms for each distribution's PDF, CDF, and quantile functions. Input validation prevents invalid parameter combinations, and results are displayed with appropriate precision (8 decimal places for probabilities, 6 for continuous quantiles).

The calculator supports both online and offline usage, requiring no server-side processing. Cross-browser compatibility is achieved through standard HTML5 and ECMAScript 5 features. Mobile responsiveness is provided through CSS media queries, ensuring usability on tablets and smartphones.

The following are the complete HTML+JavaScript codes of the calculator (http://www.iaees.org/publications/journals/selforganizology/articles/2026-13(3-4)/1-Zhang-Abstract.asp; Fig.1):

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta contentType="text/html; charset=utf-8">
<meta name="description" content="probFunCal2: A web-based calculator for calculating probability distribution functions as binomial, Poisson, exponential, gamma, beta, uniform, Weibull, lognormal, Cauchy, geometric, and negative binomial distributions" />
<meta name="keywords" content="probability distribution functions, web-based calculator, binomial, poisson, exponential, gamma, beta, uniform, weibull, lognormal, cauchy" />
<meta name="author" content="W. J. Zhang" />
<link href="../../../../../style.css" rel="stylesheet" media="screen" type="text/css">
<title>probFunCal2: A web-based calculator for calculating probability distribution functions as binomial, Poisson, exponential, gamma, beta, uniform, Weibull, lognormal, Cauchy, geometric, and negative binomial distributions</title>

<!-- Google tag (gtag.js) -->
<script async src="https://www.googletagmanager.com/gtag/js?id=G-S56S6PGDYX"></script>
<script>
  window.dataLayer = window.dataLayer || [];
  function gtag(){dataLayer.push(arguments);}
  gtag('js', new Date());
  gtag('config', 'G-S56S6PGDYX');
</script>
</head>

<body>

<script language="javascript">

// Binomial Distribution PDF
```

```
function binomialPDF(k, n, p) {
    if (k < 0 || k > n || n < 0 || p < 0 || p > 1) return 0;
    if (n === 0) return k === 0 ? 1 : 0;

    const logCoeff = logFactorial(n) - logFactorial(k) - logFactorial(n - k);
    const logProb = k * Math.log(p) + (n - k) * Math.log(1 - p);
    return Math.exp(logCoeff + logProb);
}

// Binomial Distribution CDF
function binomialCDF(k, n, p) {
    if (k < 0) return 0;
    if (k >= n) return 1;

    let sum = 0;
    for (let i = 0; i <= k; i++) {
        sum += binomialPDF(i, n, p);
    }
    return sum;
}

// Binomial Quantile (approximation)
function binomialQuantile(p, n, prob) {
    if (p <= 0) return 0;
    if (p >= 1) return n;

    // Use normal approximation for large n
    if (n > 30) {
        const mu = n * prob;
        const sigma = Math.sqrt(n * prob * (1 - prob));
        const z = normalQuantile(p);
        return Math.max(0, Math.min(n, Math.round(mu + z * sigma)));
    }

    // Binary search for small n
    let low = 0, high = n;
    while (high - low > 1e-6) {
        const mid = Math.floor((low + high) / 2);
        if (binomialCDF(mid, n, prob) < p) {
            low = mid;
        } else {
            high = mid;
        }
    }
    return Math.round(low);
```

```
}

// Poisson Distribution PDF
function poissonPDF(k, lambda) {
    if (k < 0 || lambda <= 0) return 0;
    return Math.exp(-lambda) * Math.pow(lambda, k) / factorial(k);
}

// Poisson Distribution CDF
function poissonCDF(k, lambda) {
    if (k < 0) return 0;

    let sum = 0;
    for (let i = 0; i <= k; i++) {
        sum += poissonPDF(i, lambda);
    }
    return sum;
}

// Poisson Quantile (approximation)
function poissonQuantile(p, lambda) {
    if (p <= 0) return 0;
    if (p >= 1) return Math.ceil(lambda + 10 * Math.sqrt(lambda));

    // Use normal approximation
    const mu = lambda;
    const sigma = Math.sqrt(lambda);
    const z = normalQuantile(p);
    return Math.max(0, Math.round(mu + z * sigma));
}

// Exponential Distribution PDF
function exponentialPDF(x, lambda) {
    if (x < 0 || lambda <= 0) return 0;
    return lambda * Math.exp(-lambda * x);
}

// Exponential Distribution CDF
function exponentialCDF(x, lambda) {
    if (x < 0) return 0;
    if (x === Infinity) return 1;
    return 1 - Math.exp(-lambda * x);
}

// Exponential Quantile
```

```
function exponentialQuantile(p, lambda) {
    if (p <= 0 || p >= 1 || lambda <= 0) return NaN;
    return -Math.log(1 - p) / lambda;
}

// Gamma Distribution PDF
function gammaPDF(x, shape, scale) {
    if (x <= 0 || shape <= 0 || scale <= 0) return 0;
    const rate = 1 / scale;
    return Math.pow(rate, shape) * Math.pow(x, shape - 1) * Math.exp(-rate * x) / gamma(shape);
}

// Gamma Distribution CDF (using continued fraction approximation)
function gammaCDF(x, shape, scale) {
    if (x <= 0) return 0;
    const rate = 1 / scale;
    const z = rate * x;
    return regularizedGamma(shape, z);
}

// Gamma Quantile (approximation)
function gammaQuantile(p, shape, scale) {
    if (p <= 0 || p >= 1 || shape <= 0 || scale <= 0) return NaN;

    // Wilson-Hilferty approximation for gamma quantiles
    const a = 1 / (3 * Math.sqrt(shape));
    const z = normalQuantile(p);
    const x = Math.pow(z * a + 1 - a, 3);
    return shape * scale * x;
}

// Beta Distribution PDF
function betaPDF(x, alpha, beta) {
    if (x <= 0 || x >= 1 || alpha <= 0 || beta <= 0) return 0;
    return Math.pow(x, alpha - 1) * Math.pow(1 - x, beta - 1) / betaFunction(alpha, beta);
}

// Beta Distribution CDF
function betaCDF(x, alpha, beta) {
    if (x <= 0) return 0;
    if (x >= 1) return 1;
    return regularizedBeta(x, alpha, beta);
}

// Beta Quantile
```

```
function betaQuantile(p, alpha, beta) {
    if (p <= 0 || p >= 1 || alpha <= 0 || beta <= 0) return NaN;
    return inverseRegularizedBeta(p, alpha, beta);
}

// Uniform Distribution PDF
function uniformPDF(x, a, b) {
    if (x < a || x > b || a >= b) return 0;
    return 1 / (b - a);
}

// Uniform Distribution CDF
function uniformCDF(x, a, b) {
    if (x < a) return 0;
    if (x > b) return 1;
    return (x - a) / (b - a);
}

// Uniform Quantile
function uniformQuantile(p, a, b) {
    if (p <= 0 || p >= 1 || a >= b) return NaN;
    return a + p * (b - a);
}

// Weibull Distribution PDF
function weibullPDF(x, shape, scale) {
    if (x < 0 || shape <= 0 || scale <= 0) return 0;
    return (shape / scale) * Math.pow(x / scale, shape - 1) * Math.exp(-Math.pow(x / scale, shape));
}

// Weibull Distribution CDF
function weibullCDF(x, shape, scale) {
    if (x < 0) return 0;
    if (x === Infinity) return 1;
    return 1 - Math.exp(-Math.pow(x / scale, shape));
}

// Weibull Quantile
function weibullQuantile(p, shape, scale) {
    if (p <= 0 || p >= 1 || shape <= 0 || scale <= 0) return NaN;
    return scale * Math.pow(-Math.log(1 - p), 1 / shape);
}

// Lognormal Distribution PDF
function lognormalPDF(x, mu, sigma) {
```

```
    if (x <= 0 || sigma <= 0) return 0;
    const z = (Math.log(x) - mu) / sigma;
    return normalPDF(z, 0, 1) / (x * sigma);
}

// Lognormal Distribution CDF
function lognormalCDF(x, mu, sigma) {
    if (x <= 0) return 0;
    const z = (Math.log(x) - mu) / sigma;
    return normalCDF(z);
}

// Lognormal Quantile
function lognormalQuantile(p, mu, sigma) {
    if (p <= 0 || p >= 1 || sigma <= 0) return NaN;
    const z = normalQuantile(p);
    return Math.exp(mu + sigma * z);
}

// Cauchy Distribution PDF
function cauchyPDF(x, location, scale) {
    if (scale <= 0) return 0;
    const z = (x - location) / scale;
    return 1 / (Math.PI * scale * (1 + z * z));
}

// Cauchy Distribution CDF
function cauchyCDF(x, location, scale) {
    if (scale <= 0) return NaN;
    const z = (x - location) / scale;
    return 0.5 + Math.atan(z) / Math.PI;
}

// Cauchy Quantile
function cauchyQuantile(p, location, scale) {
    if (p <= 0 || p >= 1 || scale <= 0) return NaN;
    return location + scale * Math.tan(Math.PI * (p - 0.5));
}

// Geometric Distribution PDF
function geometricPDF(k, p) {
    if (k < 0 || p <= 0 || p >= 1) return 0;
    return (1 - p) ** k * p;
}
```

```
// Geometric Distribution CDF
function geometricCDF(k, p) {
    if (k < 0) return 0;
    return 1 - Math.pow(1 - p, k + 1);
}

// Geometric Quantile
function geometricQuantile(p, prob) {
    if (p <= 0 || p >= 1 || prob <= 0 || prob >= 1) return NaN;
    return Math.ceil(Math.log(1 - p) / Math.log(1 - prob)) - 1;
}

// Negative Binomial Distribution PDF
function negBinomialPDF(k, r, p) {
    if (k < 0 || r <= 0 || p <= 0 || p >= 1) return 0;
    const logCoeff = logFactorial(k + r - 1) - logFactorial(k) - logFactorial(r - 1);
    const logProb = k * Math.log(1 - p) + r * Math.log(p);
    return Math.exp(logCoeff + logProb);
}

// Negative Binomial Distribution CDF
function negBinomialCDF(k, r, p) {
    if (k < 0) return 0;

    let sum = 0;
    for (let i = 0; i <= k; i++) {
        sum += negBinomialPDF(i, r, p);
    }
    return sum;
}

// Negative Binomial Quantile (approximation)
function negBinomialQuantile(p, r, prob) {
    if (p <= 0 || p >= 1 || r <= 0 || prob <= 0 || prob >= 1) return NaN;

    // Use normal approximation
    const mu = r * (1 - prob) / prob;
    const variance = r * (1 - prob) / (prob * prob);
    const sigma = Math.sqrt(variance);
    const z = normalQuantile(p);
    return Math.max(0, Math.round(mu + z * sigma));
}

// Helper functions
function factorial(n) {
```

```
    if (n < 0) return NaN;
    if (n === 0 || n === 1) return 1;
    let result = 1;
    for (let i = 2; i <= n; i++) {
        result *= i;
    }
    return result;
}

function logFactorial(n) {
    if (n < 0) return NaN;
    if (n === 0 || n === 1) return 0;

    // Stirling's approximation for large n
    if (n > 20) {
        return n * Math.log(n) - n + 0.5 * Math.log(2 * Math.PI * n) - 1/(12*n);
    }

    let result = 0;
    for (let i = 2; i <= n; i++) {
        result += Math.log(i);
    }
    return result;
}

// Include necessary functions from the original calculator
function normalPDF(x, mean, sd) {
    if (sd <= 0) return 0;
    const variance = sd * sd;
    const coefficient = 1 / Math.sqrt(2 * Math.PI * variance);
    const exponent = - Math.pow(x - mean, 2) / (2 * variance);
    return coefficient * Math.exp(exponent);
}

function normalCDF(z) {
    const a1 = 0.254829592;
    const a2 = -0.284496736;
    const a3 = 1.421413741;
    const a4 = -1.453152027;
    const a5 = 1.061405429;
    const p = 0.3275911;
    const sign = z < 0 ? -1 : 1;
    z = Math.abs(z) / Math.sqrt(2.0);
    const t = 1.0 / (1.0 + p * z);
    const y = 1.0 - ((((a5 * t + a4) * t + a3) * t + a2) * t + a1) * t * Math.exp(-z * z);
```

```
    return 0.5 * (1.0 + sign * y);
}

function normalQuantile(p) {
    if (p <= 0 || p >= 1) return NaN;
    const a1 = -3.969683028665376e+01;
    const a2 =  2.209460984245205e+02;
    const a3 = -2.759285104469687e+02;
    const a4 =  1.383577518672690e+02;
    const a5 = -3.066479806614716e+01;
    const a6 =  2.506628277459239e+00;
    const b1 = -5.447609879822406e+01;
    const b2 =  1.615858368580409e+02;
    const b3 = -1.556989798598866e+02;
    const b4 =  6.680131188771972e+01;
    const b5 = -1.328068155288572e+01;
    const c1 = -7.784894002430293e-03;
    const c2 = -3.223964580411365e-01;
    const c3 = -2.400758277161838e+00;
    const c4 = -2.549732539343734e+00;
    const c5 =  4.374664141464968e+00;
    const c6 =  2.938163982698783e+00;
    const d1 =  7.784695709041462e-03;
    const d2 =  3.224671290700398e-01;
    const d3 =  2.445134137142996e+00;
    const d4 =  3.754408661907416e+00;

    let q, r;
    if (p < 0.02425) {
        q = Math.sqrt(-2 * Math.log(p));
        return (((((c1 * q + c2) * q + c3) * q + c4) * q + c5) * q + c6) /
                (((d1 * q + d2) * q + d3) * q + d4) * q + 1);
    } else if (p > 0.97575) {
        q = Math.sqrt(-2 * Math.log(1 - p));
        return -(((((c1 * q + c2) * q + c3) * q + c4) * q + c5) * q + c6) /
                (((d1 * q + d2) * q + d3) * q + d4) * q + 1);
    } else {
        q = p - 0.5;
        r = q * q;
        return (((((a1 * r + a2) * r + a3) * r + a4) * r + a5) * r + a6) * q /
                (((((b1 * r + b2) * r + b3) * r + b4) * r + b5) * r + 1);
    }
}

function gamma(x) {
```

```
    const p = [0.99999999999980993, 676.5203681218851, -1259.1392167224028,
                771.32342877765313, -176.61502916214059, 12.507343278686905,
                -0.13857109526572012, 9.9843695780195716e-6, 1.5056327351493116e-7];
    if (x < 0.5) {
        return Math.PI / (Math.sin(Math.PI * x) * gamma(1 - x));
    }
    x -= 1;
    let t = p[0];
    for (let i = 1; i < p.length; i++) {
        t += p[i] / (x + i);
    }
    const w = x + p.length - 1.5;
    return Math.sqrt(2 * Math.PI) * Math.pow(w, x + 0.5) * Math.exp(-w) * t;
}

function betaFunction(a, b) {
    return Math.exp(logGamma(a) + logGamma(b) - logGamma(a + b));
}

function logGamma(z) {
    const g = 7;
    const p = [0.99999999999980993, 676.5203681218851, -1259.1392167224028,
                771.32342877765313, -176.61502916214059, 12.507343278686905,
                -0.13857109526572012, 9.9843695780195716e-6, 1.5056327351493116e-7];
    if (z < 0.5) {
        return Math.log(Math.PI) - Math.log(Math.sin(Math.PI * z)) - logGamma(1 - z);
    }
    z -= 1;
    let x = p[0];
    for (let i = 1; i < p.length; i++) {
        x += p[i] / (z + i);
    }
    const t = z + g + 0.5;
    return 0.5 * Math.log(2 * Math.PI) + (z + 0.5) * Math.log(t) - t + Math.log(x);
}

function regularizedBeta(x, a, b) {
    if (x < 0 || x > 1) return NaN;
    if (x === 0) return 0;
    if (x === 1) return 1;
    if (a <= 0 || b <= 0) return NaN;
    const eps = 1e-12;
    const maxIter = 10000;
    const logBeta = logGamma(a + b) - logGamma(a) - logGamma(b) +
                    a * Math.log(x) + b * Math.log1p(-x);
```

```
    if (x < (a + 1) / (a + b + 2)) {
        const cf = continuedFraction(x, a, b, eps, maxIter);
        return Math.exp(logBeta) * cf / a;
    } else {
        const cf = continuedFraction(1 - x, b, a, eps, maxIter);
        return 1 - Math.exp(logBeta) * cf / b;
    }
}

function continuedFraction(x, a, b, eps, maxIter) {
    let f = 1, c = 1, d = 0;
    let m = 0;
    do {
        m++;
        const _2m = 2 * m;
        const _2m1 = 2 * m - 1;
        const numerator = m * (b - m) * x;
        const denominator1 = (a + _2m1) * (a + _2m);
        const denominator2 = (a + _2m) * (a + _2m1);
        d = 1 + numerator / denominator1 * d;
        if (Math.abs(d) < eps) d = eps;
        d = 1 / d;
        c = 1 + numerator / denominator2 * c;
        if (Math.abs(c) < eps) c = eps;
        const delta = d * c;
        f *= delta;
        if (m > maxIter) break;
    } while (Math.abs(f - 1) >= eps);
    return f;
}

function inverseRegularizedBeta(p, a, b, tol = 1e-12, maxIter = 1000) {
    let x = (a < 1 && b < 1) ? 0.5 :
            (a < 1) ? 0.25 :
            (b < 1) ? 0.75 :
            a / (a + b);
    let iter = 0;
    let delta = 0;
    do {
        const f = regularizedBeta(x, a, b) - p;
        const pdf = Math.exp(
            (a - 1) * Math.log(x) +
            (b - 1) * Math.log1p(-x) -
            logBeta(a, b)
        );
```

```
            delta = f / pdf;
            x -= delta;
            if (x <= 0) x = tol;
            if (x >= 1) x = 1 - tol;
            iter++;
        } while (Math.abs(delta) > tol && iter < maxIter);
        return x;
}

function logBeta(a, b) {
        return logGamma(a) + logGamma(b) - logGamma(a + b);
}

function regularizedGamma(s, x) {
        if (x < 0 || s <= 0) return NaN;
        if (x === 0) return 0;

        // Use continued fraction approximation
        const eps = 1e-12;
        const maxIter = 1000;

        let gln = logGamma(s);
        let b = x + 1 - s;
        let c = 1 / 1e-30;
        let d = 1 / b;
        let h = d;

        for (let i = 1; i <= maxIter; i++) {
            const an = -i * (i - s);
            b += 2;
            d = an * d + b;
            if (Math.abs(d) < eps) d = eps;
            c = b + an / c;
            if (Math.abs(c) < eps) c = eps;
            d = 1 / d;
            const del = d * c;
            h *= del;
            if (Math.abs(del - 1) < eps) break;
        }

        return 1 - Math.exp(-x + s * Math.log(x) - gln) * h;
}

// Form handler functions
function runBinomialPDF() {
```

```
    var k = parseInt(document.formBinomialPDF.k.value);
    var n = parseInt(document.formBinomialPDF.n.value);
    var p = parseFloat(document.formBinomialPDF.p.value);
    var fx = binomialPDF(k, n, p);
    var strs = fx.toExponential(8);
    document.formBinomialPDF.textBinomialOutPDF.value = strs;
}

function runBinomialCDF() {
    var k = parseInt(document.formBinomialCDF.k.value);
    var n = parseInt(document.formBinomialCDF.n.value);
    var p = parseFloat(document.formBinomialCDF.p.value);
    var fx = binomialCDF(k, n, p);
    var strs = fx.toFixed(8);
    document.formBinomialCDF.textBinomialOutCDF.value = strs;
}

function runBinomialQuantile() {
    var p = parseFloat(document.formBinomialQuantile.p.value);
    var n = parseInt(document.formBinomialQuantile.n.value);
    var prob = parseFloat(document.formBinomialQuantile.prob.value);
    var x = binomialQuantile(p, n, prob);
    var strs = String(x);
    document.formBinomialQuantile.textBinomialOutQuantile.value = strs;
}

function runPoissonPDF() {
    var k = parseInt(document.formPoissonPDF.k.value);
    var lambda = parseFloat(document.formPoissonPDF.lambda.value);
    var fx = poissonPDF(k, lambda);
    var strs = fx.toExponential(8);
    document.formPoissonPDF.textPoissonOutPDF.value = strs;
}

function runPoissonCDF() {
    var k = parseInt(document.formPoissonCDF.k.value);
    var lambda = parseFloat(document.formPoissonCDF.lambda.value);
    var fx = poissonCDF(k, lambda);
    var strs = fx.toFixed(8);
    document.formPoissonCDF.textPoissonOutCDF.value = strs;
}

function runPoissonQuantile() {
    var p = parseFloat(document.formPoissonQuantile.p.value);
    var lambda = parseFloat(document.formPoissonQuantile.lambda.value);
```

```
	var x = poissonQuantile(p, lambda);
	var strs = String(x);
	document.formPoissonQuantile.textPoissonOutQuantile.value = strs;
}

function runExponentialPDF() {
	var x = parseFloat(document.formExponentialPDF.x.value);
	var lambda = parseFloat(document.formExponentialPDF.lambda.value);
	var fx = exponentialPDF(x, lambda);
	var strs = fx.toExponential(8);
	document.formExponentialPDF.textExponentialOutPDF.value = strs;
}

function runExponentialCDF() {
	var x = parseFloat(document.formExponentialCDF.x.value);
	var lambda = parseFloat(document.formExponentialCDF.lambda.value);
	var fx = exponentialCDF(x, lambda);
	var strs = fx.toFixed(8);
	document.formExponentialCDF.textExponentialOutCDF.value = strs;
}

function runExponentialQuantile() {
	var p = parseFloat(document.formExponentialQuantile.p.value);
	var lambda = parseFloat(document.formExponentialQuantile.lambda.value);
	var x = exponentialQuantile(p, lambda);
	var strs = x.toFixed(6);
	document.formExponentialQuantile.textExponentialOutQuantile.value = strs;
}

function runGammaPDF() {
	var x = parseFloat(document.formGammaPDF.x.value);
	var shape = parseFloat(document.formGammaPDF.shape.value);
	var scale = parseFloat(document.formGammaPDF.scale.value);
	var fx = gammaPDF(x, shape, scale);
	var strs = fx.toExponential(8);
	document.formGammaPDF.textGammaOutPDF.value = strs;
}

function runGammaCDF() {
	var x = parseFloat(document.formGammaCDF.x.value);
	var shape = parseFloat(document.formGammaCDF.shape.value);
	var scale = parseFloat(document.formGammaCDF.scale.value);
	var fx = gammaCDF(x, shape, scale);
	var strs = fx.toFixed(8);
	document.formGammaCDF.textGammaOutCDF.value = strs;
```

```
}

function runGammaQuantile() {
    var p = parseFloat(document.formGammaQuantile.p.value);
    var shape = parseFloat(document.formGammaQuantile.shape.value);
    var scale = parseFloat(document.formGammaQuantile.scale.value);
    var x = gammaQuantile(p, shape, scale);
    var strs = x.toFixed(6);
    document.formGammaQuantile.textGammaOutQuantile.value = strs;
}

function runBetaPDF() {
    var x = parseFloat(document.formBetaPDF.x.value);
    var alpha = parseFloat(document.formBetaPDF.alpha.value);
    var beta = parseFloat(document.formBetaPDF.beta.value);
    var fx = betaPDF(x, alpha, beta);
    var strs = fx.toExponential(8);
    document.formBetaPDF.textBetaOutPDF.value = strs;
}

function runBetaCDF() {
    var x = parseFloat(document.formBetaCDF.x.value);
    var alpha = parseFloat(document.formBetaCDF.alpha.value);
    var beta = parseFloat(document.formBetaCDF.beta.value);
    var fx = betaCDF(x, alpha, beta);
    var strs = fx.toFixed(8);
    document.formBetaCDF.textBetaOutCDF.value = strs;
}

function runBetaQuantile() {
    var p = parseFloat(document.formBetaQuantile.p.value);
    var alpha = parseFloat(document.formBetaQuantile.alpha.value);
    var beta = parseFloat(document.formBetaQuantile.beta.value);
    var x = betaQuantile(p, alpha, beta);
    var strs = x.toFixed(6);
    document.formBetaQuantile.textBetaOutQuantile.value = strs;
}

function runUniformPDF() {
    var x = parseFloat(document.formUniformPDF.x.value);
    var a = parseFloat(document.formUniformPDF.a.value);
    var b = parseFloat(document.formUniformPDF.b.value);
    var fx = uniformPDF(x, a, b);
    var strs = fx.toExponential(8);
    document.formUniformPDF.textUniformOutPDF.value = strs;
```

```
}

function runUniformCDF() {
    var x = parseFloat(document.formUniformCDF.x.value);
    var a = parseFloat(document.formUniformCDF.a.value);
    var b = parseFloat(document.formUniformCDF.b.value);
    var fx = uniformCDF(x, a, b);
    var strs = fx.toFixed(8);
    document.formUniformCDF.textUniformOutCDF.value = strs;
}

function runUniformQuantile() {
    var p = parseFloat(document.formUniformQuantile.p.value);
    var a = parseFloat(document.formUniformQuantile.a.value);
    var b = parseFloat(document.formUniformQuantile.b.value);
    var x = uniformQuantile(p, a, b);
    var strs = x.toFixed(6);
    document.formUniformQuantile.textUniformOutQuantile.value = strs;
}

function runWeibullPDF() {
    var x = parseFloat(document.formWeibullPDF.x.value);
    var shape = parseFloat(document.formWeibullPDF.shape.value);
    var scale = parseFloat(document.formWeibullPDF.scale.value);
    var fx = weibullPDF(x, shape, scale);
    var strs = fx.toExponential(8);
    document.formWeibullPDF.textWeibullOutPDF.value = strs;
}

function runWeibullCDF() {
    var x = parseFloat(document.formWeibullCDF.x.value);
    var shape = parseFloat(document.formWeibullCDF.shape.value);
    var scale = parseFloat(document.formWeibullCDF.scale.value);
    var fx = weibullCDF(x, shape, scale);
    var strs = fx.toFixed(8);
    document.formWeibullCDF.textWeibullOutCDF.value = strs;
}

function runWeibullQuantile() {
    var p = parseFloat(document.formWeibullQuantile.p.value);
    var shape = parseFloat(document.formWeibullQuantile.shape.value);
    var scale = parseFloat(document.formWeibullQuantile.scale.value);
    var x = weibullQuantile(p, shape, scale);
    var strs = x.toFixed(6);
    document.formWeibullQuantile.textWeibullOutQuantile.value = strs;
```

```
}

function runLognormalPDF() {
    var x = parseFloat(document.formLognormalPDF.x.value);
    var mu = parseFloat(document.formLognormalPDF.mu.value);
    var sigma = parseFloat(document.formLognormalPDF.sigma.value);
    var fx = lognormalPDF(x, mu, sigma);
    var strs = fx.toExponential(8);
    document.formLognormalPDF.textLognormalOutPDF.value = strs;
}

function runLognormalCDF() {
    var x = parseFloat(document.formLognormalCDF.x.value);
    var mu = parseFloat(document.formLognormalCDF.mu.value);
    var sigma = parseFloat(document.formLognormalCDF.sigma.value);
    var fx = lognormalCDF(x, mu, sigma);
    var strs = fx.toFixed(8);
    document.formLognormalCDF.textLognormalOutCDF.value = strs;
}

function runLognormalQuantile() {
    var p = parseFloat(document.formLognormalQuantile.p.value);
    var mu = parseFloat(document.formLognormalQuantile.mu.value);
    var sigma = parseFloat(document.formLognormalQuantile.sigma.value);
    var x = lognormalQuantile(p, mu, sigma);
    var strs = x.toFixed(6);
    document.formLognormalQuantile.textLognormalOutQuantile.value = strs;
}

function runCauchyPDF() {
    var x = parseFloat(document.formCauchyPDF.x.value);
    var location = parseFloat(document.formCauchyPDF.location.value);
    var scale = parseFloat(document.formCauchyPDF.scale.value);
    var fx = cauchyPDF(x, location, scale);
    var strs = fx.toExponential(8);
    document.formCauchyPDF.textCauchyOutPDF.value = strs;
}

function runCauchyCDF() {
    var x = parseFloat(document.formCauchyCDF.x.value);
    var location = parseFloat(document.formCauchyCDF.location.value);
    var scale = parseFloat(document.formCauchyCDF.scale.value);
    var fx = cauchyCDF(x, location, scale);
    var strs = fx.toFixed(8);
    document.formCauchyCDF.textCauchyOutCDF.value = strs;
```

```
}

function runCauchyQuantile() {
    var p = parseFloat(document.formCauchyQuantile.p.value);
    var location = parseFloat(document.formCauchyQuantile.location.value);
    var scale = parseFloat(document.formCauchyQuantile.scale.value);
    var x = cauchyQuantile(p, location, scale);
    var strs = x.toFixed(6);
    document.formCauchyQuantile.textCauchyOutQuantile.value = strs;
}

function runGeometricPDF() {
    var k = parseInt(document.formGeometricPDF.k.value);
    var p = parseFloat(document.formGeometricPDF.p.value);
    var fx = geometricPDF(k, p);
    var strs = fx.toFixed(8);
    document.formGeometricPDF.textGeometricOutPDF.value = strs;
}

function runGeometricCDF() {
    var k = parseInt(document.formGeometricCDF.k.value);
    var p = parseFloat(document.formGeometricCDF.p.value);
    var fx = geometricCDF(k, p);
    var strs = fx.toFixed(8);
    document.formGeometricCDF.textGeometricOutCDF.value = strs;
}

function runGeometricQuantile() {
    var p = parseFloat(document.formGeometricQuantile.p.value);
    var prob = parseFloat(document.formGeometricQuantile.prob.value);
    var x = geometricQuantile(p, prob);
    var strs = String(x);
    document.formGeometricQuantile.textGeometricOutQuantile.value = strs;
}

function runNegBinomialPDF() {
    var k = parseInt(document.formNegBinomialPDF.k.value);
    var r = parseFloat(document.formNegBinomialPDF.r.value);
    var p = parseFloat(document.formNegBinomialPDF.p.value);
    var fx = negBinomialPDF(k, r, p);
    var strs = fx.toExponential(8);
    document.formNegBinomialPDF.textNegBinomialOutPDF.value = strs;
}

function runNegBinomialCDF() {
```

```
	var k = parseInt(document.formNegBinomialCDF.k.value);
	var r = parseFloat(document.formNegBinomialCDF.r.value);
	var p = parseFloat(document.formNegBinomialCDF.p.value);
	var fx = negBinomialCDF(k, r, p);
	var strs = fx.toFixed(8);
	document.formNegBinomialCDF.textNegBinomialOutCDF.value = strs;
}

function runNegBinomialQuantile() {
	var p = parseFloat(document.formNegBinomialQuantile.p.value);
	var r = parseFloat(document.formNegBinomialQuantile.r.value);
	var prob = parseFloat(document.formNegBinomialQuantile.prob.value);
	var x = negBinomialQuantile(p, r, prob);
	var strs = String(x);
	document.formNegBinomialQuantile.textNegBinomialOutQuantile.value = strs;
}

</script>

<table border=1 cellspacing="1" cellpadding="1" width="100%">
<tr>
<th colspan=6><IMG SRC="../../../../../IAEES-Title.jpg" width="100%"></th>
</tr>
<tr bgcolor=yellow>
<th width="10%"><a
href="http://www.iaees.org/publications/journals/selforganizology/articles/2026-13(1-2)/2-Zhang-Abstract.asp">Home</a></th>
<th width="90%"></th>
</tr>
</table>

<font face="Times New Roman">

<br>
<div id="pagehead"></div>
<div align=center><b><h2>probFunCal2</h2></b></div>
<div align=center><b><h3>probFunCal2: A Web-Based Calculator For Calculating Probability Distribution Functions As Binomial, Poisson, Exponential, Gamma, Beta, Uniform, Weibull, Lognormal, Cauchy, Geometric, And Negative Binomial Distributions</h3></b></div>
<div align=center><b><h4>By W. J. Zhang</h4></b></div>
<br>

<br>The user manual guide and suggested citation of this page:
<br>
<b><a
href="http://www.iaees.org/publications/journals/selforganizology/articles/2026-13(3-4)/probFunCal2-for-advanced-probability-
```

distribution-functions.pdf">Zhang W. J. 2026. probFunCal2: A web-based calculator for calculating probability distribution functions as binomial, Poisson, exponential, gamma, beta, uniform, Weibull, lognormal, Cauchy, geometric, and negative binomial distributions. Selforganizology, 13(3-4): 26-65

Also, click here to download the corresponding offline calculator.

<hr>

Binomial Distribution
Poisson Distribution
Exponential Distribution
Gamma Distribution
Beta Distribution
Uniform Distribution
Weibull Distribution
Lognormal Distribution
Cauchy Distribution
Geometric Distribution
Negative Binomial Distribution

<hr color=maroon size="2px">

<!-- Binomial Distribution -->

Binomial Distribution

<form name="formBinomialPDF">
Number of trials (n): <input type="text" name="n" value="10">

Number of successes (k): <input type="text" name="k" value="5">

Probability of success (p): <input type="text" name="p" value="0.5">

<input type="button" name="buttonBinomialPDF" value="PDF" style="width:100px;" onClick="runBinomialPDF()">

Probability mass f(k):

<textarea id="textBinomialOutPDF" name="textBinomialOutPDF" rows="1" cols="30" style="background:cyan;font-family:Times New Roman;font-size:11pt;color:black" wrap="off" readonly></textarea>

```
</form>

<form name="formBinomialCDF">
Number of trials (<i>n</i>): <input type="text" name="n" value="10">
<br><br>
Number of successes (<i>k</i>): <input type="text" name="k" value="5">
<br><br>
Probability of success (<i>p</i>): <input type="text" name="p" value="0.5">
<br><br>
<input type="button" name="buttonBinomialCDF" value="CDF" style="width:100px;" onClick="runBinomialCDF()">
<br><br>
Cumulative probability <i>F</i>(<i>k</i>):
<br>
<textarea id="textBinomialOutCDF" name="textBinomialOutCDF" rows="1" cols="30"
style="background:cyan;font-family:Times New Roman;font-size:11pt;color:black" wrap="off" readonly></textarea>
</form>

<form name="formBinomialQuantile">
Probability (<i>p</i>): <input type="text" name="p" value="0.95">
<br><br>
Number of trials (<i>n</i>): <input type="text" name="n" value="10">
<br><br>
Probability of success (<i>prob</i>): <input type="text" name="prob" value="0.5">
<br><br>
<input type="button" name="buttonBinomialQuantile" value="Quantile" style="width:100px;"
onClick="runBinomialQuantile()">
<br><br>
Quantile (<i>k</i>):
<br>
<textarea id="textBinomialOutQuantile" name="textBinomialOutQuantile" rows="1" cols="30"
style="background:cyan;font-family:Times New Roman;font-size:11pt;color:black" wrap="off" readonly></textarea>
</form>

<br><br>
<a href="#pagehead" style="background-color:lightgray;border-style:solid;border-color:silver">   Back to
Top   </a>
<br><br>
<hr color=maroon size="2px"><br>

<!-- Poisson Distribution -->
<font face="Times New Roman" size=5>
<b><div id="poisson">Poisson Distribution</div></b>
</font>
<br>
```

```
<form name="formPoissonPDF">
Number of events (<i>k</i>): <input type="text" name="k" value="5">
<br><br>
Rate parameter (<i>λ</i>): <input type="text" name="lambda" value="3">
<br><br>
<input type="button" name="buttonPoissonPDF" value="PDF" style="width:100px;" onClick="runPoissonPDF()">
<br><br>
Probability mass <i>f</i>(<i>k</i>):
<br>
<textarea id="textPoissonOutPDF" name="textPoissonOutPDF" rows="1" cols="30" style="background:cyan;font-family:Times New Roman;font-size:11pt;color:black" wrap="off" readonly></textarea>
</form>

<form name="formPoissonCDF">
Number of events (<i>k</i>): <input type="text" name="k" value="5">
<br><br>
Rate parameter (<i>λ</i>): <input type="text" name="lambda" value="3">
<br><br>
<input type="button" name="buttonPoissonCDF" value="CDF" style="width:100px;" onClick="runPoissonCDF()">
<br><br>
Cumulative probability <i>F</i>(<i>k</i>):
<br>
<textarea id="textPoissonOutCDF" name="textPoissonOutCDF" rows="1" cols="30" style="background:cyan;font-family:Times New Roman;font-size:11pt;color:black" wrap="off" readonly></textarea>
</form>

<form name="formPoissonQuantile">
Probability (<i>p</i>): <input type="text" name="p" value="0.95">
<br><br>
Rate parameter (<i>λ</i>): <input type="text" name="lambda" value="3">
<br><br>
<input type="button" name="buttonPoissonQuantile" value="Quantile" style="width:100px;" onClick="runPoissonQuantile()">
<br><br>
Quantile (<i>k</i>):
<br>
<textarea id="textPoissonOutQuantile" name="textPoissonOutQuantile" rows="1" cols="30" style="background:cyan;font-family:Times New Roman;font-size:11pt;color:black" wrap="off" readonly></textarea>
</form>

<br><br>
<a href="#pagehead" style="background-color:lightgray;border-style:solid;border-color:silver">   Back to Top   </a>
<br><br>
<hr color=maroon size="2px"><br>
```

```
<!-- Exponential Distribution -->
<font face="Times New Roman" size=5>
<b><div id="exponential">Exponential Distribution</div></b>
</font>
<br>

<form name="formExponentialPDF">
Random variable (<i>x</i>): <input type="text" name="x" value="1">
<br><br>
Rate parameter (<i>λ</i>): <input type="text" name="lambda" value="1">
<br><br>
<input type="button" name="buttonExponentialPDF" value="PDF" style="width:100px;" onClick="runExponentialPDF()">
<br><br>
Probability density <i>f</i>(<i>x</i>):
<br>
<textarea id="textExponentialOutPDF" name="textExponentialOutPDF" rows="1" cols="30"
style="background:cyan;font-family:Times New Roman;font-size:11pt;color:black" wrap="off" readonly></textarea>
</form>

<form name="formExponentialCDF">
Random variable (<i>x</i>): <input type="text" name="x" value="1">
<br><br>
Rate parameter (<i>λ</i>): <input type="text" name="lambda" value="1">
<br><br>
<input type="button" name="buttonExponentialCDF" value="CDF" style="width:100px;" onClick="runExponentialCDF()">
<br><br>
Cumulative probability <i>F</i>(<i>x</i>):
<br>
<textarea id="textExponentialOutCDF" name="textExponentialOutCDF" rows="1" cols="30"
style="background:cyan;font-family:Times New Roman;font-size:11pt;color:black" wrap="off" readonly></textarea>
</form>

<form name="formExponentialQuantile">
Probability (<i>p</i>): <input type="text" name="p" value="0.95">
<br><br>
Rate parameter (<i>λ</i>): <input type="text" name="lambda" value="1">
<br><br>
<input type="button" name="buttonExponentialQuantile" value="Quantile" style="width:100px;"
onClick="runExponentialQuantile()">
<br><br>
Quantile (<i>x</i>):
<br>
<textarea id="textExponentialOutQuantile" name="textExponentialOutQuantile" rows="1" cols="30"
style="background:cyan;font-family:Times New Roman;font-size:11pt;color:black" wrap="off" readonly></textarea>
</form>
```

```
<br><br>
<a href="#pagehead" style="background-color:lightgray;border-style:solid;border-color:silver">   Back to Top   </a>
<br><br>
<hr color=maroon size="2px"><br>

<!-- Gamma Distribution -->
<font face="Times New Roman" size=5>
<b><div id="gamma">Gamma Distribution</div></b>
</font>
<br>

<form name="formGammaPDF">
Random variable (<i>x</i>): <input type="text" name="x" value="2">
<br><br>
Shape parameter (<i>α</i>): <input type="text" name="shape" value="2">
<br><br>
Scale parameter (<i>β</i>): <input type="text" name="scale" value="1">
<br><br>
<input type="button" name="buttonGammaPDF" value="PDF" style="width:100px;" onClick="runGammaPDF()">
<br><br>
Probability density <i>f</i>(<i>x</i>):
<br>
<textarea id="textGammaOutPDF" name="textGammaOutPDF" rows="1" cols="30" style="background:cyan;font-family:Times New Roman;font-size:11pt;color:black" wrap="off" readonly></textarea>
</form>

<form name="formGammaCDF">
Random variable (<i>x</i>): <input type="text" name="x" value="2">
<br><br>
Shape parameter (<i>α</i>): <input type="text" name="shape" value="2">
<br><br>
Scale parameter (<i>β</i>): <input type="text" name="scale" value="1">
<br><br>
<input type="button" name="buttonGammaCDF" value="CDF" style="width:100px;" onClick="runGammaCDF()">
<br><br>
Cumulative probability <i>F</i>(<i>x</i>):
<br>
<textarea id="textGammaOutCDF" name="textGammaOutCDF" rows="1" cols="30" style="background:cyan;font-family:Times New Roman;font-size:11pt;color:black" wrap="off" readonly></textarea>
</form>

<form name="formGammaQuantile">
Probability (<i>p</i>): <input type="text" name="p" value="0.95">
```

```
<br><br>
Shape parameter (<i>α</i>): <input type="text" name="shape" value="2">
<br><br>
Scale parameter (<i>β</i>): <input type="text" name="scale" value="1">
<br><br>
<input type="button" name="buttonGammaQuantile" value="Quantile" style="width:100px;" onClick="runGammaQuantile()">
<br><br>
Quantile (<i>x</i>):
<br>
<textarea id="textGammaOutQuantile" name="textGammaOutQuantile" rows="1" cols="30" style="background:cyan;font-family:Times New Roman;font-size:11pt;color:black" wrap="off" readonly></textarea>
</form>

<br><br>
<a href="#pagehead" style="background-color:lightgray;border-style:solid;border-color:silver">   Back to Top   </a>
<br><br>
<hr color=maroon size="2px"><br>

<!-- Beta Distribution -->
<font face="Times New Roman" size=5>
<b><div id="beta">Beta Distribution</div></b>
</font>
<br>

<form name="formBetaPDF">
Random variable (<i>x</i>): <input type="text" name="x" value="0.5">
<br><br>
Shape parameter <i>α</i>: <input type="text" name="alpha" value="2">
<br><br>
Shape parameter <i>β</i>: <input type="text" name="beta" value="2">
<br><br>
<input type="button" name="buttonBetaPDF" value="PDF" style="width:100px;" onClick="runBetaPDF()">
<br><br>
Probability density <i>f</i>(<i>x</i>):
<br>
<textarea id="textBetaOutPDF" name="textBetaOutPDF" rows="1" cols="30" style="background:cyan;font-family:Times New Roman;font-size:11pt;color:black" wrap="off" readonly></textarea>
</form>

<form name="formBetaCDF">
Random variable (<i>x</i>): <input type="text" name="x" value="0.5">
<br><br>
Shape parameter <i>α</i>: <input type="text" name="alpha" value="2">
<br><br>
```

```
Shape parameter <i>β</i>: <input type="text" name="beta" value="2">
<br><br>
<input type="button" name="buttonBetaCDF" value="CDF" style="width:100px;" onClick="runBetaCDF()">
<br><br>
Cumulative probability <i>F</i>(<i>x</i>):
<br>
<textarea id="textBetaOutCDF" name="textBetaOutCDF" rows="1" cols="30" style="background:cyan;font-family:Times New Roman;font-size:11pt;color:black" wrap="off" readonly></textarea>
</form>

<form name="formBetaQuantile">
Probability (<i>p</i>): <input type="text" name="p" value="0.95">
<br><br>
Shape parameter <i>α</i>: <input type="text" name="alpha" value="2">
<br><br>
Shape parameter <i>β</i>: <input type="text" name="beta" value="2">
<br><br>
<input type="button" name="buttonBetaQuantile" value="Quantile" style="width:100px;" onClick="runBetaQuantile()">
<br><br>
Quantile (<i>x</i>):
<br>
<textarea id="textBetaOutQuantile" name="textBetaOutQuantile" rows="1" cols="30" style="background:cyan;font-family:Times New Roman;font-size:11pt;color:black" wrap="off" readonly></textarea>
</form>

<br><br>
<a href="#pagehead" style="background-color:lightgray;border-style:solid;border-color:silver">   Back to Top   </a>
<br><br>
<hr color=maroon size="2px"><br>

<!-- Uniform Distribution -->
<font face="Times New Roman" size=5>
<b><div id="uniform">Uniform Distribution</div></b>
</font>
<br>

<form name="formUniformPDF">
Random variable (<i>x</i>): <input type="text" name="x" value="0.5">
<br><br>
Lower bound (<i>a</i>): <input type="text" name="a" value="0">
<br><br>
Upper bound (<i>b</i>): <input type="text" name="b" value="1">
<br><br>
<input type="button" name="buttonUniformPDF" value="PDF" style="width:100px;" onClick="runUniformPDF()">
```

```
<br><br>
Probability density <i>f</i>(<i>x</i>):
<br>
<textarea id="textUniformOutPDF" name="textUniformOutPDF" rows="1" cols="30" style="background:cyan;font-family:Times New Roman;font-size:11pt;color:black" wrap="off" readonly></textarea>
</form>

<form name="formUniformCDF">
Random variable (<i>x</i>): <input type="text" name="x" value="0.5">
<br><br>
Lower bound (<i>a</i>): <input type="text" name="a" value="0">
<br><br>
Upper bound (<i>b</i>): <input type="text" name="b" value="1">
<br><br>
<input type="button" name="buttonUniformCDF" value="CDF" style="width:100px;" onClick="runUniformCDF()">
<br><br>
Cumulative probability <i>F</i>(<i>x</i>):
<br>
<textarea id="textUniformOutCDF" name="textUniformOutCDF" rows="1" cols="30" style="background:cyan;font-family:Times New Roman;font-size:11pt;color:black" wrap="off" readonly></textarea>
</form>

<form name="formUniformQuantile">
Probability (<i>p</i>): <input type="text" name="p" value="0.95">
<br><br>
Lower bound (<i>a</i>): <input type="text" name="a" value="0">
<br><br>
Upper bound (<i>b</i>): <input type="text" name="b" value="1">
<br><br>
<input type="button" name="buttonUniformQuantile" value="Quantile" style="width:100px;" onClick="runUniformQuantile()">
<br><br>
Quantile (<i>x</i>):
<br>
<textarea id="textUniformOutQuantile" name="textUniformOutQuantile" rows="1" cols="30" style="background:cyan;font-family:Times New Roman;font-size:11pt;color:black" wrap="off" readonly></textarea>
</form>

<br><br>
<a href="#pagehead" style="background-color:lightgray;border-style:solid;border-color:silver">   Back to Top   </a>
<br><br>
<hr color=maroon size="2px"><br>

<!-- Weibull Distribution -->
```

```
<font face="Times New Roman" size=5>
<b><div id="weibull">Weibull Distribution</div></b>
</font>
<br>

<form name="formWeibullPDF">
Random variable (<i>x</i>): <input type="text" name="x" value="1">
<br><br>
Shape parameter (<i>k</i>): <input type="text" name="shape" value="2">
<br><br>
Scale parameter (<i>λ</i>): <input type="text" name="scale" value="1">
<br><br>
<input type="button" name="buttonWeibullPDF" value="PDF" style="width:100px;" onClick="runWeibullPDF()">
<br><br>
Probability density <i>f</i>(<i>x</i>):
<br>
<textarea id="textWeibullOutPDF" name="textWeibullOutPDF" rows="1" cols="30" style="background:cyan;font-family:Times
New Roman;font-size:11pt;color:black" wrap="off" readonly></textarea>
</form>

<form name="formWeibullCDF">
Random variable (<i>x</i>): <input type="text" name="x" value="1">
<br><br>
Shape parameter (<i>k</i>): <input type="text" name="shape" value="2">
<br><br>
Scale parameter (<i>λ</i>): <input type="text" name="scale" value="1">
<br><br>
<input type="button" name="buttonWeibullCDF" value="CDF" style="width:100px;" onClick="runWeibullCDF()">
<br><br>
Cumulative probability <i>F</i>(<i>x</i>):
<br>
<textarea id="textWeibullOutCDF" name="textWeibullOutCDF" rows="1" cols="30"
style="background:cyan;font-family:Times New Roman;font-size:11pt;color:black" wrap="off" readonly></textarea>
</form>

<form name="formWeibullQuantile">
Probability (<i>p</i>): <input type="text" name="p" value="0.95">
<br><br>
Shape parameter (<i>k</i>): <input type="text" name="shape" value="2">
<br><br>
Scale parameter (<i>λ</i>): <input type="text" name="scale" value="1">
<br><br>
<input type="button" name="buttonWeibullQuantile" value="Quantile" style="width:100px;" onClick="runWeibullQuantile()">
<br><br>
Quantile (<i>x</i>):
```

```
<br>
<textarea id="textWeibullOutQuantile" name="textWeibullOutQuantile" rows="1" cols="30"
style="background:cyan;font-family:Times New Roman;font-size:11pt;color:black" wrap="off" readonly></textarea>
</form>

<br><br>
<a href="#pagehead" style="background-color:lightgray;border-style:solid;border-color:silver">   Back to
Top   </a>
<br><br>
<hr color=maroon size="2px"><br>

<!-- Lognormal Distribution -->
<font face="Times New Roman" size=5>
<b><div id="lognormal">Lognormal Distribution</div></b>
</font>
<br>

<form name="formLognormalPDF">
Random variable (<i>x</i>): <input type="text" name="x" value="2">
<br><br>
Location parameter (<i>μ</i>): <input type="text" name="mu" value="0">
<br><br>
Scale parameter (<i>σ</i>): <input type="text" name="sigma" value="1">
<br><br>
<input type="button" name="buttonLognormalPDF" value="PDF" style="width:100px;" onClick="runLognormalPDF()">
<br><br>
Probability density <i>f</i>(<i>x</i>):
<br>
<textarea id="textLognormalOutPDF" name="textLognormalOutPDF" rows="1" cols="30"
style="background:cyan;font-family:Times New Roman;font-size:11pt;color:black" wrap="off" readonly></textarea>
</form>

<form name="formLognormalCDF">
Random variable (<i>x</i>): <input type="text" name="x" value="2">
<br><br>
Location parameter (<i>μ</i>): <input type="text" name="mu" value="0">
<br><br>
Scale parameter (<i>σ</i>): <input type="text" name="sigma" value="1">
<br><br>
<input type="button" name="buttonLognormalCDF" value="CDF" style="width:100px;" onClick="runLognormalCDF()">
<br><br>
Cumulative probability <i>F</i>(<i>x</i>):
<br>
<textarea id="textLognormalOutCDF" name="textLognormalOutCDF" rows="1" cols="30"
style="background:cyan;font-family:Times New Roman;font-size:11pt;color:black" wrap="off" readonly></textarea>
```

```
</form>

<form name="formLognormalQuantile">
Probability (<i>p</i>): <input type="text" name="p" value="0.95">
<br><br>
Location parameter (<i>μ</i>): <input type="text" name="mu" value="0">
<br><br>
Scale parameter (<i>σ</i>): <input type="text" name="sigma" value="1">
<br><br>
<input type="button" name="buttonLognormalQuantile" value="Quantile" style="width:100px;" onClick="runLognormalQuantile()">
<br><br>
Quantile (<i>x</i>):
<br>
<textarea id="textLognormalOutQuantile" name="textLognormalOutQuantile" rows="1" cols="30" style="background:cyan;font-family:Times New Roman;font-size:11pt;color:black" wrap="off" readonly></textarea>
</form>

<br><br>
<a href="#pagehead" style="background-color:lightgray;border-style:solid;border-color:silver">   Back to Top   </a>
<br><br>
<hr color=maroon size="2px"><br>

<!-- Cauchy Distribution -->
<font face="Times New Roman" size=5>
<b><div id="cauchy">Cauchy Distribution</div></b>
</font>
<br>

<form name="formCauchyPDF">
Random variable (<i>x</i>): <input type="text" name="x" value="0">
<br><br>
Location parameter (<i>x0</i>): <input type="text" name="location" value="0">
<br><br>
Scale parameter (<i>γ</i>): <input type="text" name="scale" value="1">
<br><br>
<input type="button" name="buttonCauchyPDF" value="PDF" style="width:100px;" onClick="runCauchyPDF()">
<br><br>
Probability density <i>f</i>(<i>x</i>):
<br>
<textarea id="textCauchyOutPDF" name="textCauchyOutPDF" rows="1" cols="30" style="background:cyan;font-family:Times New Roman;font-size:11pt;color:black" wrap="off" readonly></textarea>
</form>
```

```
<form name="formCauchyCDF">
Random variable (<i>x</i>): <input type="text" name="x" value="0">
<br><br>
Location parameter (<i>x0</i>): <input type="text" name="location" value="0">
<br><br>
Scale parameter (<i>γ</i>): <input type="text" name="scale" value="1">
<br><br>
<input type="button" name="buttonCauchyCDF" value="CDF" style="width:100px;" onClick="runCauchyCDF()">
<br><br>
Cumulative probability <i>F</i>(<i>x</i>):
<br>
<textarea id="textCauchyOutCDF" name="textCauchyOutCDF" rows="1" cols="30" style="background:cyan;font-family:Times New Roman;font-size:11pt;color:black" wrap="off" readonly></textarea>
</form>

<form name="formCauchyQuantile">
Probability (<i>p</i>): <input type="text" name="p" value="0.95">
<br><br>
Location parameter (<i>x0</i>): <input type="text" name="location" value="0">
<br><br>
Scale parameter (<i>γ</i>): <input type="text" name="scale" value="1">
<br><br>
<input type="button" name="buttonCauchyQuantile" value="Quantile" style="width:100px;" onClick="runCauchyQuantile()">
<br><br>
Quantile (<i>x</i>):
<br>
<textarea id="textCauchyOutQuantile" name="textCauchyOutQuantile" rows="1" cols="30" style="background:cyan;font-family:Times New Roman;font-size:11pt;color:black" wrap="off" readonly></textarea>
</form>

<br><br>
<a href="#pagehead" style="background-color:lightgray;border-style:solid;border-color:silver">   Back to Top   </a>
<br><br>
<hr color=maroon size="2px"><br>

<!-- Geometric Distribution -->
<font face="Times New Roman" size=5>
<b><div id="geometric">Geometric Distribution</div></b>
</font>
<br>

<form name="formGeometricPDF">
Number of failures (<i>k</i>): <input type="text" name="k" value="3">
<br><br>
```

```
Probability of success (<i>p</i>): <input type="text" name="p" value="0.3">
<br><br>
<input type="button" name="buttonGeometricPDF" value="PDF" style="width:100px;" onClick="runGeometricPDF()">
<br><br>
Probability mass <i>f</i>(<i>k</i>):
<br>
<textarea id="textGeometricOutPDF" name="textGeometricOutPDF" rows="1" cols="30"
style="background:cyan;font-family:Times New Roman;font-size:11pt;color:black" wrap="off" readonly></textarea>
</form>

<form name="formGeometricCDF">
Number of failures (<i>k</i>): <input type="text" name="k" value="3">
<br><br>
Probability of success (<i>p</i>): <input type="text" name="p" value="0.3">
<br><br>
<input type="button" name="buttonGeometricCDF" value="CDF" style="width:100px;" onClick="runGeometricCDF()">
<br><br>
Cumulative probability <i>F</i>(<i>k</i>):
<br>
<textarea id="textGeometricOutCDF" name="textGeometricOutCDF" rows="1" cols="30"
style="background:cyan;font-family:Times New Roman;font-size:11pt;color:black" wrap="off" readonly></textarea>
</form>

<form name="formGeometricQuantile">
Probability (<i>p</i>): <input type="text" name="p" value="0.95">
<br><br>
Probability of success (<i>prob</i>): <input type="text" name="prob" value="0.3">
<br><br>
<input type="button" name="buttonGeometricQuantile" value="Quantile" style="width:100px;"
onClick="runGeometricQuantile()">
<br><br>
Quantile (<i>k</i>):
<br>
<textarea id="textGeometricOutQuantile" name="textGeometricOutQuantile" rows="1" cols="30"
style="background:cyan;font-family:Times New Roman;font-size:11pt;color:black" wrap="off" readonly></textarea>
</form>

<br><br>
<a href="#pagehead" style="background-color:lightgray;border-style:solid;border-color:silver">   Back to
Top   </a>
<br><br>
<hr color=maroon size="2px"><br>

<!-- Negative Binomial Distribution -->
<font face="Times New Roman" size=5>
```

```
<b><div id="negbinomial">Negative Binomial Distribution</div></b>
</font>
<br>

<form name="formNegBinomialPDF">
Number of failures (<i>k</i>): <input type="text" name="k" value="5">
<br><br>
Number of successes (<i>r</i>): <input type="text" name="r" value="3">
<br><br>
Probability of success (<i>p</i>): <input type="text" name="p" value="0.4">
<br><br>
<input type="button" name="buttonNegBinomialPDF" value="PDF" style="width:100px;" onClick="runNegBinomialPDF()">
<br><br>
Probability mass <i>f</i>(<i>k</i>):
<br>
<textarea id="textNegBinomialOutPDF" name="textNegBinomialOutPDF" rows="1" cols="30"
style="background:cyan;font-family:Times New Roman;font-size:11pt;color:black" wrap="off" readonly></textarea>
</form>

<form name="formNegBinomialCDF">
Number of failures (<i>k</i>): <input type="text" name="k" value="5">
<br><br>
Number of successes (<i>r</i>): <input type="text" name="r" value="3">
<br><br>
Probability of success (<i>p</i>): <input type="text" name="p" value="0.4">
<br><br>
<input type="button" name="buttonNegBinomialCDF" value="CDF" style="width:100px;" onClick="runNegBinomialCDF()">
<br><br>
Cumulative probability <i>F</i>(<i>k</i>):
<br>
<textarea id="textNegBinomialOutCDF" name="textNegBinomialOutCDF" rows="1" cols="30"
style="background:cyan;font-family:Times New Roman;font-size:11pt;color:black" wrap="off" readonly></textarea>
</form>

<form name="formNegBinomialQuantile">
Probability (<i>p</i>): <input type="text" name="p" value="0.95">
<br><br>
Number of successes (<i>r</i>): <input type="text" name="r" value="3">
<br><br>
Probability of success (<i>prob</i>): <input type="text" name="prob" value="0.4">
<br><br>
<input type="button" name="buttonNegBinomialQuantile" value="Quantile" style="width:100px;"
onClick="runNegBinomialQuantile()">
<br><br>
Quantile (<i>k</i>):
```


<textarea id="textNegBinomialOutQuantile" name="textNegBinomialOutQuantile" rows="1" cols="30" style="background:cyan;font-family:Times New Roman;font-size:11pt;color:black" wrap="off" readonly></textarea>
</form>

   Back to Top

<hr color=maroon size="2px">

User manual guide:

<a
href="http://www.iaees.org/publications/journals/selforganizology/articles/2026-13(3-4)/probFunCal2-for-advanced-probability-distribution-functions.pdf">Zhang W. J. 2026. probFunCal2: A web-based calculator for calculating probability distribution functions as binomial, Poisson, exponential, gamma, beta, uniform, Weibull, lognormal, Cauchy, geometric, and negative binomial distributions. Selforganizology, 13(3-4): 26-65

   Back to Top

<hr color=maroon size="2px">

<hr>
<table border=0 cellspacing="1" cellpadding="1" width="100%">
<tr>
<td>

<font size=-1>

Translate page to:

<script>

```
function googleTranslateElementInit() {
   new google.translate.TranslateElement({pageLanguage: 'en'}, 'google_translate_element');
}
</script><script src="http://translate.google.com/translate_a/element.js?cb=googleTranslateElementInit"></script>

</font>
</td>
</tr>
</table>

</font>

</body>
</html>
```

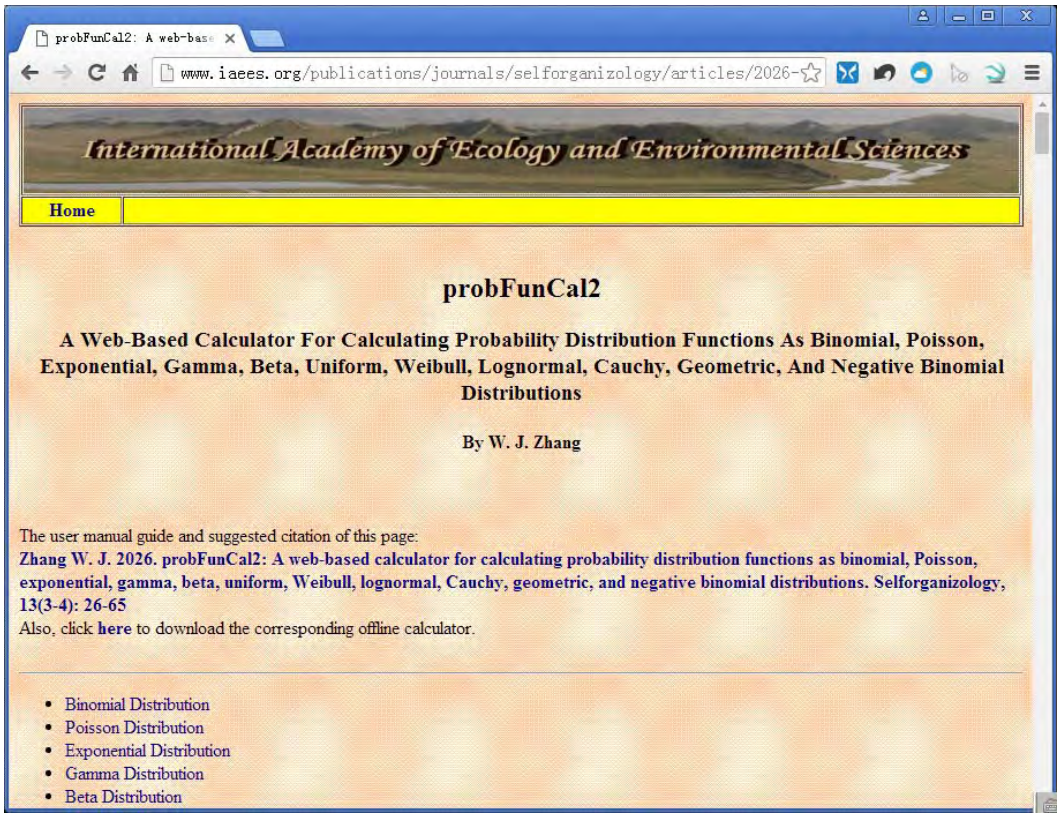

Fig. 1 probFunCal2.

5 Applications

probFunCal2 finds applications in statistical modeling, hypothesis testing, simulation studies, and educational demonstrations. The binomial and Poisson distributions are essential for count data analysis; exponential, gamma, and Weibull for survival and reliability analysis; beta for proportion modeling; lognormal for financial and environmental data; and Cauchy for robust statistics.

6 Discussion

probFunCal2 significantly extends the applicabilities and capabilities of probFunCal etc (Zhang, 2025, 2026; Zhang and Qi, 2025), providing a comprehensive toolkit for probability calculations. The inclusion of both discrete and continuous distributions, along with complete PDF/CDF/quantile functionality, makes it suitable for both theoretical and applied work. The web-based implementation ensures accessibility while maintaining computational accuracy through sophisticated numerical methods.

Future enhancements could include additional distributions (e.g., hypergeometric, Pareto, Student's *t* for non-standard cases), graphical capabilities for distribution visualization, and support for vectorized computations. Integration with data analysis workflows through export functionality would further enhance practical utility.

The development of probFunCal2 demonstrates the power of client-side JavaScript for scientific computing, challenging the notion that complex numerical algorithms require server-side processing or specialized software. By making advanced probability tools freely available through web browsers, we democratize access to essential statistical methods.

References

Liu GH, Zhang WJ. 2011. Computer generation of initial spatial distribution for cell automata. Computational Ecology and Software, 1(4): 244-248. http://www.iaees.org/publications/journals/ces/articles/2011-1(4)/computer-generation-of-initial-spatial-distrib ution.pdf

Zhang WJ. 2007. Methodology on Ecology Research. Sun Yat-sen University, Guangzhou, China

Zhang WJ. 2026. probFunCal: A webpage calculator for probability distribution functions. Selforganizology, 13(1-2): 1-25. http://www.iaees.org/publications/journals/selforganizology/articles/2026-13(1-2)/1-Zhang-Abstract.asp

Zhang WJ. 2025. probTable: A Matlab calculator for lookuping probability tables. Selforganizology, 12(3-4): 21-29. http://www.iaees.org/publications/journals/selforganizology/articles/2025-12(3-4)/1-Zhang-Abstract.asp

Zhang WJ, Qi YH. 2025. probDistriCal: The online calculator for probability distributions. Computational Ecology and Software, 15(2): 57-77. http://www.iaees.org/publications/journals/ces/articles/2025-15(2)/3-Zhang-Abstract.asp