Article

probFunCal: A webpage calculator for probability distribution functions

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

Received 11 April 2025; *Accepted 22 April 2025*; *Published online 25 April 2025*; *Published 1 June 2026*

Abstract
In present study, a calculator, probFunCal, was developed for calculating probability distributions. In the calculator, probability distributions as normal distribution, *t* distribution, χ^2 distribution, *F* distribution, etc., were available for use. Probability density function, cumulative distribution function, and inverse cumulative distribution function of these probability distributions can be calculated. The calculator is web browser based that includes both online and offline versions and can be used on various computing devices (PCs, iPads, smartphones, etc.), operating systems (Windows, Mac, Android, Harmony, etc.) and web browsers (Chrome, Firefox, etc).

Keywords calculator; web application; JavaScript; probability distribution; probability density function; cumulative distribution function; inverse cumulative distribution function; normal distribution; *t* distribution; *F* distribution; χ^2 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

In an earlier study, the Matlab calculator, probTable, was developed (Zhang, 2025). In probTable, probability distributions as normal distribution, *t* distribution, *F* distribution, χ^2 distribution, Weibull distribution, lognormal distribution, binomial distribution, Poission distribution, negative binomial distribution, and uniform distribution, etc., were available for use, and probability density function, cumulative distribution function, and inverse cumulative distribution function of these probability distributions can be calculated. However, it was the standalone software based on Matlab. 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.

In present study, I developed a calculator, probFunCal, for calculating probability distribution functions. In the calculator, probability distributions as normal distribution, *t* distribution, χ^2 distribution, *F* distribution, etc., were available for use. Probability density function, cumulative distribution function, and inverse cumulative distribution function of these probability distributions can be calculated. The calculator is web browser based

which includes both online and offline versions and can be used on various computing devices and web browsers.

2 Probability Distributions

There is a random variable X, and x is a value of X. Assume that the probability density function of X is $f(x)$. The often used probability distributions are described as follows (Zhang, 2025; Zhang and Qi, 2025).

(1) Normal distribution $N(\mu, \sigma)$

$$f(x)=\frac{1}{\sigma\sqrt{2\pi}}e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

where μ: mean, σ: standard deviation, $\sigma>0$. In the standard normal distribution $N(0, 1)$, $\mu=0$, $\sigma=1$.

(2) t distribution $t(n)$

$$f(x)=\frac{\Gamma(\frac{n+1}{2})}{\sqrt{n\pi}\Gamma(\frac{n}{2})}(1+\frac{x^2}{n})^{-\frac{n+1}{2}}$$

where n: the positive integer, i.e., degree of freedom.

(3) χ^2 distribution $\chi^2(n)$

$$f(x)=\frac{1}{2^{\frac{n}{2}}\Gamma(\frac{n}{2})}x^{\frac{n}{2}-1}e^{-\frac{x}{2}}$$

where n: the positive integer, i.e., the degree of freedom.

(4) F distribution $F(m,n)$

$$f(x)=\frac{\Gamma(\frac{m+n}{2})m^{\frac{m}{2}}n^{\frac{n}{2}}}{\Gamma(\frac{m}{2})\Gamma(\frac{n}{2})}\frac{x^{\frac{m}{2}-1}}{(mx+n)^{\frac{m+n}{2}}} \quad , x>0$$

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

where m, n: the positive integers,i.e., 1st and 2nd degree of freedoms.

Known x, we can achieve the value of the probability density function $f(x)$. The probability that a random variable X takes the value less than x is $p= F(x) =P(X<x)=\int_{-\infty}^{x} f(x)\,dx$, where $F(x)$ is the cumulative distribution function, i.e., the probability p that the random variable X takes a value less than x. Known the probability p that the random variable X takes a value less than x, i.e., the cumulative distribution function value $F(x)$, we can achieve x, which is the value of the inverse cumulative distribution function (Zhang, 2025).

3 Full Codes

Full codes of the calculator are as follows:

```
<!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="probFunCal: A webpage calculator for probability distribution functions" />
<meta name="keywords" content="probability distribution functions, webpage calculator" />
<meta name="author" content="W. J. Zhang" />
<link href="../../../../../style.css" rel="stylesheet" media="screen" type="text/css">
<title>probFunCal: A webpage calculator for probability distribution functions</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">

// Normal probability density function (PDF)
function normalPDF(x, mean, sd) {
if (sd <= 0) {
      throw new Error("Standard error must be greater than 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);
}

// Probability density function (PDF) of t-distribution
function tPDF(t, df) {
      const gamma = (a) => {
            // Approximate calculation of the gamma function
            const p = [0.99999999999980993, 676.5203681218851, -1259.1392167224028,
                        771.32342877765313, -176.61502916214059, 12.507343278686905,
                              -0.13857109526572012, 9.9843695780195716e-6, 1.5056327351493116e-7];
            let g = 7;
            if (a < 0.5) return Math.PI / (Math.sin(Math.PI * a) * gamma(1 - a));
            a -= 1;
            let x = p[0];
            for (let i = 1; i < p.length; i++) {
                  x += p[i] / (a + i);
```

```
            }
            const t = a + g + 0.5;
            return Math.sqrt(2 * Math.PI) * Math.pow(t, a + 0.5) * Math.exp(-t) * x;
        };
        const numerator = gamma((df + 1) / 2);
        const denominator = Math.sqrt(df * Math.PI) * gamma(df / 2);
        return numerator / denominator * Math.pow(1 + (t * t) / df, -(df + 1) / 2);
}

// Probability density function (PDF) of the chi-square distribution
function chisqPDF(x, df) {
        if (x <= 0) return 0;
        return Math.pow(x, df/2 - 1) * Math.exp(-x/2) / (Math.pow(2, df/2) * gamma(df/2));
}

// Probability density function (PDF) of the F distribution
function fPDF(x, d1, d2) {
        if (x < 0 || d1 <= 0 || d2 <= 0) {
            return 0;
        }
        if (x === 0) {
            if (d1 < 2) return Infinity;
            if (d1 === 2) return 1;
            return 0;
        }
        const betaPart = betaFunction(d1 / 2, d2 / 2);
        const numerator = Math.pow(d1 * x, d1) * Math.pow(d2, d2);
        const denominator = Math.pow(d1 * x + d2, d1 + d2);
        const mainPart = Math.sqrt(numerator / denominator);
        const pdf = mainPart / (x * betaPart);
        return pdf;
}

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);
}

// Cumulative distribution function (CDF) of the standard normal distribution
function normalCDF(z) {
    // function pnorm(x)
    // function normCDF(x)
    // function normCdf(x)
    // Hastings approximation
    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);
}

/*
// Cumulative distribution function (CDF) of the standard normal distribution
function normalCDF(x) {
    // Standard normal cumulative distribution function
    return jStat.normal.cdf(x, 0, 1);
}
*/

// Cumulative distribution function (CDF) of the t-distribution
function tCDF(x, df) {
    if (df <= 0) {
        throw new Error("Degrees of freedom must be greater than 0");
    }
    // Define the probability density function (PDF) of the t-distribution
    // Simpson's rule for numerical integration
```

```
    function simpson(f, a, b, n) {
        if (n % 2 !== 0) throw new Error("n must be even");
        const h = (b - a) / n;
        let sum = f(a) + f(b);
        for (let i = 1; i < n; i++) {
            const x = a + i * h;
            sum += f(x) * (i % 2 == 0 ? 2 : 4);
        }
        return sum * h / 3;
    }
    // Compute the CDF of the t-distribution
    const lowerLimit = -100; // Approximately negative infinity
    const upperLimit = x;
    const n = 1000; // Number of divisions of the integration interval
    return simpson(t => tPDF(t, df), lowerLimit, upperLimit, n);
}

// Cumulative distribution function (CDF) of the chi-squared distribution
function chisqCDF(x, df, lowerTail = true) {
    if (x <= 0) return lowerTail ? 0 : 1;
    // Wilson-Hilferty transformation
    const mu = 1 - 2/(9 * df);
    const sigma = Math.sqrt(2/(9 * df));
    const z = (Math.pow(x/df, 1/3) - mu) / sigma;
    const p = normalCDF(z);
    return lowerTail ? p : 1 - p;
}

/*
// Cumulative distribution function (CDF) of the chi-squared distribution
function chisqCDF(x, df, lowerTail = true) {
    if (lowerTail) {
        return jStat.chisquare.cdf(x, df);
    } else {
        return 1 - jStat.chisquare.cdf(x, df);
    }
}
*/

// Cumulative distribution function (CDF) of F distribution
function fCDF(x, d1, d2) {
    if (x < 0) return 0;
    if (x === 0) return 0;
    if (!isFinite(x)) return 1;
    if (d1 <= 0 || d2 <= 0) return NaN;
```

```
    const y = d1 * x / (d1 * x + d2);
    return regularizedBeta(y, d1 / 2, d2 / 2);
}

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) {
            console.warn('Maximum iterations achieved');
            break;
        }
    } while (Math.abs(f - 1) >= eps);
    return f;
```

```
}

// Inverse normal CDF approximation
function normalQuantile(p) {
    // Acklam's inverse normal CDF approximation
    // precision 1.15e-9
    if (p <= 0 || p >= 1) {
        throw new Error("p must be in (0, 1)");
    }
    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) {
        // Rational approximation for lower region
        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) {
        // Rational approximation for upper region
        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 {
        // Rational approximation for central region
        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);
        }
}

/*
// Inverse normal CDF approximation
function normalQuantile(p) {
    // Inverse of the standard normal cumulative distribution function
    // This can be implemented using an approximation or a library like jStat
    return jStat.normal.inv(p, 0, 1);
}
*/

// Inverse t-distribution CDF approximation
function tQuantile(p, df) {
    if (p <= 0 || p >= 1 || df <= 0) {
        throw new Error("Invalid input: p must be between 0 and 1, and df must be positive.");
    }
    // Approximate quantiles of a t-distribution using Hill's algorithm; inverse t-distribution CDF approximation
    function hillApprox(p, df) {
        const a = (1 / (df - 0.5)) - 1;
        const b = 48 / (a * a);
        const c = ((20700 * a / b - 98) * a - 16) * a + 96.36;
        const d = ((94.5 / (b + c) - 3) / b + 1) * Math.sqrt(a * Math.PI / 2) * df;
        const x = d * p;
        const y = Math.pow(x, 2 / df);
        if (y > 0.05 + a) {
            return x;
        } else {
            // Further optimization using the Newton-Raphson method
            let q = p - 0.5;
            let r = q * q;
            let z = q * (((-25.44106049637 * r + 41.39119773534) * r - 18.61500062529) * r + 2.50662823884) /
                    ((((3.13082909833 * r - 21.06224101826) * r + 23.08336743743) * r - 8.4735109309) * r + 1);
            z = z * Math.sqrt(df / (df + z * z));
            return z;
        }
    }
    return hillApprox(p, df);
}

// Gamma function
function gamma(x) {
    // Lanczos approximation
```

```
    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 qchisq_WilsonHilferty(p, df) {
    const z = normalQuantile(p);    // Standard normal distribution quantile
    const h = 2/(9 * df);
    const approx = df * Math.pow(z * Math.sqrt(h) + 1 - h, 3);
    return Math.max(0, approx);
}

function qchisq_NewtonRaphson(p, df, maxIter = 100, tol = 1e-8) {
    // Initial guess (using the Wilson-Hilferty approximation as a starting point)
    let x = Math.max(0.1, qchisq_WilsonHilferty(p, df));
    for (let i = 0; i < maxIter; i++) {
        const fx = chisqCDF(x, df) - p;
        if (Math.abs(fx) < tol) return x;
        // PDF of the chi-square distribution (for derivatives)
        const fpx = chisqPDF(x, df);
        if (fpx == 0) break;    // Avoid division by zero
        x = x - fx / fpx;
        x = Math.max(x, 0);    // Keep non-negative
    }
    return x;
}

// Quantile function (inverse CDF) of the chi-squared distribution
function chisqQuantile(p, df) {
    if (p < 0 || p > 1) return NaN;
    if (df <= 0) return NaN;
    // Use Wilson-Hilferty approximation for large degrees of freedom
    if (df > 30) {
```

```
            return qchisq_WilsonHilferty(p, df);
        }
        // Use Newton-Raphson iteration for small degrees of freedom
        return qchisq_NewtonRaphson(p, df);
}

// Quantile function (inverse CDF) of F distribution
function fQuantile(p, d1, d2, tol = 1e-12, maxIter = 1000) {
        if (p < 0 || p > 1 || d1 <= 0 || d2 <= 0) {
            return NaN;
        }
        if (p === 0) return 0;
        if (p === 1) return Infinity;
        const y = inverseRegularizedBeta(p, d1/2, d2/2, tol, maxIter);
        return (y * d2) / (d1 * (1 - y));
}

function inverseRegularizedBeta(p, a, b, tol, maxIter) {
        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);
        if (iter >= maxIter) {
            console.warn('Maximum iterations achieved');
        }
        return x;
}

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

```
}

function runnormpdf() {
var miu=document.formnormpdf.miu.value;
var sigma=document.formnormpdf.sigma.value;
var x=document.formnormpdf.x.value;
var fx=normalPDF(x,miu,sigma);
var strs=String(fx);
document.formnormpdf.textnormoutpdf.value=strs;
}

function runnormcdf() {
var x=document.formnormcdf.x.value;
var p=normalCDF(x);
var strs=String(p);
document.formnormcdf.textnormoutcdf.value=strs;
}

function runnorminv() {
var p=document.formnorminv.p.value;
var x=normalQuantile(p);
var strs=String(x);
document.formnorminv.textnormoutinv.value=strs;
}

function runtpdf() {
var x=document.formtpdf.x.value;
var n=parseInt(document.formtpdf.n.value);
var fx=tPDF(x,n);
var strs=String(fx);
document.formtpdf.texttoutpdf.value=strs;
}

function runtcdf() {
var x=document.formtcdf.x.value;
var n=parseInt(document.formtcdf.n.value);
var p=tCDF(x,n);
var strs=String(p);
document.formtcdf.texttoutcdf.value=strs;
}

function runtinv() {
var p=document.formtinv.p.value;
var n=parseInt(document.formtinv.n.value);
var x=tQuantile(p,n);
```

```
var strs=String(x);
document.formtinv.texttoutinv.value=strs;
}

function runchipdf() {
var x=document.formchipdf.x.value;
var n=parseInt(document.formchipdf.n.value);
var fx=chisqPDF(x, n);
var strs=String(fx);
document.formchipdf.textchioutpdf.value=strs;
}

function runchicdf() {
var x=document.formchicdf.x.value;
var n=parseInt(document.formchicdf.n.value);
var p=chisqCDF(x,n);
var strs=String(p);
document.formchicdf.textchioutcdf.value=strs;
}

function runchiinv() {
var p=document.formchiinv.p.value;
var n=parseInt(document.formchiinv.n.value);
var x=chisqQuantile(p,n);
var strs=String(x);
document.formchiinv.textchioutinv.value=strs;
}

function runfpdf() {
var x=document.formfpdf.x.value;
var m=parseInt(document.formfpdf.m.value);
var n=parseInt(document.formfpdf.n.value);
var fx=fPDF(x,m,n);
var strs=String(fx);
document.formfpdf.textfoutpdf.value=strs;
}

function runfcdf() {
var x=document.formfcdf.x.value;
var m=parseInt(document.formfcdf.m.value);
var n=parseInt(document.formfcdf.n.value);
var p=fCDF(x,m,n);
var strs=String(p);
document.formfcdf.textfoutcdf.value=strs;
}
```

```
function runfinv() {
var p=document.formfinv.p.value;
var m=parseInt(document.formfinv.m.value);
var n=parseInt(document.formfinv.n.value);
var x=fQuantile(p,m,n);
var strs=String(x);
document.formfinv.textfoutinv.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 bgcolor=yellow>
<th width="10%"><a
href="http://www.iaees.org/publications/journals/selforganizology/articles/2026-13(1-2)/1-Zhang-Abstract.asp">Home</a></th>
<th width="90%"></th>
</table>

<font face="Times New Roman">

<br>
<div id="pagehead"></div>
<div align=center><b><h2>probFunCal</h2></b></div>
<div align=center><b><h3>probFunCal: A webpage calculator for probability distribution functions</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(1-2)/probFunCal-for-probability-distribution-functions.pdf">Zhang W. J. 2026. probFunCal: A webpage calculator for probability distribution functions. Selforganizology, 13(1-2): 1-25</a></b>
<br>Also, click <a
href="http://www.iaees.org/publications/journals/selforganizology/articles/2026-13(1-2)/1-Zhang-Abstract.asp"><b>here</b></a> to download the corresponding offline calculator.
<br><br>
<hr>

<ul>
<li><a href="#npdf">Normal Distribution: Density Function</a></li>
<li><a href="#ncdf">Standard Normal Distribution: Cumulative Distribution Function</a></li>
```

```
<li><a href="#ninv">Standard Normal Distribution: Inverse Cumulative Distribution Function</a></li>

<li><a href="#tpdf"><i>t</i> Distribution: Density Function</a></li>
<li><a href="#tcdf"><i>t</i> Distribution: Cumulative Distribution Function</a></li>
<li><a href="#tinv"><i>t</i> Distribution: Inverse Cumulative Distribution Function</a></li>

<li><a href="#chipdf"><i>χ</i><sup>2</sup> Distribution: Density Function</a></li>
<li><a href="#chicdf"><i>χ</i><sup>2</sup> Distribution: Cumulative Distribution Function</a></li>
<li><a href="#chiinv"><i>χ</i><sup>2</sup> Distribution: Inverse Cumulative Distribution Function</a></li>

<li><a href="#fpdf"><i>F</i> Distribution: Density Function</a></li>
<li><a href="#fcdf"><i>F</i> Distribution: Cumulative Distribution Function</a></li>
<li><a href="#finv"><i>F</i> Distribution: Inverse Cumulative Distribution Function</a></li>

</ul>
<hr color=maroon size="2px">
<br>

<font face="Times New Roman" size=5>
<b><div id="npdf">Normal Distribution: Density Function</div></b>
</font>
<br>

<form name="formnormpdf">
Mean (<i>μ</i>): <input type="text" name="miu" value="">
<br><br>
Standard deviation (<i>σ</i>): <input type="text" name="sigma" value="">
<br><br>
Random variable (<i>x</i>): <input type="text" name="x" value="">
<br><br>
<input type="button" name="buttonnormpdf" value="Run" style="width:100px;" onClick="runnormpdf()">
<br><br>
Probability density <i>f</i>(<i>x</i>):
<br>
<textarea id="textnormoutpdf" name="textnormoutpdf" rows="1" cols="30" style="background:cyan;font-family:Times New
Roman;font-size:11pt;color:black" wrap="off" value="      " readonly>
</textarea>
<br>
<br>
</form>
<font face="Times New Roman" size=1 color=green>
User manual guide:
<br><a
href="http://www.iaees.org/publications/journals/selforganizology/articles/2026-13(1-2)/probFunCal-for-probability-distribution-
functions.pdf">Zhang W. J. 2026. probFunCal: A webpage calculator for probability distribution functions. Selforganizology,
```

```
13(1-2): 1-25</a></b>
</font>
<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>

<font face="Times New Roman" size=5>
<b><div id="ncdf">Standard Normal Distribution: Cumulative Distribution Function</div></b>
</font>
<br>

<form name="formnormcdf">
Standard random variable (<i>x</i>): <input type="text" name="x" value="">
<br><br>
<input type="button" name="buttonnormcdf" value="Run" style="width:100px;" onClick="runnormcdf()">
<br><br>
Cumulative distribution (probability <i>p</i>) ∫<i>f</i>(<i>x</i>)<i>dx</i>:
<br>
<textarea id="textnormoutcdf" name="textnormoutcdf" rows="1" cols="30" style="background:cyan;font-family:Times New
Roman;font-size:11pt;color:black" wrap="off" value="      " readonly>
</textarea>
<br>
<br>
</form>
<font face="Times New Roman" size=1 color=green>
User manual guide:
<br><a
href="http://www.iaees.org/publications/journals/selforganizology/articles/2026-13(1-2)/probFunCal-for-probability-distribution-
functions.pdf">Zhang W. J. 2026. probFunCal: A webpage calculator for probability distribution functions. Selforganizology,
13(1-2): 1-25</a></b>
</font>
<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>

<font face="Times New Roman" size=5>
<b><div id="ninv">Standard Normal Distribution: Inverse Cumulative Distribution Function</div></b>
</font>
<br>

<form name="formnorminv">
```

```
Probability (<i>p</i>): <input type="text" name="p" value="">
<br><br>
<input type="button" name="buttonnorminv" value="Run" style="width:100px;" onClick="runnorminv()">
<br><br>
Inverse cumulative distribution (<i>x</i>):
<br>
<textarea id="textnormoutinv" name="textnormoutinv" rows="1" cols="30" style="background:cyan;font-family:Times New Roman;font-size:11pt;color:black" wrap="off" value="      " readonly>
</textarea>
<br>
<br>
</form>
<font face="Times New Roman" size=1 color=green>
User manual guide:
<br><a
href="http://www.iaees.org/publications/journals/selforganizology/articles/2026-13(1-2)/probFunCal-for-probability-distribution-functions.pdf">Zhang W. J. 2026. probFunCal: A webpage calculator for probability distribution functions. Selforganizology, 13(1-2): 1-25</a></b>
</font>
<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>

<font face="Times New Roman" size=5>
<b><div id="tpdf"><i>t</i> Distribution: Density Function</div></b>
</font>
<br>

<form name="formtpdf">
Degree of freedom (<i>n</i>): <input type="text" name="n" value="">
<br><br>
Random variable (<i>x</i>): <input type="text" name="x" value="">
<br><br>
<input type="button" name="buttontpdf" value="Run" style="width:100px;" onClick="runtpdf()">
<br><br>
Probability density <i>f</i>(<i>x</i>):
<br>
<textarea id="texttoutpdf" name="texttoutpdf" rows="1" cols="30" style="background:cyan;font-family:Times New Roman;font-size:11pt;color:black" wrap="off" value="      " readonly>
</textarea>
<br><br>
</form>
<font face="Times New Roman" size=1 color=green>
```

```
User manual guide:
<br><a
href="http://www.iaees.org/publications/journals/selforganizology/articles/2026-13(1-2)/probFunCal-for-probability-distribution-functions.pdf">Zhang W. J. 2026. probFunCal: A webpage calculator for probability distribution functions. Selforganizology, 13(1-2): 1-25</a></b>
</font>
<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>

<font face="Times New Roman" size=5>
<b><div id="tcdf"><i>t</i> Distribution: Cumulative Distribution Function</div></b>
</font>
<br>

<form name="formtcdf">
Degree of freedom (<i>n</i>): <input type="text" name="n" value="">
<br><br>
Random variable (<i>x</i>): <input type="text" name="x" value="">
<br><br>
<input type="button" name="buttontcdf" value="Run" style="width:100px;" onClick="runtcdf()">
<br><br>
Cumulative distribution (probability <i>p</i>) ∫<i>f</i>(<i>x</i>)<i>dx</i>:
<br>
<textarea id="texttoutcdf" name="texttoutcdf" rows="1" cols="30" style="background:cyan;font-family:Times New Roman;font-size:11pt;color:black" wrap="off" value="      " readonly>
</textarea>
<br><br>
</form>
<font face="Times New Roman" size=1 color=green>
User manual guide:
<br><a
href="http://www.iaees.org/publications/journals/selforganizology/articles/2026-13(1-2)/probFunCal-for-probability-distribution-functions.pdf">Zhang W. J. 2026. probFunCal: A webpage calculator for probability distribution functions. Selforganizology, 13(1-2): 1-25</a></b>
</font>
<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>

<font face="Times New Roman" size=5>
```

```
<b><div id="tinv"><i>t</i> Distribution: Inverse Cumulative Distribution Function</div></b>
</font>
<br>

<form name="formtinv">
Degree of freedom (<i>n</i>): <input type="text" name="n" value="">
<br><br>
Probability (<i>p</i>): <input type="text" name="p" value="">
<br><br>
<input type="button" name="buttontinv" value="Run" style="width:100px;" onClick="runtinv()">
<br><br>
Inverse cumulative distribution (<i>x</i>):
<br>
<textarea id="texttoutinv" name="texttoutinv" rows="1" cols="30" style="background:cyan;font-family:Times New
Roman;font-size:11pt;color:black" wrap="off" value="      " readonly>
</textarea>
<br><br>
</form>
<font face="Times New Roman" size=1 color=green>
User manual guide:
<br><a
href="http://www.iaees.org/publications/journals/selforganizology/articles/2026-13(1-2)/probFunCal-for-probability-distribution-
functions.pdf">Zhang W. J. 2026. probFunCal: A webpage calculator for probability distribution functions. Selforganizology,
13(1-2): 1-25</a></b>
</font>
<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>

<font face="Times New Roman" size=5>
<b><div id="chipdf"><i>χ</i><sup>2</sup> Distribution: Density Function</div></b>
</font>
<br>

<form name="formchipdf">
Degree of freedom (<i>n</i>): <input type="text" name="n" value="">
<br><br>
Random variable (<i>x</i>): <input type="text" name="x" value="">
<br><br>
<input type="button" name="buttonchipdf" value="Run" style="width:100px;" onClick="runchipdf()">
<br><br>
Probability density <i>f</i>(<i>x</i>):
<br>
```

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

</form>

User manual guide:

<a
href="http://www.iaees.org/publications/journals/selforganizology/articles/2026-13(1-2)/probFunCal-for-probability-distribution-functions.pdf">Zhang W. J. 2026. probFunCal: A webpage calculator for probability distribution functions. Selforganizology, 13(1-2): 1-25

   Back to Top

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

χ² Distribution: Cumulative Distribution Function

<form name="formchicdf">
Degree of freedom (n): <input type="text" name="n" value="">

Random variable (x): <input type="text" name="x" value="">

<input type="button" name="buttonchicdf" value="Run" style="width:100px;" onClick="runchicdf()">

Cumulative distribution (probability p) ∫f(x)dx:

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

</form>

User manual guide:

<a
href="http://www.iaees.org/publications/journals/selforganizology/articles/2026-13(1-2)/probFunCal-for-probability-distribution-functions.pdf">Zhang W. J. 2026. probFunCal: A webpage calculator for probability distribution functions. Selforganizology, 13(1-2): 1-25

   Back to Top

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

χ² Distribution: Inverse Cumulative Distribution Function

<form name="formchiinv">
Degree of freedom (n): <input type="text" name="n" value="">

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

<input type="button" name="buttonchiinv" value="Run" style="width:100px;" onClick="runchiinv()">

Inverse cumulative distribution (x):

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

</form>

User manual guide:

<a
href="http://www.iaees.org/publications/journals/selforganizology/articles/2026-13(1-2)/probFunCal-for-probability-distribution-functions.pdf">Zhang W. J. 2026. probFunCal: A webpage calculator for probability distribution functions. Selforganizology, 13(1-2): 1-25

   Back to Top

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

F Distribution: Density Function

<form name="formfpdf">
1st degree of freedom (m): <input type="text" name="m" value="">

```
<br><br>
2nd degree of freedom (<i>n</i>): <input type="text" name="n" value="">
<br><br>
Random variable (<i>x</i>): <input type="text" name="x" value="">
<br><br>
<input type="button" name="buttonfpdf" value="Run" style="width:100px;" onClick="runfpdf()">
<br><br>
Probability density <i>f</i>(<i>x</i>):
<br>
<textarea id="textfoutpdf" name="textfoutpdf" rows="1" cols="30" style="background:cyan;font-family:Times New
Roman;font-size:11pt;color:black" wrap="off" value="      " readonly>
</textarea>
<br><br>
</form>
<font face="'Times New Roman" size=1 color=green>
User manual guide:
<br><a
href="http://www.iaees.org/publications/journals/selforganizology/articles/2026-13(1-2)/probFunCal-for-probability-distribution-
functions.pdf">Zhang W. J. 2026. probFunCal: A webpage calculator for probability distribution functions. Selforganizology,
13(1-2): 1-25</a></b>
</font>
<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>

<font face="Times New Roman" size=5>
<b><div id="fcdf"><i>F</i> Distribution: Cumulative Distribution Function</div></b>
</font>
<br>

<form name="formfcdf">
1st degree of freedom (<i>m</i>): <input type="text" name="m" value="">
<br><br>
2nd degree of freedom (<i>n</i>): <input type="text" name="n" value="">
<br><br>
Random variable (<i>x</i>): <input type="text" name="x" value="">
<br><br>
<input type="button" name="buttonfcdf" value="Run" style="width:100px;" onClick="runfcdf()">
<br><br>
Cumulative distribution (probability <i>p</i>) ∫<i>f</i>(<i>x</i>)<i>dx</i>:
<br>
<textarea id="textfoutcdf" name="textfoutcdf" rows="1" cols="30" style="background:cyan;font-family:Times New
Roman;font-size:11pt;color:black" wrap="off" value="      " readonly>
```

```
</textarea>
<br><br>
</form>
<font face="Times New Roman" size=1 color=green>
User manual guide:
<br><a
href="http://www.iaees.org/publications/journals/selforganizology/articles/2026-13(1-2)/probFunCal-for-probability-distribution-functions.pdf">Zhang W. J. 2026. probFunCal: A webpage calculator for probability distribution functions. Selforganizology, 13(1-2): 1-25</a></b>
</font>
<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>

<font face="Times New Roman" size=5>
<b><div id="finv"><i>F</i> Distribution: Inverse Cumulative Distribution Function</div></b>
</font>
<br>

<form name="formfinv">
1st degree of freedom (<i>m</i>): <input type="text" name="m" value="">
<br><br>
2nd degree of freedom (<i>n</i>): <input type="text" name="n" value="">
<br><br>
Probability (<i>p</i>): <input type="text" name="p" value="">
<br><br>
<input type="button" name="buttonfinv" value="Run" style="width:100px;" onClick="runfinv()">
<br><br>
Inverse cumulative distribution (<i>x</i>):
<br>
<textarea id="textfoutinv" name="textfoutinv" rows="1" cols="30" style="background:cyan;font-family:Times New Roman;font-size:11pt;color:black" wrap="off" value="      " readonly>
</textarea>
<br><br>
</form>
<font face="Times New Roman" size=1 color=green>
User manual guide:
<br><a
href="http://www.iaees.org/publications/journals/selforganizology/articles/2026-13(1-2)/probFunCal-for-probability-distribution-functions.pdf">Zhang W. J. 2026. probFunCal: A webpage calculator for probability distribution functions. Selforganizology, 13(1-2): 1-25</a></b>
</font>
<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>

<font face="Times New Roman" size=2>
Copyright &copy; 2025 - W. J. Zhang(E-mail: wjzhang@iaees.org)
</font>

<hr>
<table border=0 cellspacing="1" cellpadding="1" width="100%">
<tr>
<td>
<div align=center>
<font size=-1>
International Academy of Ecology and Environmental Sciences. E-mail: office@iaees.org<br>
Copyright &copy; 2009-2024. International Academy of Ecology and Environmental Sciences. All rights reserved.<br>
Web administrator: office@iaees.org, website@iaees.org; <br><br>
<img src="http://www.easycounter.com/counter.php?mathsoft" border="0"> <br>
</div>

Translate page to:<br>
<div id="google_translate_element"></div>
<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>
```

The webpage of the calculator, probFunCal, is shown in Fig. 1.

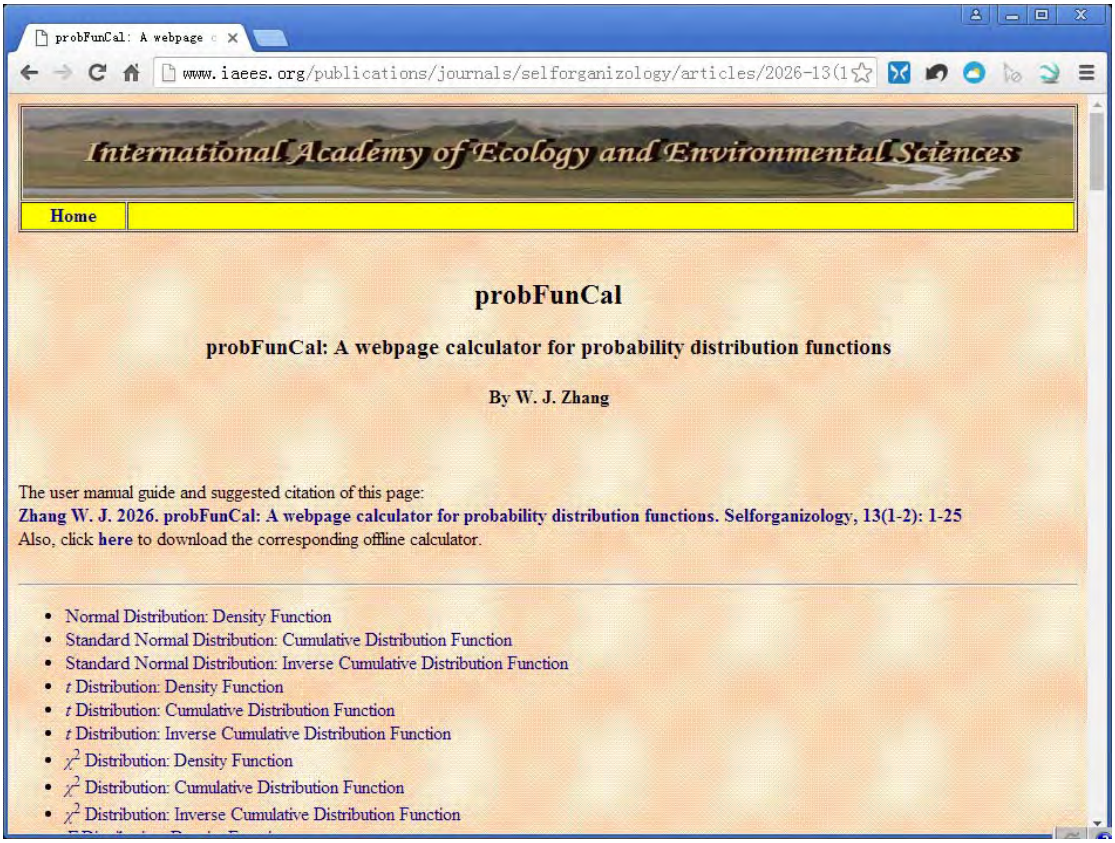

Fig. 1 probFunCal.

References

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