Basics: Working with arbitrary distributions

mathStatica adds about 100 new functions to *Mathematica*. But most of the time, we can get by with just four of them:

function	description	
PlotDensity[*f*]	Plotting (automated)	
Expect[*x*, *f*]	Expectation operator	$E[X]$
Prob[*x*, *f*]	Probability	$P(X \le x)$
Transform[*eqn*, *f*]	Transformations	

Table 1: Core functions for a random variable X with density $f(x)$

This ability to handle plotting, expectations, probability, and transformations, with just four functions, makes the **mathStatica** system very easy to use, even for those not familiar with *Mathematica*.

To illustrate, let us suppose the continuous random variable X has probability density function (pdf)

$$f(x) = \frac{1}{\pi\sqrt{1-x}\sqrt{x}}, \quad \text{for } x \in (0,1).$$

We enter this as:

```
In[1]:= f = 1/(π Sqrt[1 - x] Sqrt[x]);      domain[f] = {x, 0, 1};
```

This is known as the Arc--Sine distribution. Here is a plot of $f(x)$:

```
In[2]:= PlotDensity[f];
```

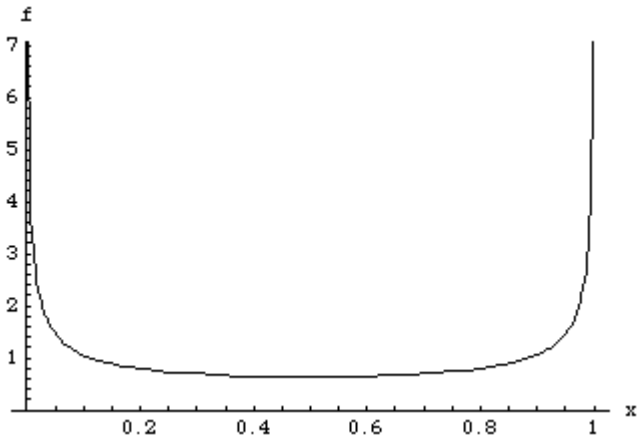


Fig. 1: The Arc-Sine pdf

Here is the cumulative distribution function (cdf), $P(X \le x)$, which also provides the clue to the naming of this distribution:

```
In[3]:= Prob[x, f]

Out[3]= 2 ArcSin[Sqrt[x]] / π
```

The mean, $E[X]$, is:

```
In[4]:= Expect[x, f]
```

$$\text{Out[4]}= \frac{1}{2}$$

while the variance of X is:

```
In[5]:= Var[x, f]
```

$$\text{Out[5]}= \frac{1}{8}$$

The r^{th} moment of X is $E[X^r]$:

```
In[6]:= Expect[x^r, f]
```

This further assumes that: $\left\{r > -\frac{1}{2}\right\}$

$$\text{Out[6]}= \frac{\Gamma\left[\frac{1}{2} + r\right]}{\sqrt{\pi}\,\Gamma[1 + r]}$$

Now consider the transformation to a new random variable Y such that $Y = \sqrt{X}$. By using the `Transform` and `TransformExtremum` functions, the pdf of Y, say $g(y)$, and the domain of its support can be found:

```
In[7]:= g = Transform[y == Sqrt[x], f]
```

$$\text{Out[7]}= \frac{2y}{\pi\sqrt{y^2 - y^4}}$$

```
In[8]:= domain[g] = TransformExtremum[y == Sqrt[x], f]
```

```
Out[8]= {y, 0, 1}
```

So, we have started out with a quite arbitrary pdf $f(x)$, transformed it to a new one $g(y)$, and since both density *g* and its `domain` have been entered into *Mathematica*, we can also apply the **mathStatica** tool set to density g. For example, use `PlotDensity[g]` to plot the pdf of $Y = \sqrt{X}$.

Discrete Random Variables

mathStatica automatically handles discrete random variables in the standard way. The only difference is that, when we define the density, we add a flag to tell *Mathematica* that the random variable is `{Discrete}`. To illustrate, let the discrete random variable X have probability mass function (pmf)

$$f(x) = P(X = x) = \binom{r+x-1}{x} p^r (1-p)^x, \qquad \text{for } x \in \{0, 1, 2, \ldots\}.$$

Here, parameter p is the probability of success, while parameter r is a positive integer. In *Mathematica*, we enter this as:

```
In[1]:=        f = Binomial[r + x - 1, x] p^r (1 - p)^x;
      domain[f] = {x, 0, ∞} && {Discrete} && {0 < p < 1, r > 0, r ∈ Integers};
```

This is known as the Pascal distribution. Here is a plot of $f(x)$:

```
In[2]:= PlotDensity[f /. {p → 1/2, r → 10}];
```

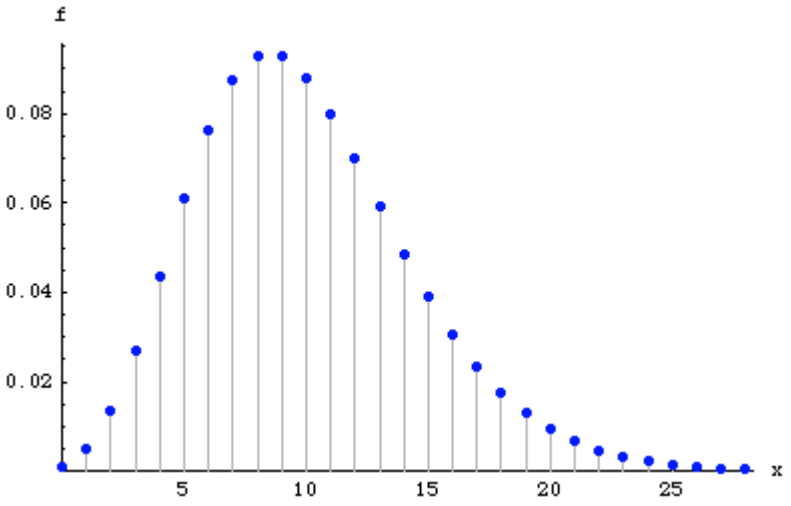


Fig. 1: The pmf of a Pascal discrete random variable

Here is the cdf, equal to $P(X \le x)$:

```
In[3]:= Prob[x, f]

Out[3]= 1 - 1/(Γ[r] Γ[2 + Floor[x]]) ((1 - p)^(1+Floor[x]) p^r Γ[1 + r + Floor[x]]
          Hypergeometric2F1[1, 1 + r + Floor[x], 2 + Floor[x], 1 - p])
```

The mean $E[X]$ and variance of X are given by:

```
In[4]:= Expect[x, f]

Out[4]= (-1 + 1/p) r

In[5]:= Var[x, f]
```

Out[5]= $\frac{r - p\,r}{p^2}$

The probability generating function (pgf) is $E[t^X]$:

In[6]:= **Expect[t^x, f]**

Out[6]= $p^r\ (1 + (-1 + p)\ t)^{-r}$

Fisher Information

The Fisher Information on a parameter is easily derived using **mathStatica**'s `FisherInformation` function. To illustrate, let $X \sim \text{InverseGaussian}(\mu, \lambda)$ with pdf $f(x)$:

```
In[1]:= f = Sqrt[λ / (2 π x^3)] Exp[-λ (x - μ)^2 / (2 μ^2 x)];
        domain[f] = {x, 0, ∞} && {μ > 0, λ > 0};
```

Then, Fisher's Information on (μ, λ) is the (2×2) matrix:

```
In[2]:= FisherInformation[{μ, λ}, f]

Out[2]= ( λ/μ^3   0     )
        ( 0       1/(2λ^2) )
```

Parameter-Mix Distributions

The notation:

$$\text{Poisson}(\Theta) \bigwedge_{\Theta} \text{InverseGaussian}(\mu, \lambda)$$

denotes a $\text{Poisson}(\Theta)$ distribution in which parameter Θ (instead of being fixed) has an $\text{InverseGaussian}(\mu, \lambda)$ distribution. We wish to find the unconditional distribution of X...

Given: The pmf of $X \mid (\Theta = \theta) \sim \text{Poisson}(\theta)$ is $f(x;\theta)$:

```
In[1]:=  f = e^-θ θ^x / x!;
         domain[f] = {x, 0, ∞} && {θ > 0} && {Discrete};
```

Given: The pdf of parameter $\Theta \sim \text{InverseGaussian}(\mu, \lambda)$ is $g(\theta;\mu,\lambda)$:

```
In[2]:=  g = Sqrt[λ / (2 π θ^3)] Exp[-λ (θ - μ)^2 / (2 μ^2 θ)];
         domain[g] = {θ, 0, ∞} && {μ > 0, λ > 0};
```

Then, the parameter-mix distribution is the expectation $E[f(x;\theta)]$ with respect to the distribution of T. The solution with **mathStatica** is simply:

```
In[3]:= h = Expect[f, g]
```

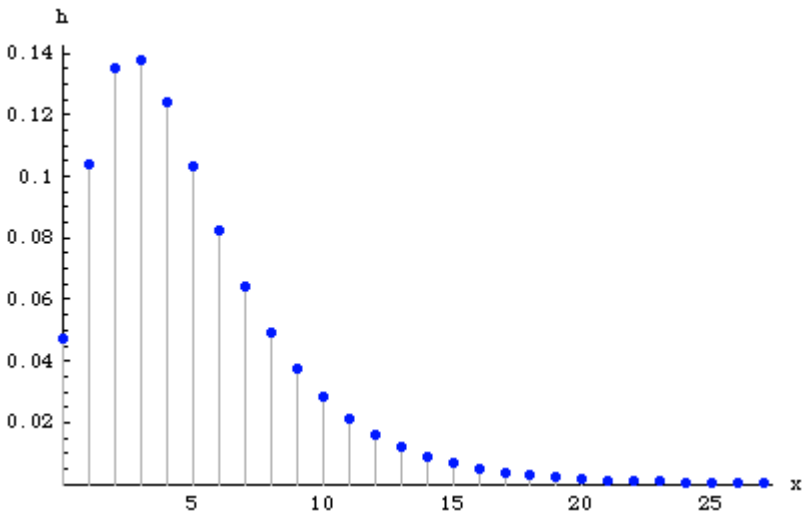
$$\text{Out[3]=}\quad \frac{e^{\lambda/\mu}\sqrt{\frac{2}{\pi}}\,\lambda^{\frac{1}{4}(1+2x)}\,\mu^{-\frac{1}{2}+x}\,(\lambda+2\mu^2)^{\frac{1}{4}(1-2x)}\,\text{BesselK}\left[\frac{1}{2}-x,\ \frac{\sqrt{\lambda(\lambda+2\mu^2)}}{\mu}\right]}{x!}$$

with domain of support:

```
In[4]:= domain[h] = {x, 0, ∞} && {μ > 0, λ > 0} && {Discrete};
```

This is known as Holla's distribution. Here is a plot of its pmf when $\mu = 5$ and $\lambda = 12$:

```
In[5]:= PlotDensity[h /. {μ → 5, λ → 12}];
```



Moment Conversion Functions

mathStatica allows one to express any moment (raw $\acute{\mu}$, central μ, or cumulant ?) in terms of any other moment ($\acute{\mu}$, μ, or ?). For instance, to express the second central moment (the variance) $\mu_2 = E[(X - E[X])^2]$ in terms of raw moments, we enter:

```
In[1]:= CentralToRaw[2]
```

$$\text{Out[1]= } \mu_2 \to -\acute{\mu}_1^2 + \acute{\mu}_2$$

This is just the well-known result that $\mu_2 = E[X^2] - (E[X])^2$. As a further example, here is the sixth cumulant expressed in terms of raw moments:

```
In[2]:= CumulantToRaw[6]
```

$$\text{Out[2]= } \kappa_6 \to -120\,\acute{\mu}_1^6 + 360\,\acute{\mu}_1^4\,\acute{\mu}_2 - 270\,\acute{\mu}_1^2\,\acute{\mu}_2^2 + 30\,\acute{\mu}_2^3 - 120\,\acute{\mu}_1^3\,\acute{\mu}_3 + 120\,\acute{\mu}_1\,\acute{\mu}_2\,\acute{\mu}_3 - 10\,\acute{\mu}_3^2 + 30\,\acute{\mu}_1^2\,\acute{\mu}_4 - 15\,\acute{\mu}_2\,\acute{\mu}_4 - 6\,\acute{\mu}_1\,\acute{\mu}_5 + \acute{\mu}_6$$

The moment converter functions are completely general, and extend in the natural manner to a multivariate framework. Here is the trivariate central moment $\mu_{4,2,2}$ expressed in terms of triivariate cumulants:

```
In[3]:= CentralToCumulant[{4, 2, 2}]
```

$$\begin{aligned}\text{Out[3]= } \mu_{4,2,2} \to\ & 24\,\kappa_{1,0,1}^2\,\kappa_{1,1,0}^2 + 12\,\kappa_{0,2,0}\,\kappa_{1,0,1}^2\,\kappa_{2,0,0} + 48\,\kappa_{0,1,1}\,\kappa_{1,0,1}\,\kappa_{1,1,0}\,\kappa_{2,0,0} + \\ & 12\,\kappa_{0,0,2}\,\kappa_{1,1,0}^2\,\kappa_{2,0,0} + 24\,\kappa_{1,1,1}^2\,\kappa_{2,0,0} + 24\,\kappa_{1,1,0}\,\kappa_{1,1,2}\,\kappa_{2,0,0} + \\ & 12\,\kappa_{1,0,2}\,\kappa_{1,2,0}\,\kappa_{2,0,0} + 24\,\kappa_{1,0,1}\,\kappa_{1,2,1}\,\kappa_{2,0,0} + 6\,\kappa_{0,1,1}^2\,\kappa_{2,0,0}^2 + \\ & 3\,\kappa_{0,0,2}\,\kappa_{0,2,0}\,\kappa_{2,0,0}^2 + 3\,\kappa_{0,2,2}\,\kappa_{2,0,0}^2 + 48\,\kappa_{1,1,0}\,\kappa_{1,1,1}\,\kappa_{2,0,1} + \\ & 24\,\kappa_{1,0,1}\,\kappa_{1,2,0}\,\kappa_{2,0,1} + 12\,\kappa_{0,2,1}\,\kappa_{2,0,0}\,\kappa_{2,0,1} + 6\,\kappa_{0,2,0}\,\kappa_{2,0,1}^2 + \\ & 12\,\kappa_{1,1,0}^2\,\kappa_{2,0,2} + 6\,\kappa_{0,2,0}\,\kappa_{2,0,0}\,\kappa_{2,0,2} + 24\,\kappa_{1,0,2}\,\kappa_{1,1,0}\,\kappa_{2,1,0} + \\ & 48\,\kappa_{1,0,1}\,\kappa_{1,1,1}\,\kappa_{2,1,0} + 12\,\kappa_{0,1,2}\,\kappa_{2,0,0}\,\kappa_{2,1,0} + 24\,\kappa_{0,1,1}\,\kappa_{2,0,1}\,\kappa_{2,1,0} + \\ & 6\,\kappa_{0,0,2}\,\kappa_{2,1,0}^2 + 48\,\kappa_{1,0,1}\,\kappa_{1,1,0}\,\kappa_{2,1,1} + 24\,\kappa_{0,1,1}\,\kappa_{2,0,0}\,\kappa_{2,1,1} + 12\,\kappa_{2,1,1}^2 + \\ & 12\,\kappa_{2,1,0}\,\kappa_{2,1,2} + 12\,\kappa_{1,0,1}^2\,\kappa_{2,2,0} + 6\,\kappa_{0,0,2}\,\kappa_{2,0,0}\,\kappa_{2,2,0} + 6\,\kappa_{2,0,2}\,\kappa_{2,2,0} + \\ & 12\,\kappa_{2,0,1}\,\kappa_{2,2,1} + 6\,\kappa_{2,0,0}\,\kappa_{2,2,2} + 8\,\kappa_{0,2,1}\,\kappa_{1,0,1}\,\kappa_{3,0,0} + 4\,\kappa_{0,2,0}\,\kappa_{1,0,2}\,\kappa_{3,0,0} + \\ & 8\,\kappa_{0,1,2}\,\kappa_{1,1,0}\,\kappa_{3,0,0} + 16\,\kappa_{0,1,1}\,\kappa_{1,1,1}\,\kappa_{3,0,0} + 4\,\kappa_{0,0,2}\,\kappa_{1,2,0}\,\kappa_{3,0,0} + \\ & 4\,\kappa_{1,2,2}\,\kappa_{3,0,0} + 8\,\kappa_{0,2,0}\,\kappa_{1,0,1}\,\kappa_{3,0,1} + 16\,\kappa_{0,1,1}\,\kappa_{1,1,0}\,\kappa_{3,0,1} + 8\,\kappa_{1,2,1}\,\kappa_{3,0,1} + \\ & 4\,\kappa_{1,2,0}\,\kappa_{3,0,2} + 16\,\kappa_{0,1,1}\,\kappa_{1,0,1}\,\kappa_{3,1,0} + 8\,\kappa_{0,0,2}\,\kappa_{1,1,0}\,\kappa_{3,1,0} + 8\,\kappa_{1,1,2}\,\kappa_{3,1,0} + \\ & 16\,\kappa_{1,1,1}\,\kappa_{3,1,1} + 8\,\kappa_{1,1,0}\,\kappa_{3,1,2} + 4\,\kappa_{1,0,2}\,\kappa_{3,2,0} + 8\,\kappa_{1,0,1}\,\kappa_{3,2,1} + \\ & 2\,\kappa_{0,1,1}^2\,\kappa_{4,0,0} + \kappa_{0,0,2}\,\kappa_{0,2,0}\,\kappa_{4,0,0} + \kappa_{0,2,2}\,\kappa_{4,0,0} + 2\,\kappa_{0,2,1}\,\kappa_{4,0,1} + \\ & \kappa_{0,2,0}\,\kappa_{4,0,2} + 2\,\kappa_{0,1,2}\,\kappa_{4,1,0} + 4\,\kappa_{0,1,1}\,\kappa_{4,1,1} + \kappa_{0,0,2}\,\kappa_{4,2,0} + \kappa_{4,2,2}\end{aligned}$$

Unbiased Estimation of Population Moments; Moments of Moments

mathStatica can find unbiased estimators of population moments. For instance, it offers h-statistics (unbiased estimators of population central moments), k-statistics (unbiased estimators of population cumulants), multivariate varieties of the same, polykays (unbiased estimators of products of cumulants) and more. Consider the k-statistic k_r which is an unbiased estimator of the r^{th} cumulant κ_r; that is, $E[k_r] = \kappa_r$, for $r = 1, 2, \ldots$. Here are the 2^{nd} are 3^{rd} k-statistics:

```
In[1]:= k2 = KStatistic[2]
        k3 = KStatistic[3]
```

$$\text{Out[1]}= k_2 \to \frac{-s_1^2 + n\, s_2}{(-1+n)\, n}$$

$$\text{Out[2]}= k_3 \to \frac{2\, s_1^3 - 3\, n\, s_1\, s_2 + n^2\, s_3}{(-2+n)\,(-1+n)\, n}$$

As per convention, the solution is expressed in terms of power sums $s_r = \sum_{i=1}^{n} X_i^r$.

Moments of moments: Because the above expressions (sample moments) are functions of random variables X_i, we might want to calculate population moments of them. With **mathStatica**, we can find any moment (raw, central, or cumulant) of the above expressions. For instance, k_3 is meant to have the property that $E[k_3] = \kappa_3$. We test this by calculating the first raw moment of k_3, and express the answer in terms of cumulants:

```
In[3]:= RawMomentToCumulant[1, k3[[2]]]
```

$$\text{Out[3]}= \kappa_3$$

In 1928, Fisher published the product cumulants of the k-statistics, which are now listed in reference bibles such as Stuart and Ord (1994). Here is the solution to $\kappa_{2,2}(k_3, k_2)$:

```
In[4]:= CumulantMomentToCumulant[{2, 2}, {k3[[2]], k2[[2]]}]
```

$$\begin{aligned}\text{Out[4]}= &\frac{288\, n\, \kappa_2^5}{(-2+n)\,(-1+n)^3} + \frac{288\,(-23+10\,n)\,\kappa_2^2\,\kappa_3^2}{(-2+n)\,(-1+n)^3} + \\ &\frac{360\,(-7+4\,n)\,\kappa_2^3\,\kappa_4}{(-2+n)\,(-1+n)^3} + \frac{36\,(160-155\,n+38\,n^2)\,\kappa_3^2\,\kappa_4}{(-2+n)\,(-1+n)^3\, n} + \\ &\frac{36\,(93-103\,n+29\,n^2)\,\kappa_2\,\kappa_4^2}{(-2+n)\,(-1+n)^3\, n} + \frac{24\,(202-246\,n+71\,n^2)\,\kappa_2\,\kappa_3\,\kappa_5}{(-2+n)\,(-1+n)^3\, n} + \\ &\frac{2\,(113-154\,n+59\,n^2)\,\kappa_5^2}{(-1+n)^3\, n^2} + \frac{6\,(-131+67\,n)\,\kappa_2^2\,\kappa_6}{(-2+n)\,(-1+n)^2\, n} + \\ &\frac{3\,(117-166\,n+61\,n^2)\,\kappa_4\,\kappa_6}{(-1+n)^3\, n^2} + \frac{6\,(-27+17\,n)\,\kappa_3\,\kappa_7}{(-1+n)^2\, n^2} + \frac{37\,\kappa_2\,\kappa_8}{(-1+n)\, n^2} + \frac{\kappa_{10}}{n^3}\end{aligned}$$

This is the correct solution. Unfortunately, the solutions given in Stuart and Ord (1994, equation (12.70)) and Fisher (1928) are actually incorrect.

Multivariate Random Variables

mathStatica extends naturally to a multivariate setting. To illustrate, let us suppose that X and Y have joint pdf $f(x, y)$ with support $x > 0$, $y > 0$:

```
In[1]:=          f  = e^(-2 (x+y)) (e^(x+y) + α (e^x - 2) (e^y - 2));
      domain[f] = {{x, 0, ∞}, {y, 0, ∞}} && {-1 < α < 1};
```

where parameter a is such that $-1 < \alpha < 1$. This is known as a Gumbel bivariate Exponential distribution. Here is a plot of $f(x, y)$:

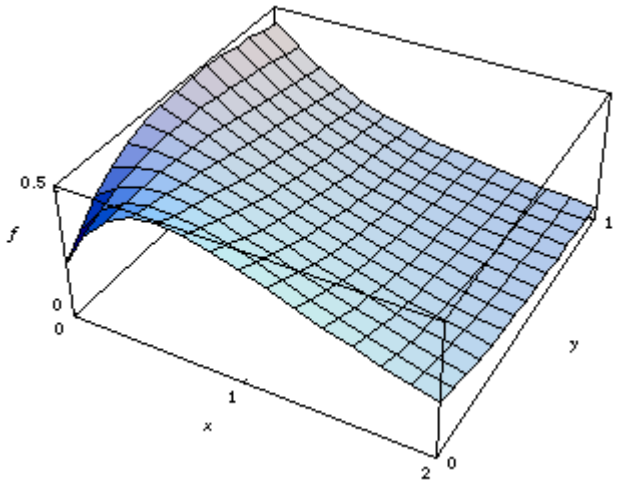


Fig. 1: A Gumbel bivariate Exponential pdf when $\alpha = -0.8$

Here is the cdf, namely $P(X \le x, Y \le y)$:

```
In[2]:= Prob[{x, y}, f]
```

$$\text{Out[2]}= e^{-2(x+y)} (-1 + e^x) (-1 + e^y) (e^{x+y} + \alpha)$$

Here is $\text{Cov}(X, Y)$, the covariance between X and Y:

```
In[3]:= Cov[{x, y}, f]
```

$$\text{Out[3]}= \frac{\alpha}{4}$$

More generally, here is the variance-covariance matrix:

```
In[4]:= Varcov[f]
```

$$\text{Out[4]}= \begin{pmatrix} 1 & \frac{\alpha}{4} \\ \frac{\alpha}{4} & 1 \end{pmatrix}$$

Here is the marginal pdf of X:

```
In[5]:= Marginal[x, f]
```

$$\text{Out[5]}= e^{-x}$$

Here is the conditional pdf of Y, given $X = x$:

In[6]:= **Conditional[y, f]**

Here is the conditional pdf f (y | x):

Out[6]= $e^{x-2(x+y)} (e^{x+y} + (-2 + e^x)(-2 + e^y)\alpha)$

Here is the bivariate mgf $E[e^{t_1 X + t_2 Y}]$:

In[7]:= **mgf = Expect[** $e^{t_1 x + t_2 y}$**, f]**

This further assumes that: $\{t_1 < 1, t_2 < 1\}$

Out[7]= $\frac{4 - 2 t_2 + t_1 (-2 + (1 + \alpha) t_2)}{(-2 + t_1)(-1 + t_1)(-2 + t_2)(-1 + t_2)}$

Differentiating the mgf is one way to derive moments. Here is the product moment $E[X^2 Y^2]$:

In[8]:= **D[mgf, {** t_1**, 2}, {** t_2**, 2}] /. t_ → 0 // Simplify**

Out[8]= $4 + \frac{9\alpha}{4}$

which we could otherwise have found directly with:

In[9]:= **Expect[** $x^2 y^2$**, f]**

Out[9]= $4 + \frac{9\alpha}{4}$

Multivariate transformations pose no problem to **mathStatica** either. For instance, let $U = \frac{Y}{1+X}$ and $V = \frac{1}{1+X}$ denote transformations of X and Y. Then our transformation equation is:

In[10]:= **eqn =** $\{u == \frac{y}{1+x}, v == \frac{1}{1+x}\}$**;**

Using `Transform`, we can find the joint pdf of random variables U and V, denoted $g(u, v)$:

In[11]:= **g = Transform[eqn, f]**

Out[11]= $\frac{e^{\frac{-2-2u+v}{v}} \left(4 e \alpha - 2 e^{1+\frac{u}{v}} \alpha - 2 e^{\frac{1}{v}} \alpha + e^{\frac{1+u}{v}} (1 + \alpha)\right)}{v^3}$

while the extremum of the domain of support of the new random variables are:

In[12]:= **TransformExtremum[eqn, f]**

Out[12]= {{u, 0, ∞}, {v, 0, 1}}

Non-Parametric Kernel Density Estimation

Example 1: Kernel density estimation

Here is some raw data measuring the diagonal length of 100 forged Swiss bank notes and 100 real Swiss bank notes (Simonoff, 1996):

```
In[1]:= data = ReadList["sd.dat"];
```

Non-parametric kernel density estimation involves two components:
(i) the choice of a kernel, and (ii) the selection of a bandwidth.

Here we use a Gaussian kernel f:

```
In[2]:= f = e^(-x^2/2) / Sqrt[2 π];    domain[f] = {x, -∞, ∞};
```

Next, we select the bandwidth c. Small values for c produce a rough estimate while large values produce a very smooth estimate. A number of methods exist to automate bandwidth choice; **mathStatica** implements both the Silverman (1986) approach and the more sophisticated Sheather and Jones (1991) method. For the Swiss bank note data set, the Sheather--Jones optimal bandwidth (using the Gaussian kernel f) is:

```
In[3]:= c = Bandwidth[data, f, Method → SheatherJones]

Out[3]= 0.200059
```

We can now plot the smoothed non-parametric kernel density estimate using the `NPKDEPlot[`*data*, *kernel*, *c*`]` function:

```
In[4]:= NPKDEPlot[data, f, c];
```

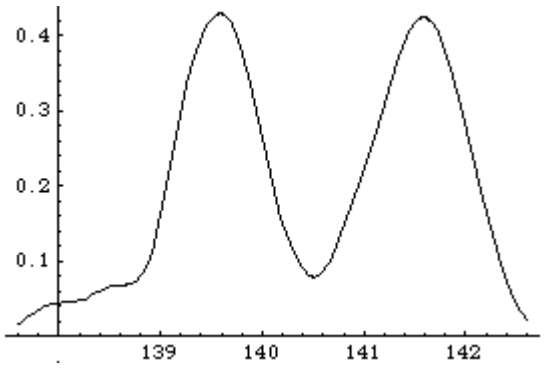


Fig. 1: The smoothed non-parametric kernel density estimate (Swiss bank notes)

Example 2: Family of Curves 

Instead of presenting a *single* curve estimate corresponding to a single bandwidth, Marron and Chung (2001) argue that it is beneficial to present a *family* of curves corresponding to a range of different bandwidths. The family of curves reveals deeper structure, can show more information in a single plot, and can make it easier to select an appropriate bandwidth.

To illustrate, we consider Parzen's (1979) yearly 'Snowfall in Buffalo' data (63 data points collected from 1910 to 1972, and measured in inches):

In[1]:=
```
data = ReadList["snowfall.dat"];
```

Kernel: We shall use an Epanechnikov kernel f here:

In[2]:=
```
f = 3/4 (1 - x^2);    domain[f] = {x, -1, 1};
```

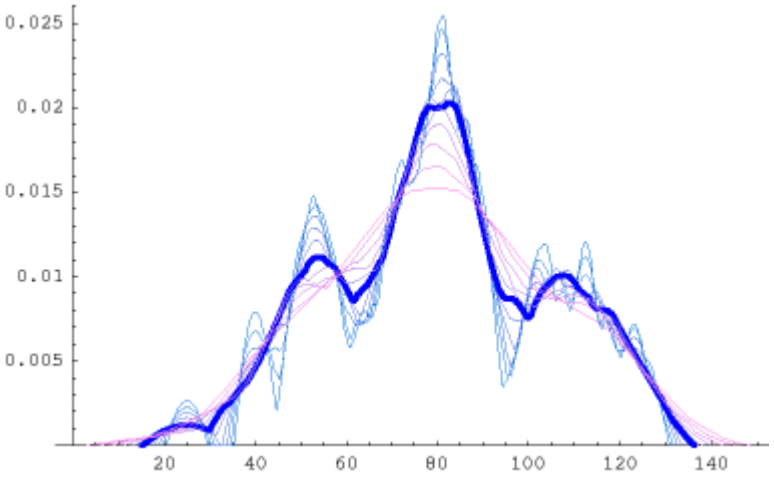
Bandwidth: We shall select qq = 11 different bandwidths, ranging from a minimum (bumpy) to a maximum (smooth), as follows:

In[3]:=
```
qq = 11;    cmin = 4.5;    cmax = 22;
c = Table[cmin (cmax/cmin)^((i-1)/(qq-1)), {i, 1, qq}] // N
```

Out[3]:=
```
{4.5, 5.27392, 6.18094, 7.24395, 8.48978, 9.94987, 11.6611, 13.6666, 16.017, 18.7716, 22.}
```

Then, in **mathStatica** 1.2, the selected family of curves is neatly obtained with:

In[4]:=
```
NPKDEPlot[data, f, c]; // Timing
```



Out[4]:=
```
{0.875 Second, Null}
```

References

Marron, J.S. and Chung, S.S. (2001), Presentation of smoothers: the family approach, *Computational Statistics*, 16, 195-207.

Example 3: Symbolic NPKDE New!

Non-parametric kernel density estimation is highly computationally intensive. Consequently, some computer programs try to resolve the speed problem by computing the kernel density estimate using approximate / inexact methods that reduce the amount of computation involved. Such methods do

not calculate the desired estimate per se; rather, they provide an approximation of the estimate. By contrast, **mathStatica** always uses exact methods, and so tries to tackle the speed problem by using carefully optimised code.

To illustrate that **mathStatica**'s NPKDE function calculates the kernel density estimate using exact methods, we now provide a completely algebraic / symbolic example. Suppose we select an Epanechnikov kernel $f(x)$:

In[1]:=
```
f = 3/4 (1 - x^2);    domain[f] = {x, -1, 1};
```

... and that $\{X_1, \ldots, X_{10}\}$ is a random sample of size $n = 10$:

In[2]:=
```
Xdata = Thread[X_Range[10]]
```

Out[2]:=
$$\{X_1, X_2, X_3, X_4, X_5, X_6, X_7, X_8, X_9, X_{10}\}$$

Then, for any arbitrary bandwidth *bin*, the symbolic non-parametric kernel density estimator, calculated at an arbitrary point *x*, is:

In[3]:=
```
NPKDE[x, Xdata, f, bin]
```

Out[3]:=
$$\frac{1}{10\,\text{bin}}\left(\text{If}\left[-1 \le \frac{x - X_1}{\text{bin}} \le 1,\ \frac{3}{4}\left(1 - \left(\frac{x - X_1}{\text{bin}}\right)^2\right),\ 0\right] + \text{If}\left[-1 \le \frac{x - X_2}{\text{bin}} \le 1,\ \frac{3}{4}\left(1 - \left(\frac{x - X_2}{\text{bin}}\right)^2\right),\ 0\right] + \text{If}\left[-1 \le \frac{x - X_3}{\text{bin}} \le 1,\ \frac{3}{4}\left(1 - \left(\frac{x - X_3}{\text{bin}}\right)^2\right),\ 0\right] + \text{If}\left[-1 \le \frac{x - X_4}{\text{bin}} \le 1,\ \frac{3}{4}\left(1 - \left(\frac{x - X_4}{\text{bin}}\right)^2\right),\ 0\right] + \text{If}\left[-1 \le \frac{x - X_5}{\text{bin}} \le 1,\ \frac{3}{4}\left(1 - \left(\frac{x - X_5}{\text{bin}}\right)^2\right),\ 0\right] + \text{If}\left[-1 \le \frac{x - X_6}{\text{bin}} \le 1,\ \frac{3}{4}\left(1 - \left(\frac{x - X_6}{\text{bin}}\right)^2\right),\ 0\right] + \text{If}\left[-1 \le \frac{x - X_7}{\text{bin}} \le 1,\ \frac{3}{4}\left(1 - \left(\frac{x - X_7}{\text{bin}}\right)^2\right),\ 0\right] + \text{If}\left[-1 \le \frac{x - X_8}{\text{bin}} \le 1,\ \frac{3}{4}\left(1 - \left(\frac{x - X_8}{\text{bin}}\right)^2\right),\ 0\right] + \text{If}\left[-1 \le \frac{x - X_9}{\text{bin}} \le 1,\ \frac{3}{4}\left(1 - \left(\frac{x - X_9}{\text{bin}}\right)^2\right),\ 0\right] + \text{If}\left[-1 \le \frac{x - X_{10}}{\text{bin}} \le 1,\ \frac{3}{4}\left(1 - \left(\frac{x - X_{10}}{\text{bin}}\right)^2\right),\ 0\right]\right)$$

Order Statistics

Example 1: Continuous Logistic distribution

Let random variable X have a Logistic distribution with pdf $f(x)$:

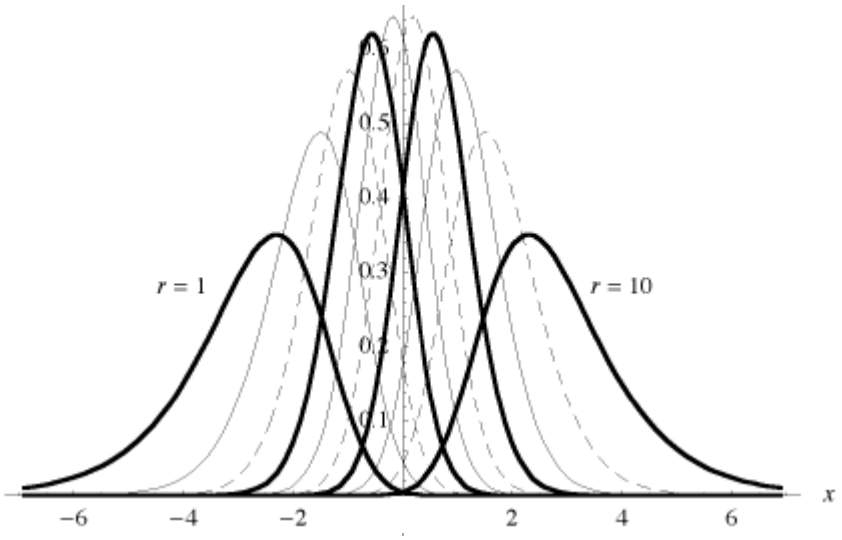
In[1]:=
```
f = e^-x / (1 + e^-x)^2;   domain[f] = {x, -∞, ∞};
```

Let $(X_1, X_2, \ldots, X_n)$ denote a sample of size n drawn on X, and let $(X_{(1)}, X_{(2)}, \ldots, X_{(n)})$ denote the ordered sample, so that $X_{(1)} < X_{(2)} < \cdots < X_{(n)}$. The pdf of the r^{th} order statistic, $X_{(r)}$, is given by the **mathStatica** function:

In[2]:=
```
OrderStat[r, f]
```

Out[2]=
$$\frac{(1+e^{-x})^{-r}\,(1+e^{x})^{-1-n+r}\,n!}{(n-r)!\,(-1+r)!}$$

The following diagram plots the pdf of the r^{th} order statistic, as r increases from 1 to 10, given a sample of size $n = 10$:



The joint pdf of $X_{(r)}$ and $X_{(s)}$, for $r < s$, is given by:

In[3]:=
```
OrderStat[{r, s}, f]
```

Out[3]=
$$\frac{e^{x_s}\,(1+e^{-x_r})^{-r}\,(1+e^{x_s})^{-1-n+s}\left(\frac{1}{1+e^{x_r}}-\frac{1}{1+e^{x_s}}\right)^{-r+s}\Gamma[1+n]}{(-e^{x_r}+e^{x_s})\,\Gamma[r]\,\Gamma[1+n-s]\,\Gamma[-r+s]}$$

Example 2: Discrete Negative Binomial distribution New!

A new feature in **mathStatica** 1.2 is that the `OrderStat` function now also supports discrete random variables. To illustrate, let random variable $X \sim \text{NegativeBinomial}(p, \lambda)$ with pmf $f(x)$:

In[1]:=
```
f = Binomial[λ + x - 1, λ - 1] p^λ (1 - p)^x;
domain[f] = {x, 0, ∞} && {0 < p < 1, λ > 0} && {Discrete};
```

Let $(X_1, \ldots, X_n)$ denote a random sample of size *n* drawn on *X*, and let $(X_{(1)}, \ldots, X_{(n)})$ denote the order statistics.

Then, the pmf of the *r* th order statistic, $X_{(r)}$, denoted $g(x)$, is given immediately by:

In[2]:=
```
g = OrderStat[r, f]
```

Out[2]:=

$$\frac{1}{\text{Beta}[r,\ 1+n-r]}\left(-\text{Beta}\left[1-\frac{(1-p)^x\, p^\lambda\, \Gamma[x+\lambda]\ \text{Hypergeometric2F1}[1,\ x+\lambda,\ 1+x,\ 1-p]}{\Gamma[1+x]\,\Gamma[\lambda]},\ r,\ 1+n-r\right] + \text{Beta}\left[1-\frac{(1-p)^{1+x}\, p^\lambda\, \Gamma[1+x+\lambda]\ \text{Hypergeometric2F1}[1,\ 1+x+\lambda,\ 2+x,\ 1-p]}{\Gamma[2+x]\,\Gamma[\lambda]},\ r,\ 1+n-r\right]\right)$$

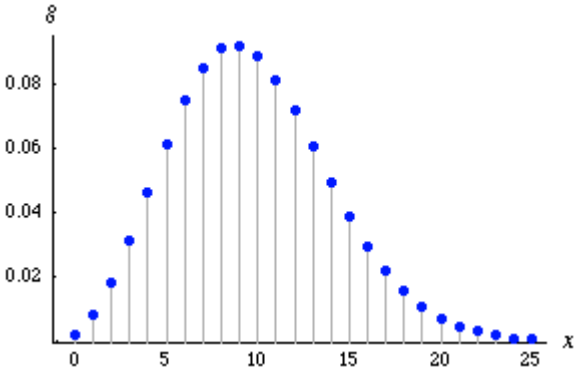
with domain of support:

In[3]:=
```
domain[g] = OrderStatDomain[r, f]
```

Out[3]:=
```
{x, 0, ∞} && {n ∈ Integers, r ∈ Integers, λ > 0, 0 < p < 1, 1 ≤ r ≤ n} && {Discrete}
```

The following diagram plots the pmf of the *r* th order statistic that we have just derived, when $p = \frac{1}{8}$, $\lambda = 4$ and *r* = 1 and the sample size is *n* = 10:

In[4]:=
```
PlotDensity[g /. {p → 1/8, λ → 4, r → 1, n → 10}];
```



Pearson Fitting

Karl Pearson showed that if we know the first four moments of a distribution, we can construct a density function that is consistent with those moments. This can provide a neat way to build density functions that approximate a given set of data. For instance, for a given data set, let us suppose that:

```
In[1]:= mean = 37.875;
        μ̂234 = {191.55, 1888.36, 107703.3};
```

denoting estimates of the mean, and of the second, third and fourth central moments. The Pearson family consists of 7 main *Types*, so our first task is to find out which type this data is consistent with. We do this with mathStatica's `PearsonPlot` function:

```
In[2]:= PearsonPlot[μ̂234];

{β1 → 0.507368, β2 → 2.93538}
```

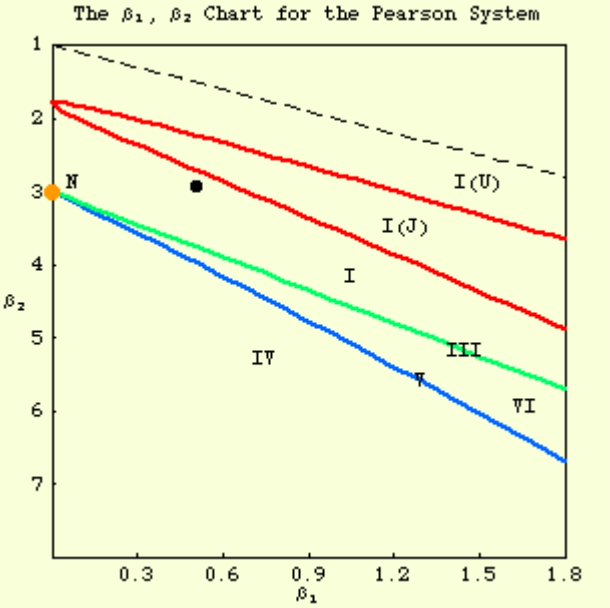


Fig. 1: The β_1, β_2 chart for the Pearson system

The big black dot in Fig. 1 is in the *Type I* zone. Then, the fitted Pearson density $f(x)$ and its domain are immediately given by:

```
In[3]:= {f, domain[f]} = PearsonI[mean, μ̂234, x]

Out[3]= {9.62522×10^-8 (94.3127 - 1. x)^2.7813 (-16.8709 + 1. x)^0.407265, {x, 16.8709, 94.3127}}
```

The actual data used to create this example is grouped data depicting the number of sick people (`freq`) at different ages (`X`):

```
In[4]:=   X = {17,  22,  27,  32,  37,  42, 47, 52, 57, 62, 67, 72, 77, 82, 87};
       freq = {34, 145, 156, 145, 123, 103, 86, 71, 55, 37, 21, 13,  7,  3,  1};
```

We can easily compare the histogram of the empirical data with our fitted Pearson pdf:

```
In[5]:= FrequencyGroupPlot[{X, freq}, f];
```

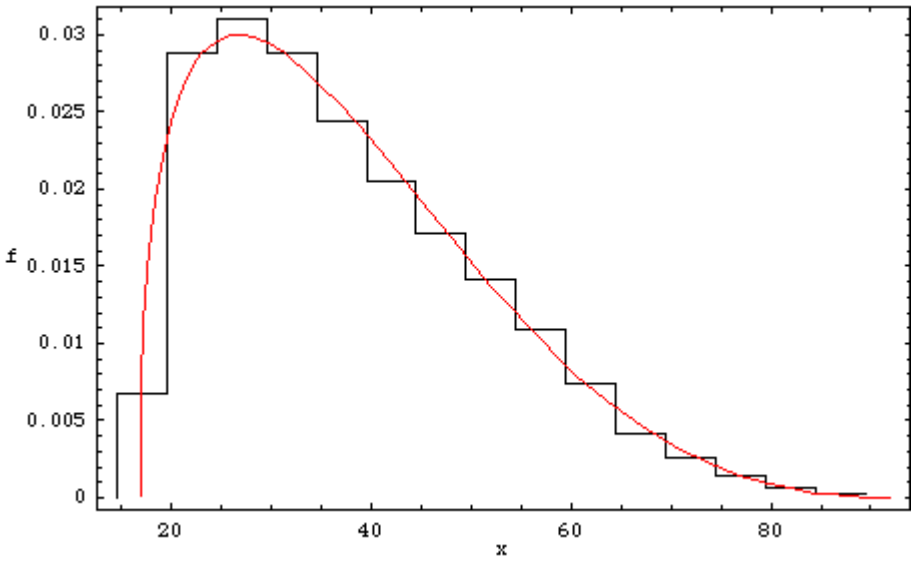


Fig. 2: The data histogram and the fitted Pearson pdf

Piecewise Distributions

Some density functions take a bipartite form. To illustrate, let us suppose X is a continuous random variable, $0 < x < 1$, with pdf

$$f(x) = \begin{cases} 2\left(\frac{c-x}{c}\right) & \text{if } x < c \\ 2\left(\frac{x-c}{1-c}\right) & \text{if } x \geq c \end{cases}$$

where $0 < c < 1$. We enter this as:

```
In[1]:=   f = If[x < c, 2 (c - x)/c, 2 (x - c)/(1 - c)];

          domain[f] = {x, 0, 1} && {0 < c < 1};
```

This is known as the Inverse Triangular distribution, as is clear from a plot of $f(x)$, as illustrated in Fig.Ûd1.

```
In[2]:= PlotDensity[f /. c → {1/4, 1/2, 3/4}];
```

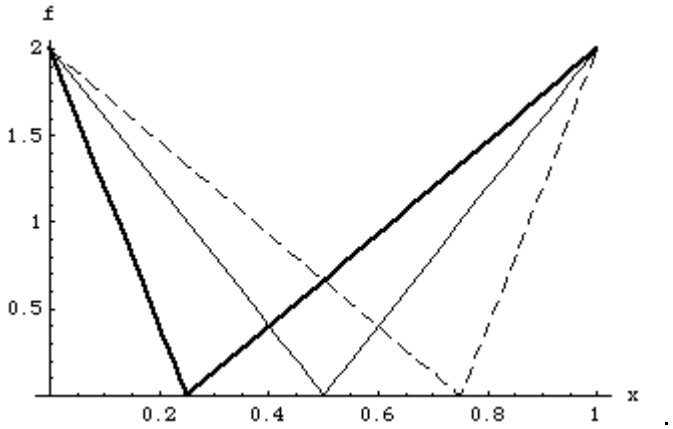


Fig. 1: The Inverse Triangular pdf, when $c = \frac{1}{4}$ (bold), $\frac{1}{2}$ (plain), $\frac{3}{4}$ (dashed)

Here is the cdf, $P(X \leq x)$:

```
In[3]:= Prob[x, f]

Out[3]= If[x < c, x (2 - x/c), (c - 2 c x + x^2)/(1 - c)]
```

Note that the solution depends on whether $x < c$ or $x \geq c$. Figure 2 plots the cdf at the same three values of c used in Fig.1.

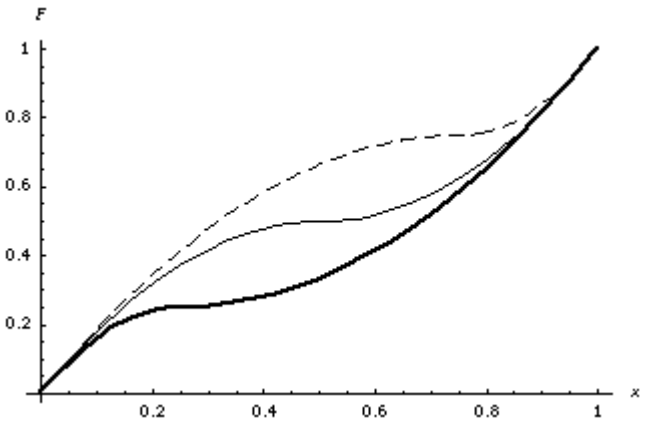


Fig. 2: The Inverse Triangular cdf, when $c = \frac{1}{4}$ (bold), $\frac{1}{2}$ (plain), $\frac{3}{4}$ (dashed)

mathStatica operates on bipartite distributions in the standard way. For instance, the mean $E[X]$ is given by:

```
In[4]:= Expect[x, f]
```

$$\text{Out[4]=} \quad \frac{2 - c}{3}$$

while the entropy is given by $E[-\log(f(X))]$:

```
In[5]:= Expect[-Log[f], f]
```

$$\text{Out[5]=} \quad \frac{1}{2} - \text{Log}[2]$$

Pseudo-Random Number Generation

Let X be **any** discrete random variable with pmf $f(x)$. To illustrate, suppose $X \sim \text{Poisson}(6)$:

```
In[1]:= f = e^-λ λ^x / x! /. λ → 6;        domain[f] = {x, 0, ∞} && {Discrete};
```

We now generate 20 copies of X:

```
In[2]:= DiscreteRNG[20, f]

Out[2]= {11, 1, 7, 3, 4, 8, 6, 9, 6, 5, 5, 7, 6, 13, 5, 8, 5, 6, 6, 6}
```

Here, in a fraction of a second, are 50000 more copies of X:

```
In[3]:= data = DiscreteRNG[50000, f]; // Timing

Out[3]= {0.39 Second, Null}
```

Contrast the empirical distribution of `data` with the true distribution of X:

```
In[4]:= FrequencyPlotDiscrete[data, f];
```

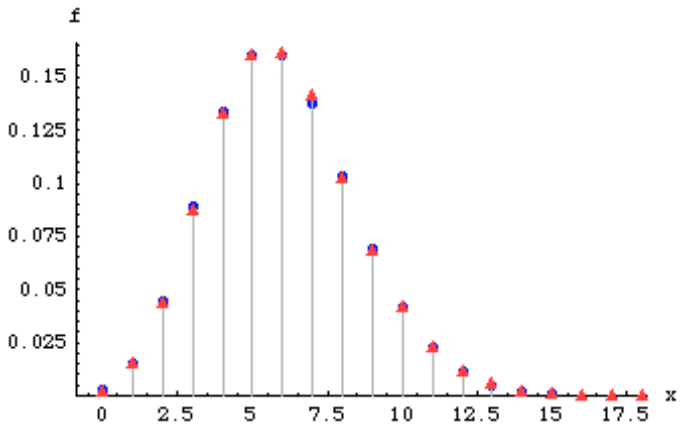


Fig. 1: The empirical pmf (red triangles) and true pmf (blue circles)

The triangular dots denote the empirical pmf, while the round dots denote the true density $f(x)$. One obtains a superb fit because **mathStatica**'s `DiscreteRNG` is an exact solution.

Symbolic Maximum Likelihood Estimation

Although statistical software has long been used for maximum likelihood (ML) estimation, the focus of attention has almost always been on obtaining ML estimates (a numerical problem), rather than on deriving ML estimators (a symbolic problem). **mathStatica** makes it possible to derive *exact* symbolic ML estimators from first principles with a computer algebra system.

For instance, consider the following simple problem: let $(X_1, \ldots, X_n)$ denote a random sample of size *n* collected on $X \sim \text{Rayleigh}(\sigma)$, where parameter $\sigma > 0$ is unknown. We wish to find the ML estimator of s . We begin in the usual way by inputting the likelihood function into *Mathematica*:

```
In[1]:= L = Product[x_i / σ^2 Exp[-x_i^2 / (2 σ^2)], {i, 1, n}];
```

If we try to evaluate the log-likelihood:

```
In[2]:= Log[L]
```

$$\text{Out[2]= } \text{Log}\left[\prod_{i=1}^{n} \frac{e^{-\frac{x_i^2}{2\sigma^2}} x_i}{\sigma^2}\right]$$

... nothing happens! (*Mathematica* assumes nothing about the symbols that have been entered, so its inaction is perfectly reasonable.) But we can enhance `Log` to do what is wanted here using the **mathStatica** function `SuperLog`. To activate this enhancement, we switch it on:

```
In[3]:= SuperLog[On]

SuperLog is now On.
```

If we now evaluate `Log[L]` again, we obtain a much more useful result:

```
In[4]:= logL = Log[L]
```

$$\text{Out[4]= } -2\,n\,\text{Log}[\sigma] + \sum_{i=1}^{n} \text{Log}[x_i] - \frac{\sum_{i=1}^{n} x_i^2}{2\sigma^2}$$

To derive the first-order conditions for a maximum:

```
In[5]:= FOC = D[logL, σ]
```

$$\text{Out[5]= } -\frac{2\,n}{\sigma} + \frac{\sum_{i=1}^{n} x_i^2}{\sigma^3}$$

... we solve `FOC==0` using *Mathematica*'s `Solve` function. The ML estimator $\hat{\sigma}$ is given as a replacement rule -> for s :

```
In[6]:= σ̂ = Solve[FOC == 0, σ][[2]]
```

$$\text{Out[6]= } \left\{\sigma \to \frac{\sqrt{\sum_{i=1}^{n} x_i^2}}{\sqrt{2}\,\sqrt{n}}\right\}$$

The second-order conditions (evaluated at the first-order conditions) are always negative, which confirms that $\hat{\sigma}$ is indeed the ML estimator:

```
In[7]:= SOC = D[logL, {σ, 2}] /. σ̂
```

$$\text{Out[7]}= -\frac{8\,n^2}{\sum_{i=1}^{n} x_i^2}$$

Finally, let us suppose that an observed random sample is $\{1, 6, 3, 4\}$:

```
In[8]:= data = {1, 6, 3, 4};
```

Then the ML estimate of s is obtained by substituting this data into the ML estimator $\hat{\sigma}$:

```
In[9]:= σ̂ /. {n → 4, x_i_ :> data[[i]]}
```

$$\text{Out[9]}= \left\{\sigma \to \frac{\sqrt{31}}{2}\right\}$$

Figure 1 plots the observed likelihood (for the given data) against values of s, noting the derived exact optimal solution $\hat{\sigma} = \frac{\sqrt{31}}{2}$.

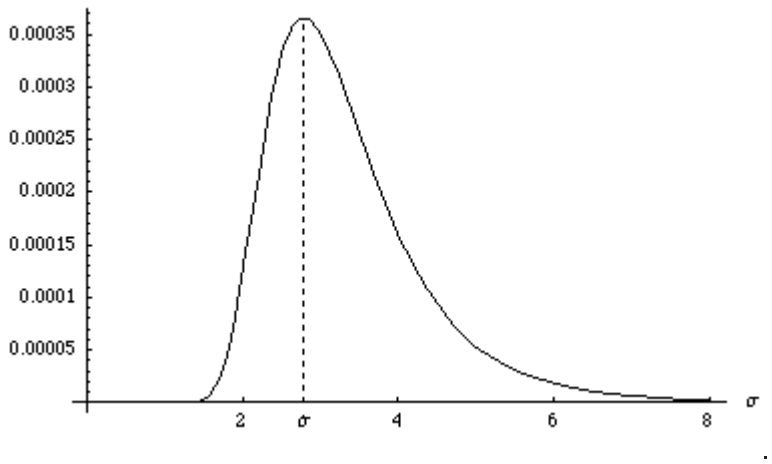


Fig. 1: The observed likelihood and $\hat{\sigma}$