Entropy loss by promediating

Let $\bar{V}$ a vector which represents a sample, a set of measures about anything. It is composed by a number, N, of IID random numbers.

We can think about an entropy associated to this sample Φ and it can be computed as $\Phi = N \cdot \phi$

Where ϕ is the entropy associated with the IID density function and the sum accounts for the independence between $\bar{V}$ components.

Relationship between σ^2 and the entropy for uniform

For uniform $\sigma^2 = \frac{(max-min)^2}{12}$

But $\Phi = N \log(\max(v_i) - \min(v_i))$ so $\sigma^2 = \frac{e^{2\frac{\Phi}{N}}}{12}$

Relationship between σ^2 and the entropy for normal

For normal $\Phi = \frac{N}{2} \log(2\pi e \sigma^2)$

So $\sigma^2 = \frac{\exp(\frac{2\Phi}{N})}{2\pi e}$

Relationship between σ^2 and the entropy for binomial

For simplicity let $q = 1 - p$

Then $\mu = np$ and $\sigma^2 = npq$

$\Phi = N \left(- \left\langle \log \binom{n}{i} \right\rangle - n \log(p^p q^q) \right)$

where $\left\langle \log \binom{n}{i} \right\rangle$ stands for the expected value of the log of the combinatorial number n choose i

If we compute the mean of the sample, then we can compute the entropy of this statistic in the limit of **central limit theorem** (**CLT**), so

$\mu \to \mathcal{N}\left(\mu, \frac{\sigma^2}{N}\right)$ and $H_\mu = \log\left(\sqrt{\frac{2\pi e \sigma^2}{N}}\right)$

The entropy lost by the mean with respect the total amount of entropy in the original sample could be computed as

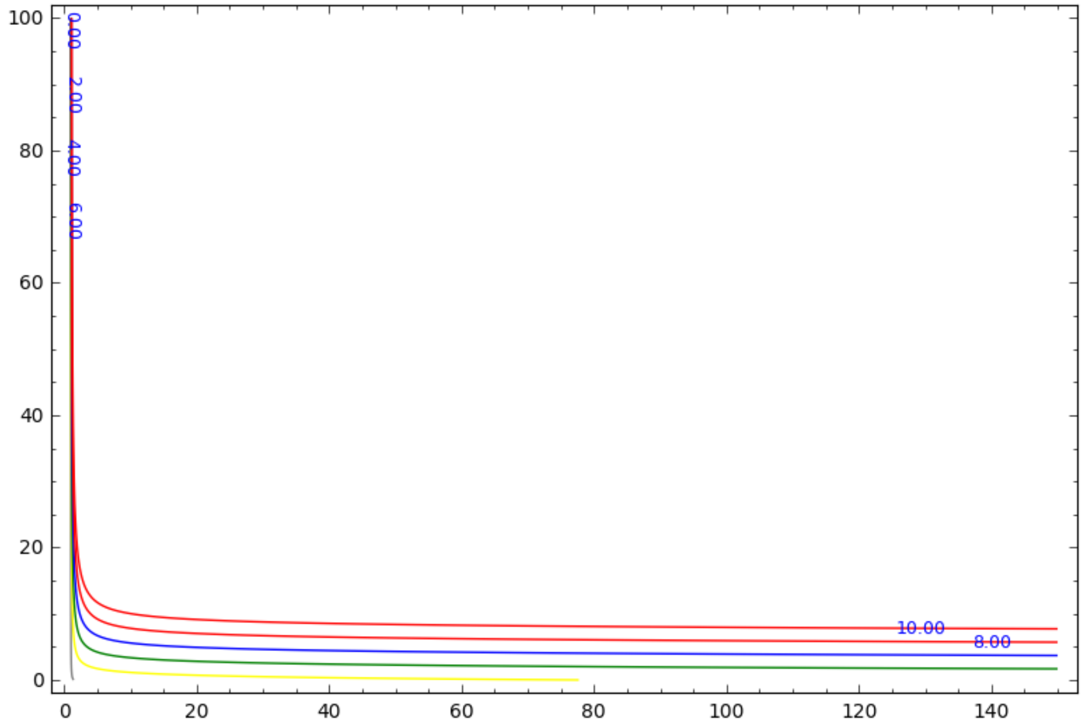
$\rho_\mu = \Phi - H_\mu$

If the IID is uniform we can write $\log(\sigma^2) = \frac{2\Phi}{N} - \log(12)$

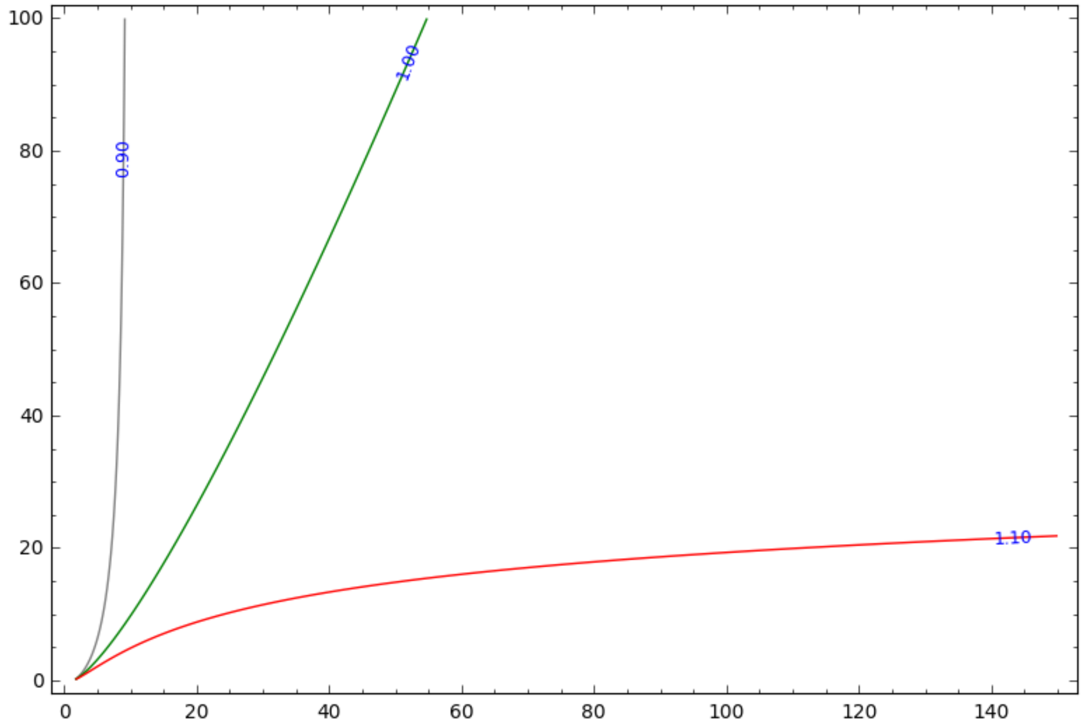
and $H_\mu = \frac{1}{2}\left(\log(2\pi e) + \frac{2\Phi}{N} - \log(12) - \log(N)\right)$

so $\rho_\mu = \Phi \frac{N-1}{N} + \frac{\log(N)}{2} - \log(\sqrt{\frac{\pi e}{6}})$

```
x,y=var('x','y')
contour_plot(log(x)/2+y*(x-1)/x-log(sqrt(pi*exp(1)/6)),(x,1,150),(y,0,100), fill=False,
plot_points=750, contours=[0,2,4,6,8,10],cmap =['grey','yellow','green','blue','red'],
labels=True)
```




```
x,y=var('x','y')
contour_plot((log(x)/2+y*(x-1)/x-log(sqrt(pi*exp(1)/6)))/y,(x,1,150),(y,0,100),
fill=False, plot_points=750, contours=[0.9,1,1.1],cmap
=['grey','yellow','green','blue','red'], labels=True)
```



If the original IID is normal, then we can write $\log(2\pi e\sigma^2) = \frac{2\Phi}{N}$

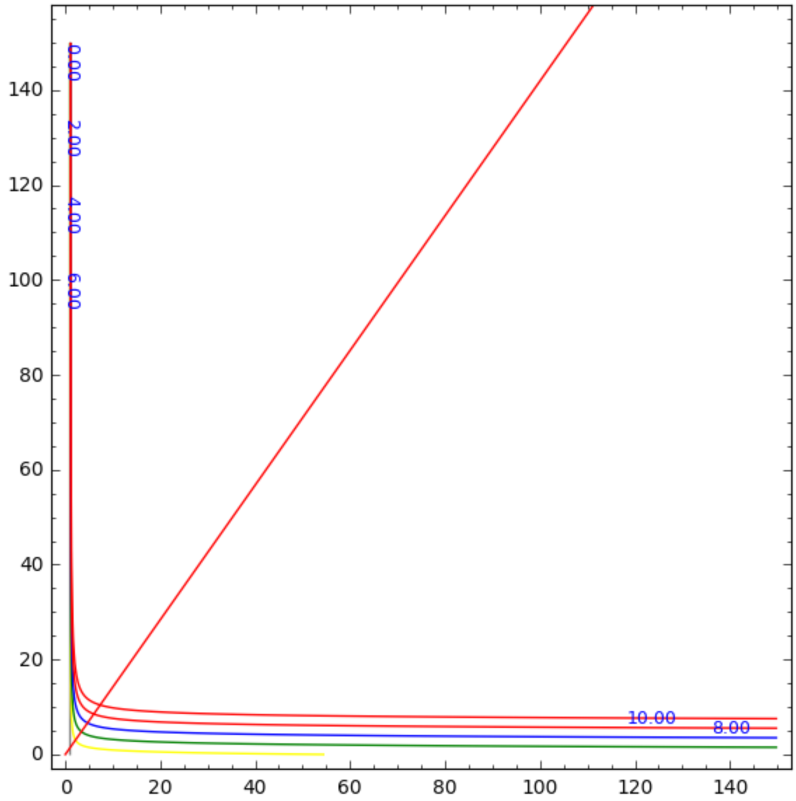
so $H_\mu = \frac{1}{2}\left(\frac{2\Phi}{N} - \log(N)\right)$

and $\rho_\mu = \Phi - H_\mu = \Phi - \frac{1}{2}\left(\frac{2\Phi}{N} - \log(N)\right)$

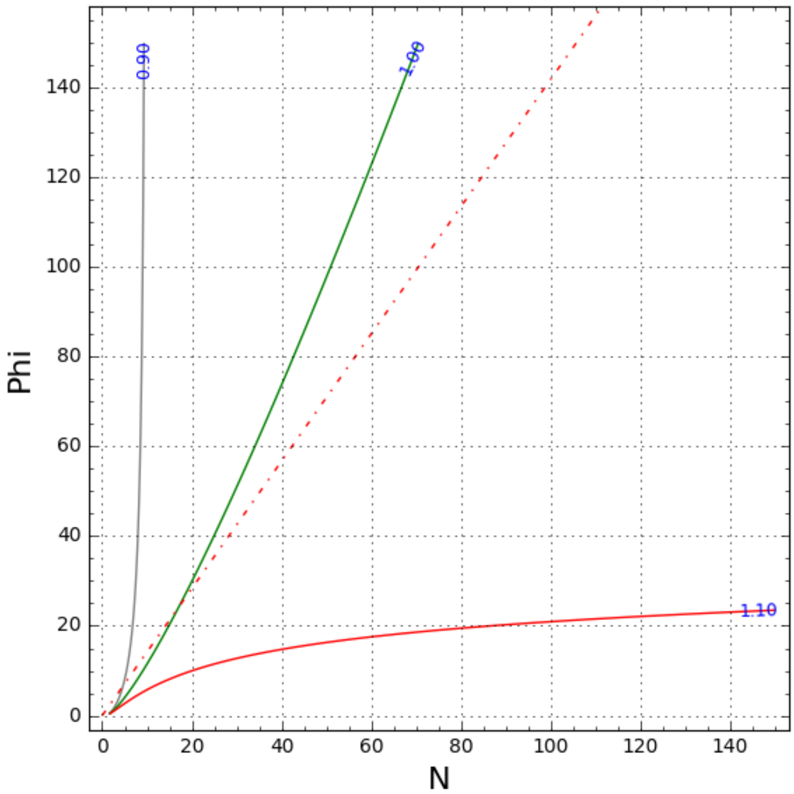
Reordering

$\rho_{\mu} = \Phi\left(\frac{N-1}{N}\right) + \frac{\log(N)}{2}$

```
x,y=var('x','y')
contour_plot(log(x)/2+y*(x-1)/x,(x,1,150),(y,0,150), fill=False, plot_points=450,
contours=[0,2,4,6,8,10],cmap =['grey','yellow','green','blue','red'],
labels=True)+plot(log(2*pi*exp(1))/2*x,(x,0,150),color='red',ymax=155)
```




```
x,y=var('x','y')
var('rho_mu,Phi')
latex(rho_mu/Phi)
contour_plot((log(x)/2+y*(x-1)/x)/y,(x,1,150),(y,0,150), fill=False, plot_points=450,
contours=[0.9,1,1.1],cmap =['grey','yellow','green','blue','red'], labels=True,
axes_labels=['N','Phi'], gridlines=True)+plot(log(2*pi*exp(1))/2*x,
(x,0,150),color='red',ymax=155,linestyle='-.')
```



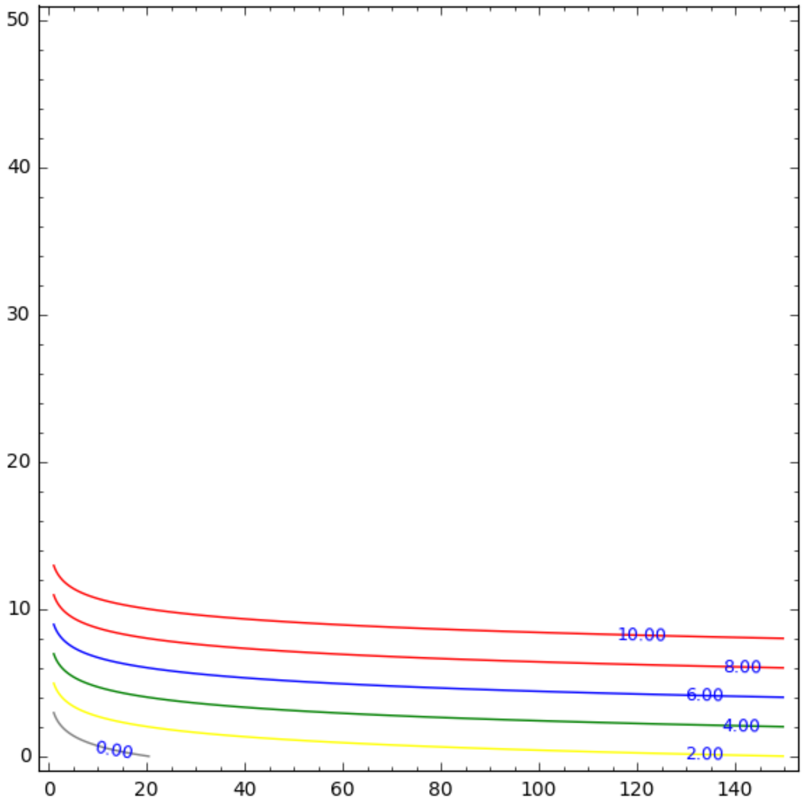
If original IID is Binomial with parameters *n,p*

$$\rho_\mu = N\left(-\left\langle \log\binom{n}{i}\right\rangle - n\log(p^p q^q)\right) - \frac{1}{2}(\log(2\pi e) + \log(npq) - \log(N))$$

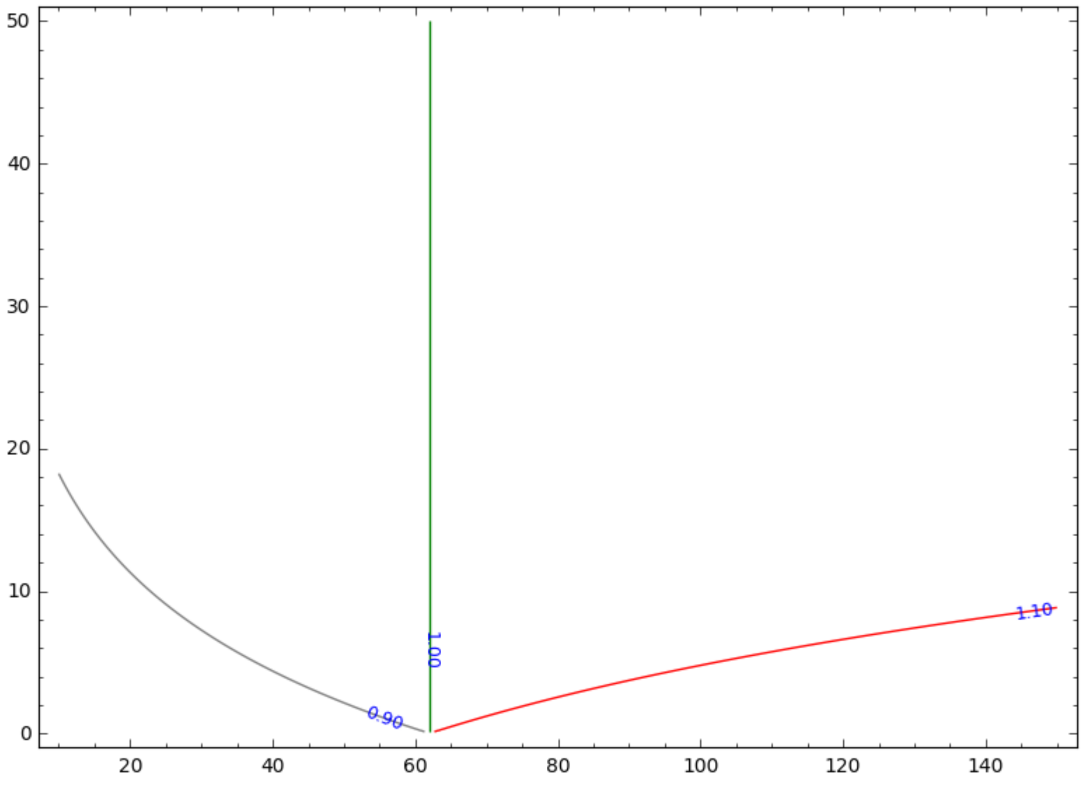
Reordering

$$\rho_\mu = \Phi + \log\left(\sqrt{N}\right) - \log\left(\sqrt{2\pi e}\right) - \log(\sigma)$$

```
x,y=var('x','y')
contour_plot(log(x)+y-log(sqrt(2*pi*exp(1)))-log(5),(x,1,150),(y,0,50), fill=False,
plot_points=450, contours=[0,2,4,6,8,10],cmap =['grey','yellow','green','blue','red'],
labels=True, aspect_ratio=3)
```




```
x,y=var('x','y')
contour_plot((log(x)+y-log(sqrt(2*pi*exp(1)))-log(15))/y, (x,10,150), (y,0,50),fill=False,
plot_points=450, contours=[0.9,1,1.1],cmap =['grey','yellow','green','blue','red'],
labels=True,aspect_ratio=2)
```

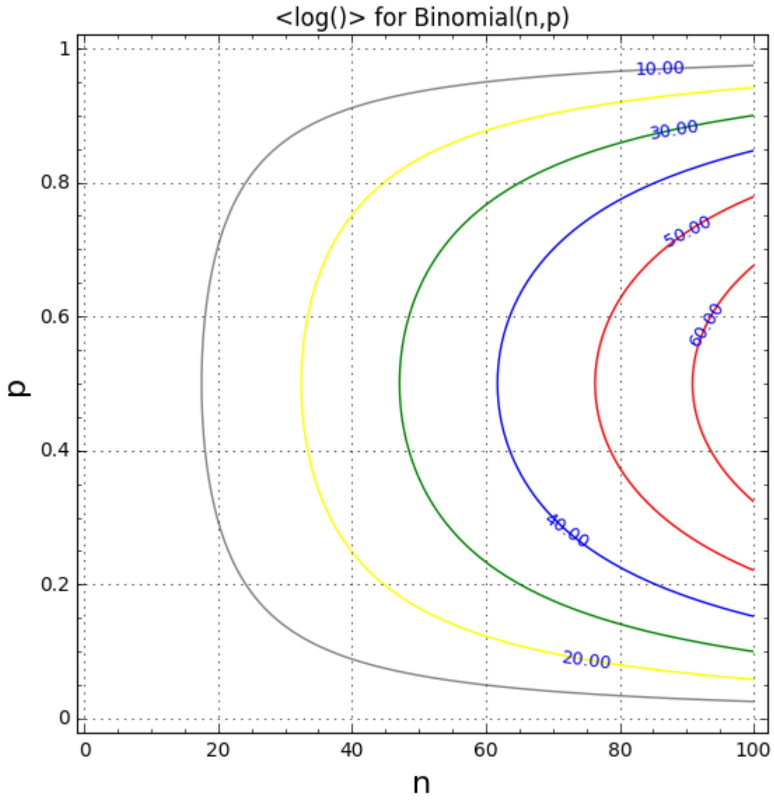


<log combinatorial number>

```
def expectec(n,p=0.5):
```

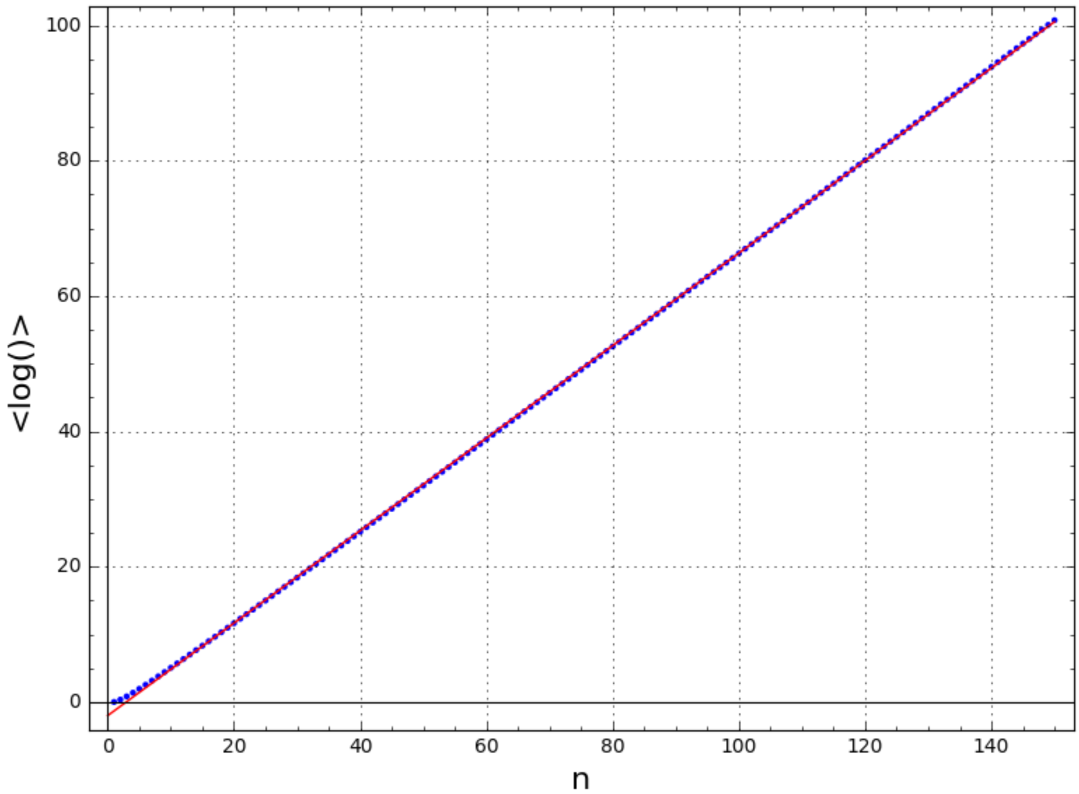
```
    '''
    returns the mean of the log of the combinatorial number
    for the Binomial(n,p)
    '''
    import scipy.stats
    binom_dist = scipy.stats.binom(n,p)
    n=int(n)
    total=0

    for i in range(n+1):
        total+= log(binomial(n,i))*binom_dist.pmf(i) #binomial(n,i)*p**i*(1-p)**(n-i))
    return total

def H_bin(n,p=0.5,method=1):
    '''
    Returns the Shannon entropy for
    Binomial(n,p)
    method=1 theoretical <>
    method=2 computes actual <>
    method=0 returns a tuple (m1,m2)
    '''


    H1=-1.0*(expectec(n,p)+n*log(p**p*(1-p)**(1-p)))

    if method==1:
        return N(H1)

    import scipy.stats
    binom_dist = scipy.stats.binom(n,p)
    H2=sum([-(i*log(p)+(n-i)*log(1-p))*binom_dist.pmf(i) for i in range(n +1)])-
expectec(n,p)
    if method==2:
        return N(H2)

    return (H1,H2)
```

```
n,p=var('n','p')
contour_plot(expectec, (n,1,100),(p,0,1),fill=False, plot_points=100, cmap
=['grey','yellow','green','blue','red'], labels=True,aspect_ratio=100,frame=True,
axes_labels=['n','p'], gridlines=True, title='<log()> for Binomial(n,p)')
```




```
data=[(i,expectec(i)) for i in range(1,151)]
var('a,b,x')
fit=find_fit(data,a*x+b,variables=[x])

xdata=[i[0] for i in data]
ydata=[i[1] for i in data]
corr=r.cor(xdata,ydata)
print('<log()> = %g·n+%g)' % (float(fit[0].rhs()),float(fit[1].rhs())))
print('R= %s' % corr)

list_plot(data)+plot(fit[0].rhs()*x+fit[1].rhs(),(x,0,150),color='red',frame=True,
axes_labels=['n','<log()>'], gridlines=True)
```

```
<log()> = 0.683441·n+-2.00887)
R= [1] 0
```




```
n,p=var('n','p')
contour_plot(H_bin, (n,1,100),(p,0,1),fill=False, plot_points=100, cmap
=['grey','yellow','green','blue','red'], labels=True,aspect_ratio=100.0,frame=True,
axes_labels=['n','p'], gridlines=True,title='H_B(n,p)')
```