



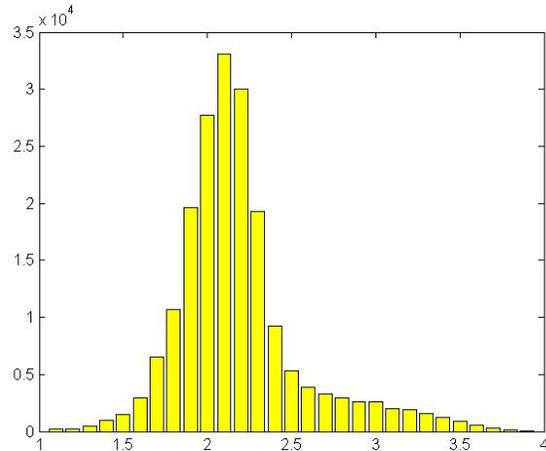
Opinionated
Lessons
in Statistics

by Bill Press

#25 Fitting Models to Counts

Let's turn from (x,y,σ) data to data that comes as counts of things.

Two common examples are “binned values” (histograms) and contingency tables.



	C_0	C_1
f_0	n_{00}	n_{01}
f_1	n_{10}	n_{11}

Counts are distributed according to (in general, unknown) probabilities p_i or p_{ij} across the bins or table entries. The model (with parameters maybe) predicts the p 's.

$$n_i \sim \text{Binomial}(N, p_i) \quad \text{or more precisely, } \{n_i\} \sim \text{Multinomial}(N, \{p_i\})$$

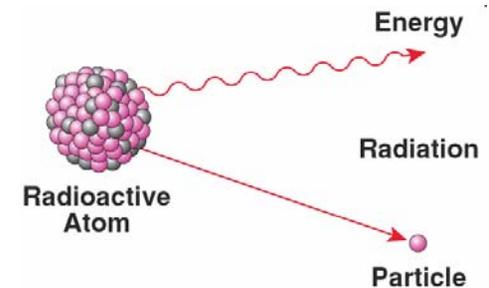
For histograms (but not necessarily contingency tables) one commonly has

$$n_i \ll N \Rightarrow p_i \ll 1 \quad \text{for all } i$$

$n_i \ll N \Rightarrow p_i \ll 1$ for all i implies that counts are (close to) Poisson distributed

Binomial(n, N, p) \Rightarrow

$$\begin{aligned} P(n) &= \frac{N!}{n!(N-n)!} p^n (1-p)^{N-n} \\ &= \frac{1}{n!} \frac{N!}{(N-n)!} p^n e^{(N-n) \ln(1-p)} \\ &\approx \frac{1}{n!} (Np)^n e^{-(Np)} \\ &\sim \text{Poisson}(Np) \end{aligned}$$



Sometimes this is not even an approximation, but exact because of how the data is gathered. Everyone's favorite example: radioactive decays.

It depends on whether N was a constraint, or "just happened". We will return to this issue when we discuss contingency tables: details of the exact protocol can subtly affect the statistics of the result.

Also recall, $x \sim \text{Poisson}(\lambda) \Rightarrow \mu(x) = \lambda, \text{Var}(x) = \lambda$

The histogram we just saw is biological:
It's the distribution of log10 of exon lengths in human.

```

exl ogl en = log10(cel l 2mat(g. exonl en));
[count cbi n] = hi st(exl ogl en, (1: . 1: 4));
count = count(2: end-1); % trim ends, which have overflow counts
cbi n = cbi n(2: end-1);
ecount = sqrt(count+1); ← "pseudo-count"
bar(cbi n, count, 'y')

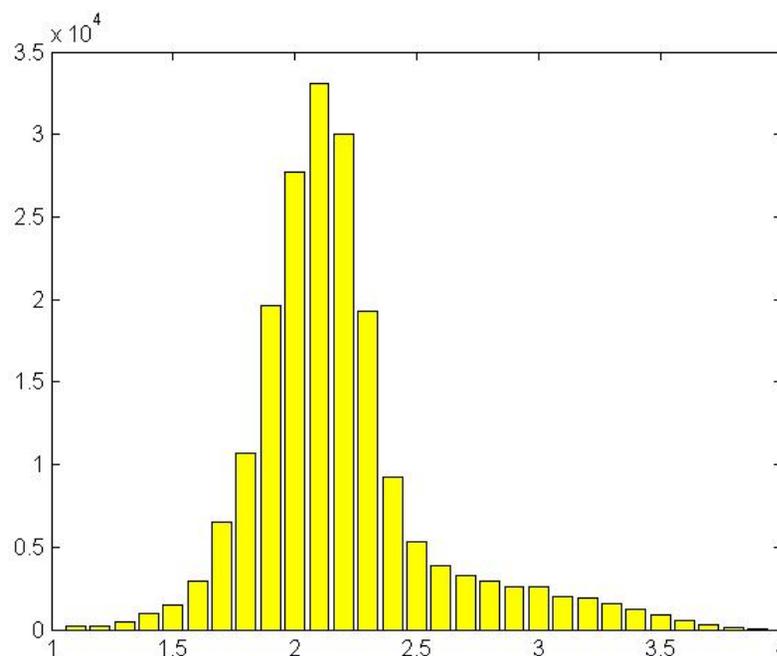
```

$$\chi^2 = \sum_i \frac{(n_i - Np_i)^2}{Np_i} \quad \begin{array}{l} \text{"Pearson"} \\ \text{"(O-E)^2 / E"} \end{array}$$

but people often use

$$\chi^2 = \sum_i \frac{(n_i - Np_i)^2}{n_i + \alpha} \quad \begin{array}{l} \text{"modified"} \\ \text{Neyman"} \\ \text{"(O-E)^2 / O_m"} \end{array}$$

← "pseudo-count"



Why do they do this?

1. It's the numerator that drives the fit to the data. Denominator shouldn't matter much.
2. Many NLS algorithms/packages require σ 's as input and can't fit them from a model.
3. Having the model in the denominator makes it more likely that you'll converge to a spurious local minimum (never recover from an iteration with a very small p_i).

Two asides:

1. The pseudocount can be thought of as resulting from a power-law prior on λ

```
In[1]:= poi = lam^n Exp[-lam]
```

```
Out[1]= e-lam lamn
```

$$P(n, \lambda) \propto \lambda^n e^{-\lambda}$$

```
In[2]:= Solve[D[poi, lam] == 0, lam]
```

$$\frac{dP}{d\lambda} = 0 \Rightarrow \lambda = n$$

```
Out[2]= {{lam -> n}}
```

```
In[4]:= Solve[D[poi lam^alpha, lam] == 0, lam]
```

$$\frac{d(P\lambda^\alpha)}{d\lambda} = 0 \Rightarrow \lambda = n + \alpha$$

```
Out[4]= {{lam -> alpha + n}}
```

2. Matlab for some reason lacks of a weighted nonlinear fit function! We can make one out of their unweighted function nlinfit (they have a help page telling how to do this):

```
function [beta r J Covar mse] = nlinfitw(x, y, sig, model, guess)
yw = y./sig;
modelw = @(b, x) model(b, x) ./ sig;
[beta r J Covar mse] = nlinfit(x, yw, modelw, guess);
Covar = Covar ./ mse; % undo Matlab's perhaps well-intentioned scaling
```

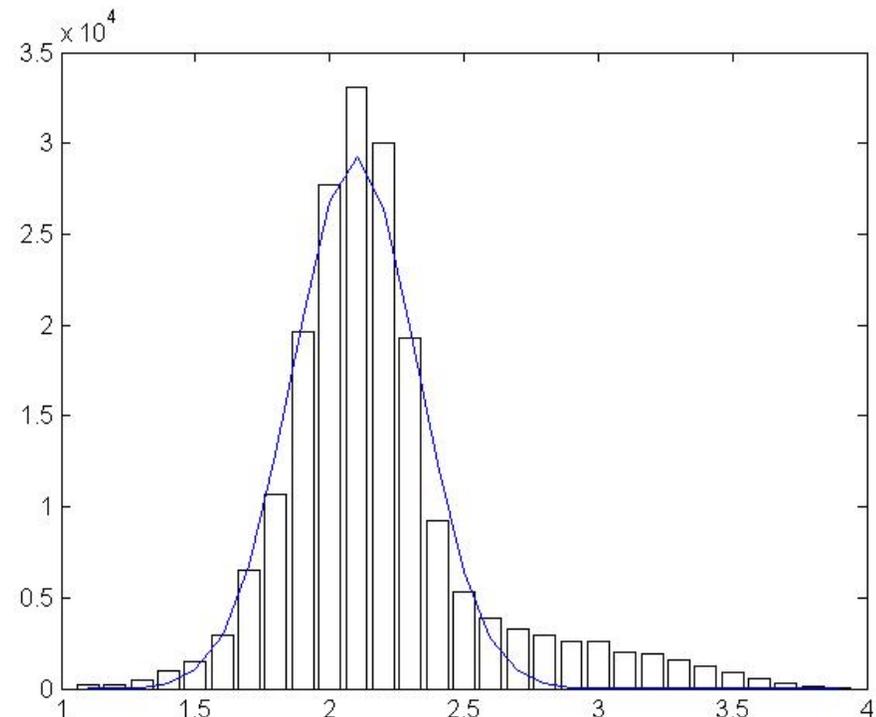
The Neyman χ^2 (previous slide) fits into this common interface to nonlinear least square (NLS), while the Pearson (truer) χ^2 doesn't.

OK, we're ready to fit a model to the exon length data.

Fit a single Gaussian (of course, it's in log space)

```
model oneg = @(b, x) b(1). *exp(-0.5. *((x-b(2)). /b(3)). ^2);
guess = [3.5e4 2.1 .3];
[bfi t r J Covar mse] = nli nfi tw(cbi n, count, ecount, model oneg, guess);
bfi t, Covar, mse
stderr = sqrt(di ag(Covar))
pl ot(cbi n, model oneg(bfi t, cbi n), ' b' )
bfi t =
    29219         2.0966         0.23196
Covar =
    8513   -0.0012396   -0.02769
   -0.0012396   3.1723e-007   9.833e-009
   -0.02769   9.833e-009   2.1986e-007
mse =
    849.37 ← "mean square error"
stderr =
    92.266
    0.00056323
    0.0004689
```

mse is just another name for χ^2/N ,
so it should be ~ 1 for a good fit



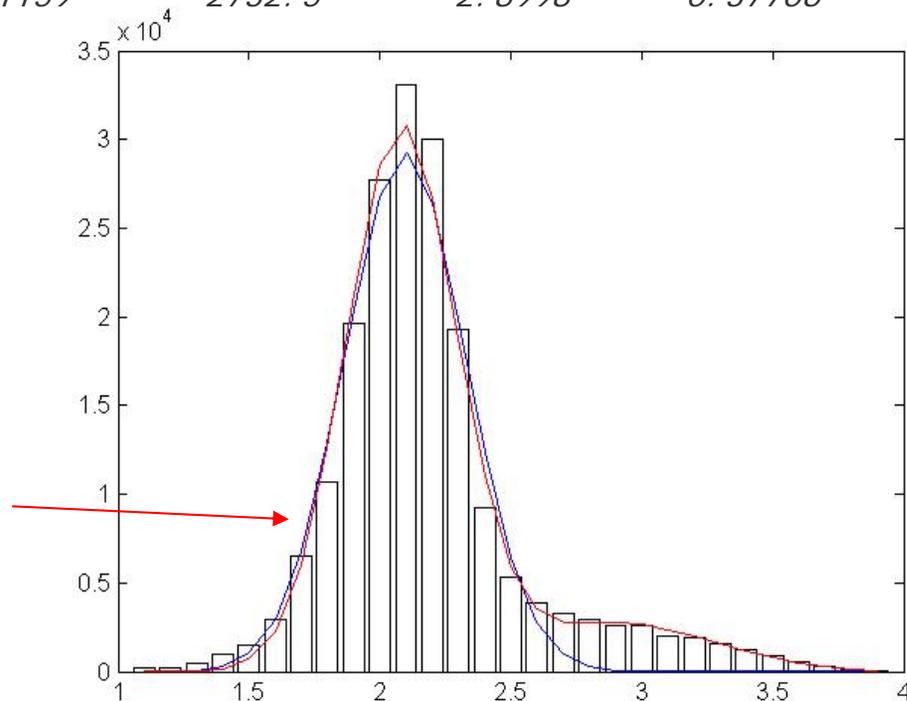
Fit sum of two Gaussians:

This time, we'll put the model function into an external file:

```
function y = model twog(b, x)
y = b(1). *exp(-0.5. *(x-b(2)). /b(3)). ^2) + ...
    b(4). *exp(-0.5. *(x-b(5)). /b(6)). ^2);
```

```
guess2 = [guess 3000 3. 0.5];
[bfi t2 r2 J2 Covar2 mse2] = nli nfi tw(cbi n, count, ecount, @model twog, guess2);
bfi t2, sqrt(di ag(Covar2)), mse2
pl ot(cbi n, model twog(bfi t2, cbi n), ' r' )
hol d off
bfi t2 =
    30633      2. 0823      0. 21159      2732. 5      2. 8998      0. 37706
ans =
    99. 609
    0. 00069061
    0. 00056174
    23. 667
    0. 0069429
    0. 0041877
mse2 =
    163. 44
```

Although it seems to capture the data qualitatively, this is still a bad fit. Whether this should worry you or not depends completely on whether you believe the model should be “exact”



We keep getting these big mse's!

Let's verify that $mse \sim 1$ is what you should get for a perfect model:

```
perfect = 2.0 .* randn(10000,1) + 4. ;
[count cbin] = hist(perfect, (-1:.2:9));
count = count(2:end-1);
cbin = cbin(2:end-1);
ecount = sqrt(count+1);
[bfit r J Sigma mse] = nlinfitw(cbin, count, ecount, model oneg, guess);
bfit, Sigma, mse
bfit =
    393.27    4.0159    2.0201
Sigma =
    25.784   -0.0005501   -0.057248
   -0.0005501   0.00046937   3.5555e-006
   -0.057248   3.5555e-006   0.00032997
mse =
    0.95507
chisq = numel(count)*mse
df = numel(count)-3;
pvalue = chi2cdf(chisq, df)
chisq =
    46.799
pvalue =
    0.56051
```

Let's see if 0.955 is actually good enough:

by definition, mse times number of bins equals chi-square

three fitted parameters

yep, good enough!

If you expect the model to be exact, then take the p-value seriously.

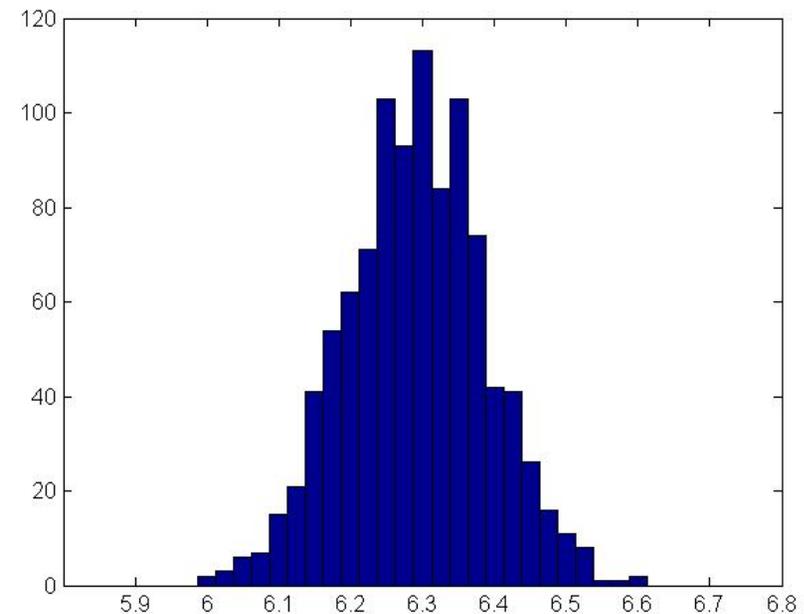
If you don't, it is called "chi-by-eye" (term used as an insult in the physical sciences).

It's OK when the intent of the model is to summarize main features of the data without necessarily fitting it exactly.

Uncertainty of a derived quantity: Ratio of the areas of the two Gaussians

Sample from the posterior distribution

```
samp = mvnrnd(bf1 t2, Covar2, 1000);  
samp(1:5, :)  
ans =  
    30430    2.082    0.21203    2754.6    2.8975    0.37636  
    30421    2.0829    0.21213    2701.1    2.9026    0.37738  
    30645    2.0815    0.21125    2775.3    2.8969    0.37969  
    30548    2.0822    0.21229    2714.7    2.9011    0.37712  
    30607    2.0826    0.21175    2718    2.9016    0.37779  
  
funcsam = (samp(:, 1) .* samp(:, 3)) ./ (samp(:, 4) .* samp(:, 6));  
funcsam(1:5)  
ans =  
    6.2234  
    6.3307  
    6.1437  
    6.3346  
    6.3116  
  
hist(funcsam, [5.9:0.025:6.7]);  
mu = mean(funcsam)  
sigma = std(funcsam)  
mu =  
    6.2911  
sigma =  
    0.096832
```



Bootstrap

```
function mu = areasboot(data)
samp = randsample(data, numel(data), true);
[count cbin] = hist(samp, (1:1:4));
count = count(2:end-1);
cbin = cbin(2:end-1);
ecount = sqrt(count+1);
guess = [3.5e4 2.1 .3 3000 3. 0.5];
[bfitr J Covar mse] = nlinfitw(cbin, count, ecount, @model twog, guess);
mu = (bfitr(1)*bfitr(3))/(bfitr(4)*bfitr(6));
```

list of individual exon lengths.
Notice that we resample before
(re-)binning.

```
areas = arrayfun(@(x) areasboot(exloglen), (1:1000));
```

takes about 1 min on my machine

```
mean(areas)
```

```
std(areas)
```

```
ans =
```

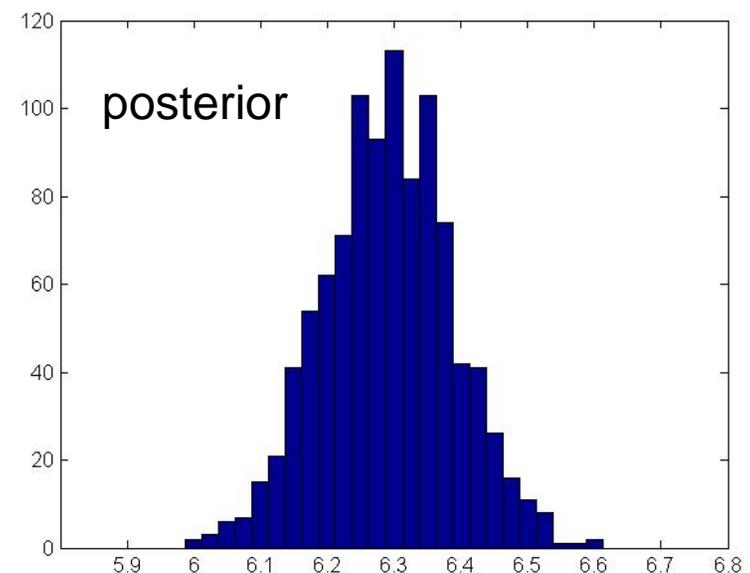
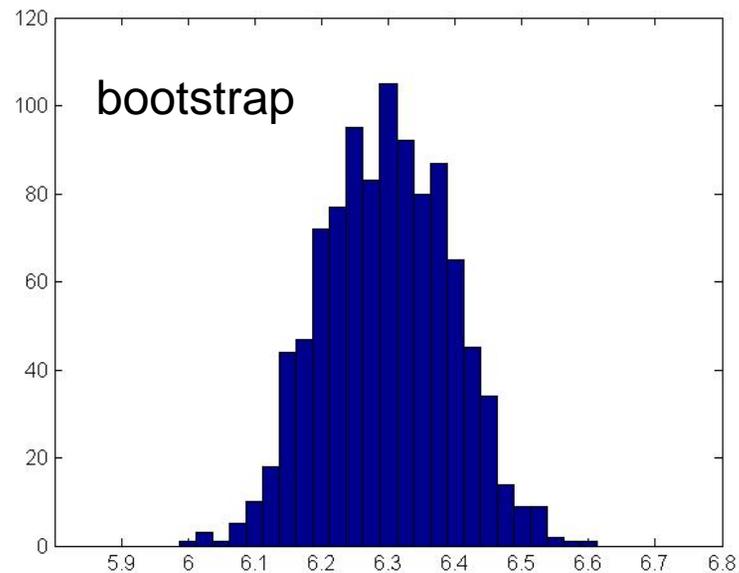
```
6.2974
```

```
ans =
```

```
0.095206
```

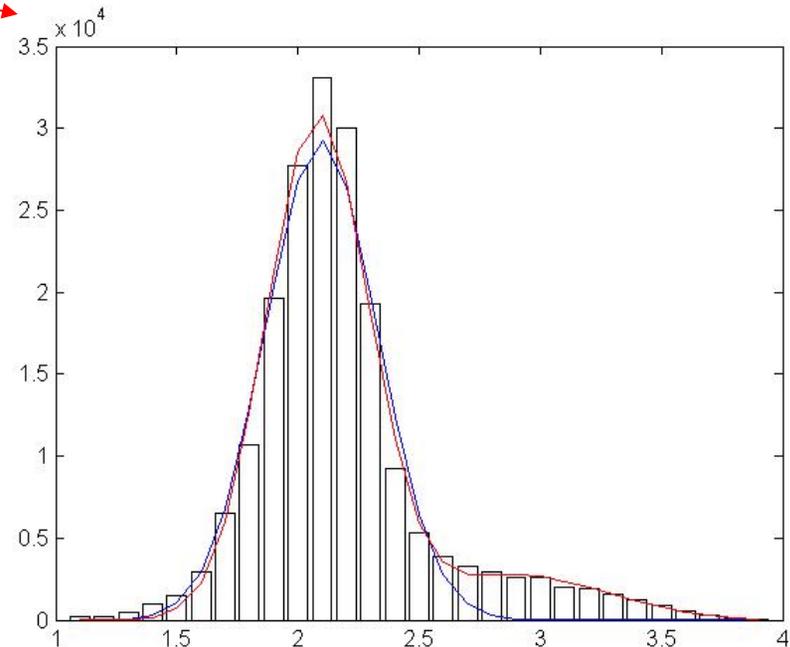
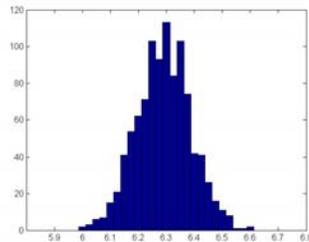
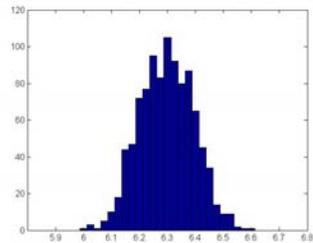
```
hist(areas, [5.9:0.025:6.7]);
```

recall, sampling from the posterior gave:



Everything works out nicely here because we have lots of data.
We are in “asymptopia”!

Ratio of areas = 6.3 ± 0.1



But remember that we did “chi by eye” here. (The model is not a perfect fit.)

Our value and uncertainty “are what they are” within the imperfect model.

They have no **magical power** to peer into the underlying heart of nature!

We’ll come back to this data set later.