CS395T
Computational Statistics with Application to Bioinformatics

Prof. William H. Press
Spring Term, 2011
The University of Texas at Austin

Lecture 21

If we generalize to contingency tables other than 2x2, Dirichlet is the relevant conjugate distribution to Multinomial

Multinomial distribution (you can derive it by "repeated binomial" or combinatorics as we did earlier):

$$P(n_1, n_2, \dots | N, q_1, q_2, \dots) = \frac{N!}{n_1! n_2! \cdots} q_1^{n_1} q_2^{n_2} \cdots , \qquad (\sum n_i = N, \ \sum q_i = 1)$$

Conjugate distribution, using conjugate priors:

$$P(q_1, q_2, \dots | N, n_1, n_2, \dots) \propto q_1^{n_1+\alpha_1} q_2^{n_2+\alpha_2} \cdots$$

the Dirichlet distribution

Normalization turns out to be:

$$P(q_1, q_2, \dots | N, n_1, n_2, \dots) = \frac{\Gamma(N + \alpha_1 + 1 + \alpha_2 + 1 + \cdots)}{\Gamma(n_1 + \alpha_1 + 1)\Gamma(n_2 + \alpha_2 + 1) \cdots} q_1^{n_1+\alpha_1} q_2^{n_2+\alpha_2} \cdots$$

Rather amazingly, there is a simple way to generate a (non-independent) set of ***q*** deviates from an independent set of Gamma deviates:

$$y_i \sim \text{Gamma}(n_i + \alpha_i + 1), \quad p(y) = \frac{y^{n_i+\alpha_i} e^{-y}}{\Gamma(n_i + \alpha_i + 1)} \qquad q_i = y_i \Big/ \sum_i y_i$$

(In fact, in the case with $I=2$, this is how Beta deviates [previous slide] are usually generated.)

So let's reanalyze assuming that the condition (column) marginals were fixed by the protocol, and we Bayes-sample the row probabilities:

	C_0	C_1
f_0	8	3
f_1	16	26

(You'll never believe my encapsulated function unless I go through an example!)

```
>> table = [8 3; 16 26]
table =
     8     3
    16    26
>> marfix = sum(table,1)'
marfix =
    24
    29
>> marvar = sum(table,2)'
marvar =
    11    42
>> gammas = gamrnd(marvar+1,1)
gammas =
   12.1000   44.5735
>> q = gammas ./ sum(gammas)
q =
    0.2135    0.7865
>> qmat = repmat(q,[size(table,2),1])
qmat =
    0.2135    0.7865
    0.2135    0.7865
>> tabout = mnrnd(marfix,qmat)'
tabout =
     4     8
    20    21
```

column marginals (transposed)

8	16
3	26

row marginals (transposed)

~Gamma(n_i+1)

generated (random) row probabilities q_i

finally, we generate multinomial deviates for each column, using the generated row probabilities

The reason everything is done in the transpose is because of the way that Matlab's mnrnd function expects its arguments to be shaped. Sorry about that!

In case it's not obvious, <u>sampling over</u> q is the same as <u>marginalizing over</u> q:

We generate a deviate pair *(f,q)* by choosing a *q*, and then, independently, an *f* <u>given</u> *q*, so

$$p(f,q) = p(q)\,p(f|q)$$

We then ignore *q*, so our *f* is drawn from the distribution

$$p(f) = \int p(f,q)\,dq = \int p(q)\,p(f|q)\,dq$$

which is the same as the desired marginalization

$$p(f) = \int p(f|q)\,p(q)\,dq$$

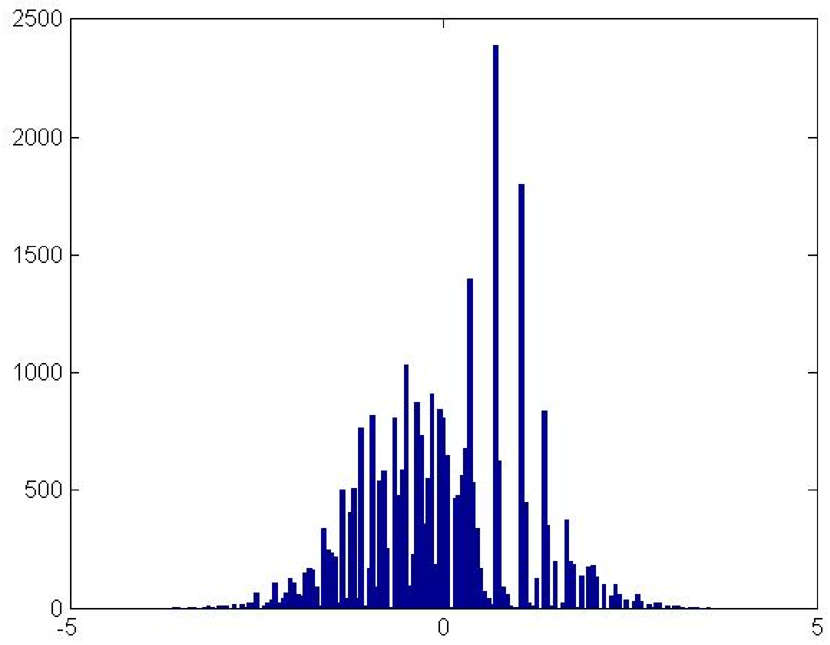
(This is so close to self-evident, that I'm not sure that the proof adds anything!)

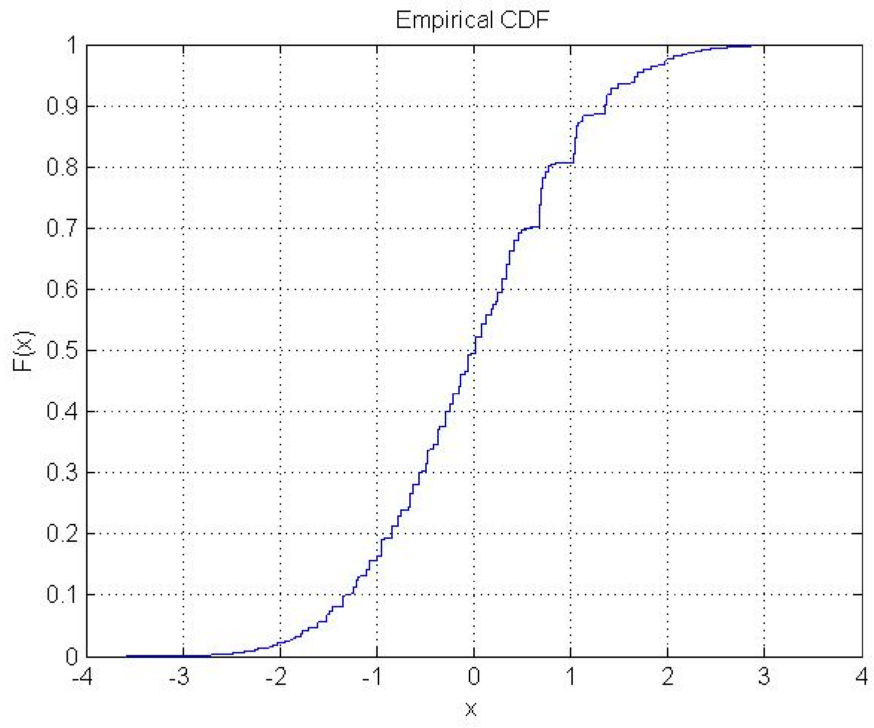
Encapsulate the sampling process into a function.
Then, generate a bunch of samples and look at their Wald statistics.

	C_0	C_1
f_0	8	3
f_1	16	26

```
function tabout = tabnullsamp(tabin)
marfix = sum(tabin,1)';
marvar = sum(tabin,2)';
q = gamrnd(marvar+1,1);
qmat = repmat(q./sum(q),[size(tabin,2),1]);
tabout = mnrnd(marfix,qmat)';

wald(table)
ans =
       2.0542

tabnullsamp(table), tabnullsamp(table)
ans =
     6     7
    18    22
ans =
     7     9
    17    20

wald(tabnullsamp(table))
ans =
       1.0392

samps = arrayfun(@(x) wald(tabnullsamp(table)), 1:30000);
hist(samps,-4:.05:4)
cdfplot(samps)
```

There are still discreteness effects (after all, these are integer tables), but they are less troubling:





```
pval = numel(samps(samps>=wald(table)))/numel(samps)
pvaltt = (numel(samps(samps>=wald(table)))+numel(samps(samps<= -wald(table))) )/numel(samps)
pval =
     0.022967
pvaltt =
     0.040067
```

one-tail vs. two-tail now much more reasonble

This is probably the most honest answer that we can get for the significance of this particular contingency table.

	C_0	C_1
f_0	8	3
f_1	16	26

Let’s reanalyze the maternal drinking data using the same methodology, but (as we did before) with the Pearson statistic:

```
function chis = pearson(table)
nhtable = sum(table,2)*sum(table,1)/sum(sum(table));
chis = sum(sum((table-nhtable).^2./nhtable));


table = [17066 14464 788 126 37; 48 38 5 1 1]'
table =
       17066           48
       14464           38
         788            5
         126            1
          37            1

pearson(table)
ans =
       12.082
```

transpose to make the unfixed marginals be the rows, as before

the (unrealistic, but don’t worry now) scenario is something like: case-control study where malformation-present came from hospitals, malformation-absent came from a door-to-door survey

columns are the “conditions”

```
samps = arrayfun(@(x) pearson(tabnullsamp(table)), 1:10000);
hist(samps,0:0.25:90)
```

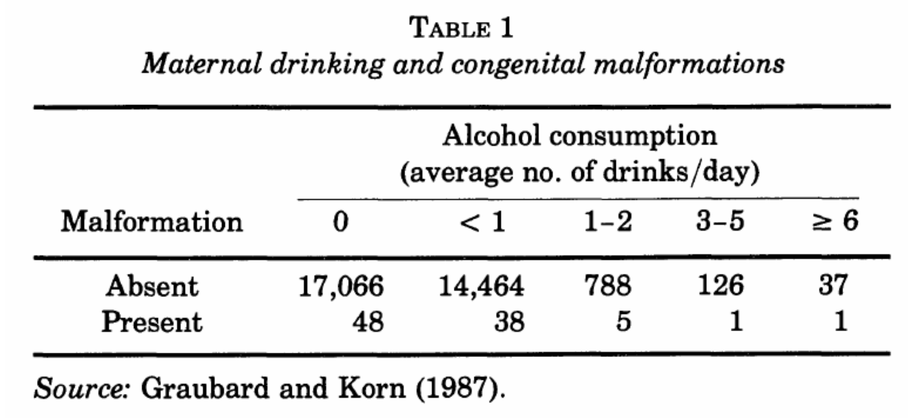
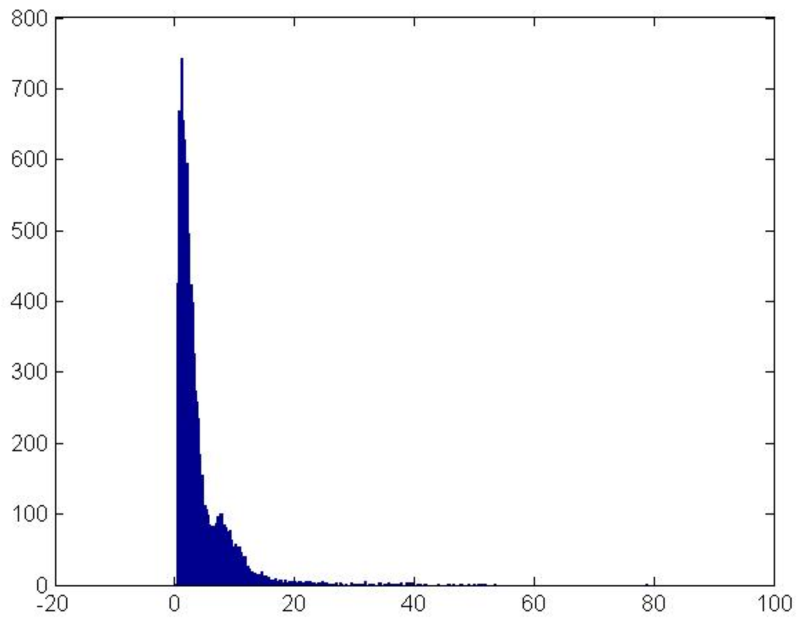


TABLE 1
Maternal drinking and congenital malformations

Malformation	Alcohol consumption (average no. of drinks/day) 0	< 1	1–2	3–5	≥ 6
Absent	17,066	14,464	788	126	37
Present	48	38	5	1	1

Source: Graubard and Korn (1987).

Giving the results



```
pval = numel(samps(samps>=pearson(table)))/numel(samps)
pval =
       0.0408
```

Why is this less significant than the difference-of-means analysis, which gave p=.015 or .011 (mean or square-mean)?

Because here we didn't use the fact that the factors were ordinal and quantitatively related. By virtue of using that information, and thereby "compressing" the rows, the difference-of-means was more powerful.

Actually, it seems likely that this data was a cross-sectional study with no fixed marginals. If so, a better sampling of the null hypothesis would be:

```
function tabout = tabnullsamp2(tabin)
marcol = sum(tabin,1);
marrow = sum(tabin,2);
ntot = sum(marcol);
q = gamrnd(marcol+1,1);
q = q./sum(q);
p = gamrnd(marrow+1,1);
p = p./sum(p);
pq = p * q;
tabout = reshape(mnrnd(ntot,pq(:)'),size(tabin));
```

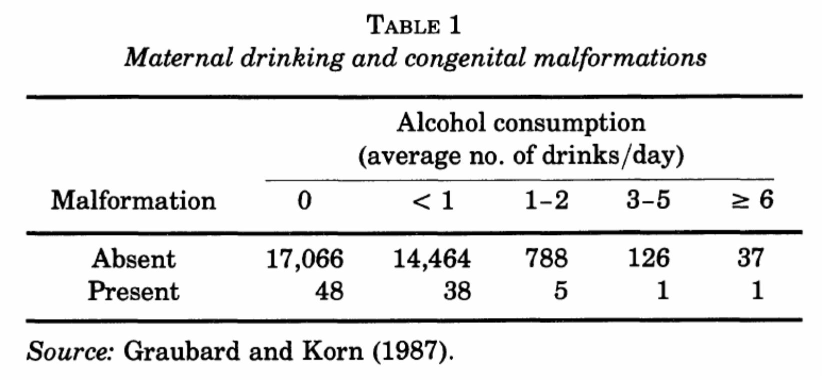


TABLE 1
Maternal drinking and congenital malformations

Malformation	0	< 1	1–2	3–5	≥ 6
Absent	17,066	14,464	788	126	37
Present	48	38	5	1	1

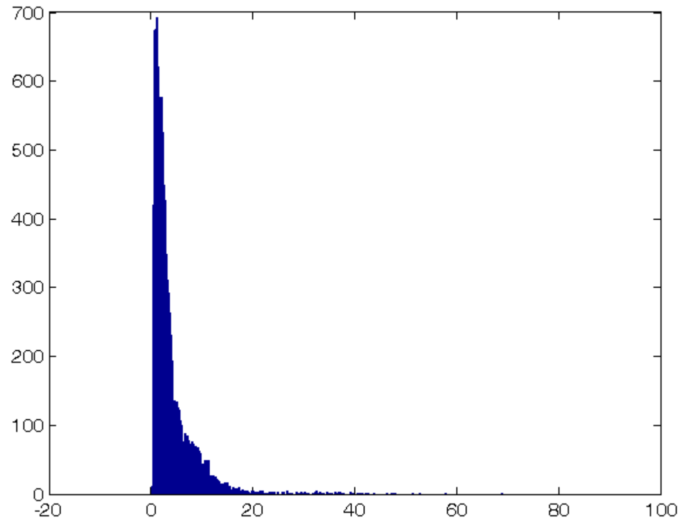
Alcohol consumption (average no. of drinks/day)

Source: Graubard and Korn (1987).

```
samps = arrayfun(@(x) pearson(tabnullsamp2(table)), 1:10000);
hist(samps,0:0.25:90)
pval = numel(samps(samps>pearson(table)))/numel(samps)
pval =
    0.0445
```

$\leftarrow$ different from previous: protocol matters!

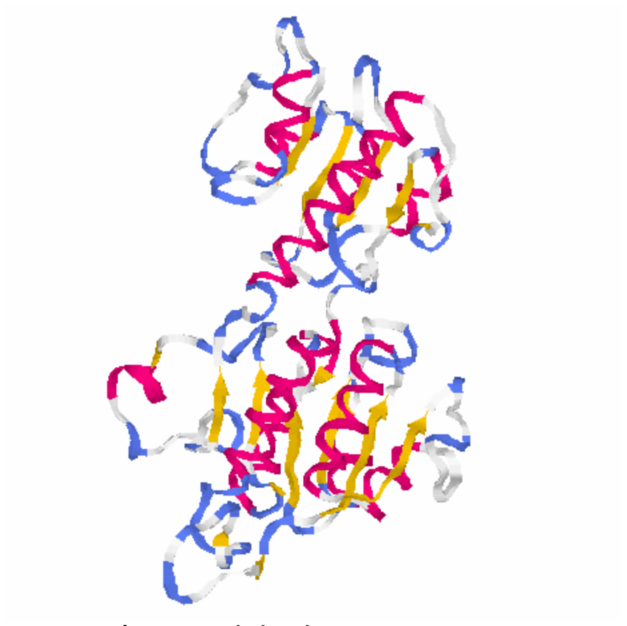
This would probably the "honest" answer if the table were nominal, not ordinal. But, as said, since it is ordinal, the previous analysis using difference of means is more powerful.



You now know all you need to know about contingency tables – and much more than almost everyone who uses them!

Information Theory Characterization of Distributions

As functioning machines, proteins have a somewhat modular three-dimensional (tertiary) structure. But the [more-or-less] complete instructions for making a protein are a one-dimensional sequence of characters representing amino acids.



lactate dehydrogenase,
showing alpha helices and beta sheets

For example:

261 characters, each in {A-Z} minus {BJOUXZ} (20 amino acids)

MAAACRSVKGLVAVITGGASGLGLATAERLVGQGASAVLLDLPNSG
GEAQAKKLGNNCVFAPADVTSEKDVQTALALAKGKFGRVDVAVNCA
GIAVASKTYNLKKGQTHTLEDFQRVLDVNLMGTFNVIRLVAGEMGQN
EPDQGGQRGVIINTASVAAFEGQVGQAAYSASKGGIVGMTLPIARDL
APIGIRVMTIAPGLFGTPLLTSLPEKVCNFLASQVPFPSRLGDPAEYAH
LVQAIIENPFLNGEVIRLDGAIRMQP*

(I picked this randomly in the human genome. A sequence search shows it to be “hydroxysteroid (17-beta) dehydrogenase “.)

How many proteins of length 261 are there? 20^{261} ? Yes, in a sense, but…

Shannon's key observation is that, if the characters in a message occur with unequal distribution p_i, then, <u>for long messages</u>, there is quite a sharp divide between rather probable messages and extremely improbable ones. Lets estimate the number of probable ones.

(The $\log_2$ of this number is the information content of the message, in bits.)

We estimate as follows

number of shuffled messages

$$2^B \approx \frac{M!}{\prod_i (Mp_i)!}$$

number of rearrangements of identical symbols i

$$\begin{aligned} B \ln 2 &\approx M \ln\left(\frac{M}{e}\right) - \sum_i (Mp_i) \ln\left(\frac{Mp_i}{e}\right) \\ &= \cancel{M \ln\left(\frac{M}{e}\right) - M\left(\sum_i p_i\right) \ln\left(\frac{M}{e}\right)} \boxed{- M \sum_i p_i \ln p_i} \\ &\equiv M\, H(\mathbf{p}) \end{aligned}$$

entropy in nats

$$n! \sim \sqrt{2\pi n}\left(\frac{n}{e}\right)^n$$

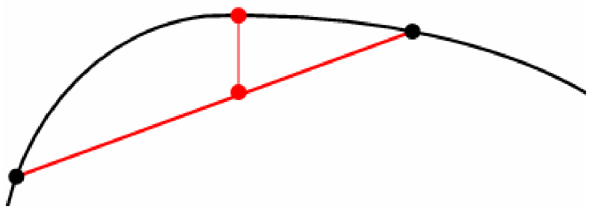
If you take all logs base 2, you get entropy in bits.
1 nat = 1.4427 bits.

$$H(\mathbf{p}) = -\sum_{i=1}^{N} p_i \ln p_i$$

Evidently positive for all **p**'s.

Minimum value zero when a single p_i=1.

Maximum when all the p_i's are equal:



$$\mathcal{L} = -\sum_i p_i \ln p_i + \lambda \left(\sum_i p_i - 1 \right)$$

$$0 = \frac{\partial \mathcal{L}}{\partial p_j} = -\ln p_j - 1 + \lambda$$

$$\Rightarrow \ln p_j = \lambda - 1 = \text{constant}$$

$$\max(H) = \ln N$$

$$H(\mathbf{p}) = -\sum_i p_i \ln p_i$$

Interpretations of the entropy of a distribution:

1. It's the (binary) message length of the maximally compressed message.

Because, just send a binary serial number among all the probable messages. (And do something else for the improbable ones – which will never happen and negligibly affect the mean length!)

2. It's the expected log cut-down in the number of remaining hypotheses with a feature distributed as $\mathbf{p}$, if we do an experiment that measures i

$$\langle \ln p_i \rangle = \textstyle\sum_i p_i \ln p_i = -H(\mathbf{p})$$

This is a figure of merit for experiments if, by repeated experiments, we want to get the number of remaining hypotheses down to 1.

3. It's the e-folding (or doubling) rate of capital for a fair game about which you have perfect predictive information.

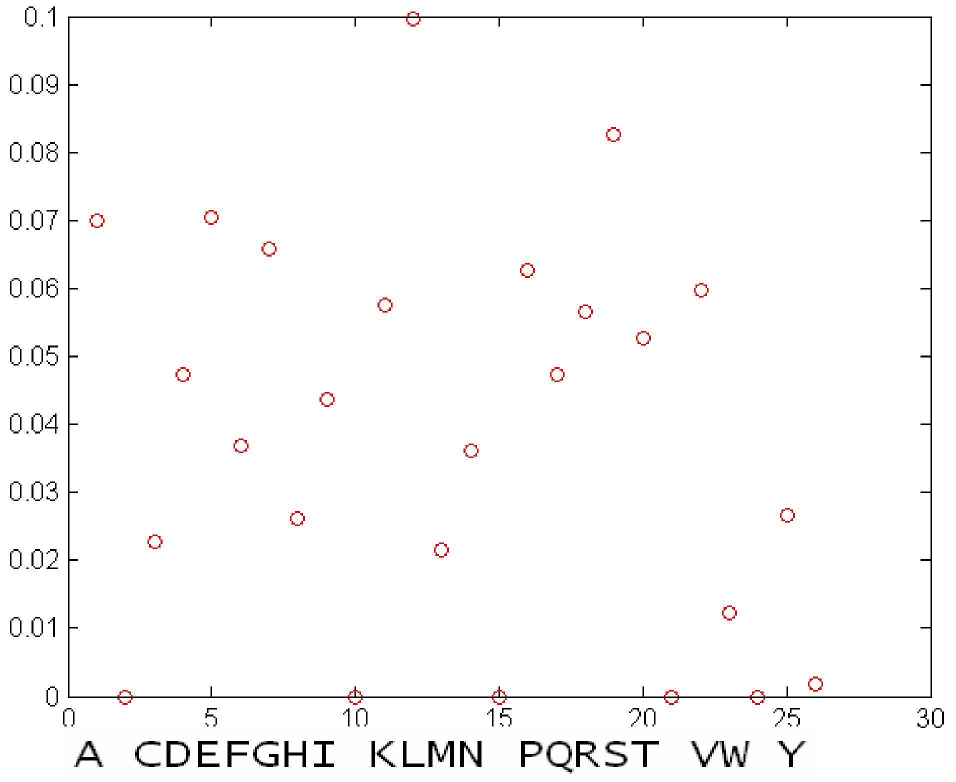
payoff (odds) $\xrightarrow{}$ $\langle o_i \rangle = p_i o_i = 1$

(This seems fanciful, but will make more sense when we discuss the case of partial predictive information.)

Example: what is the distribution of amino acids in human proteins?

```
load 'aadist_mono.txt';
mono = aadist_mono ./ sum(aadist_mono(:));
plot(mono(1:26),'or')
```

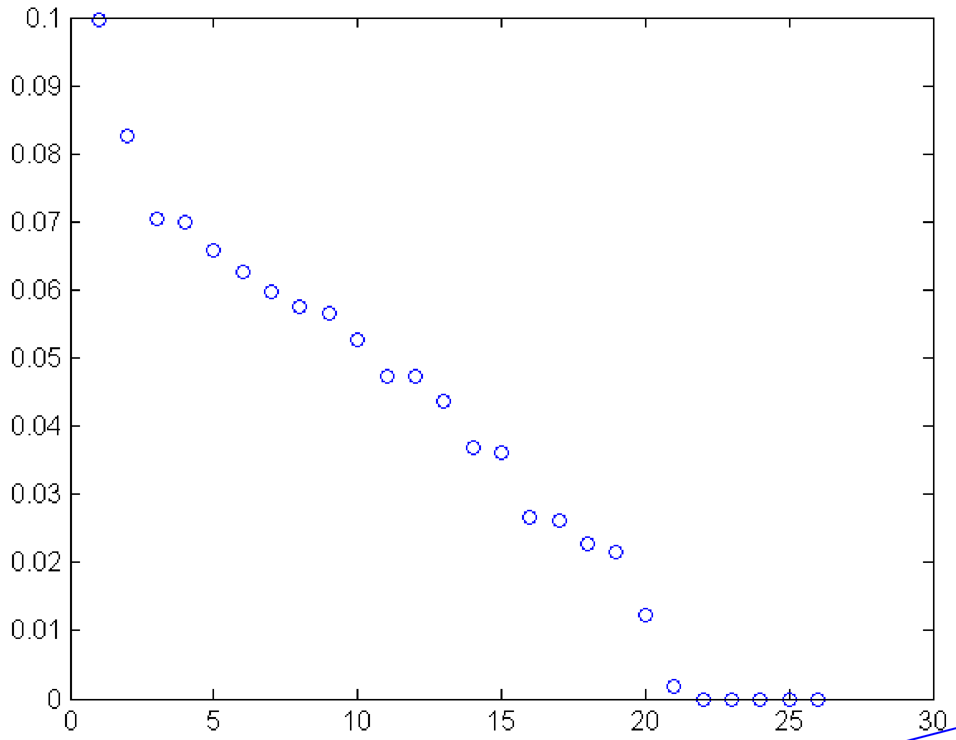
(file on course web site)



A	Alanine
R	Arginine
N	Asparagine
D	Aspartic acid (Aspartate)
C	Cysteine
Q	Glutamine
E	Glutamic acid (Glutamate)
G	Glycine
H	Histidine
I	Isoleucine
L	Leucine
K	Lysine
M	Methionine
F	Phenylalanine
P	Proline
S	Serine
T	Threonine
W	Tryptophan
Y	Tyrosine
V	Valine

Plot distribution in descending order. Also calculate entropy:

```
plot(sort(mono(1:26),'descend'),'ob')
```



```
entropy2 = @(x) sum(-x(:).*log(x(:)+1.e-99))/log(2);

h2bound = log(20)/log(2)
h2mono = entropy2(mono)
h2bound =
    4.3219
h2mono =
    4.1908
```

Notice that we flatten any structure in x when calculating the entropy.

maximum entropy that 20 characters could have

actual (single peptide) entropy of the AA's