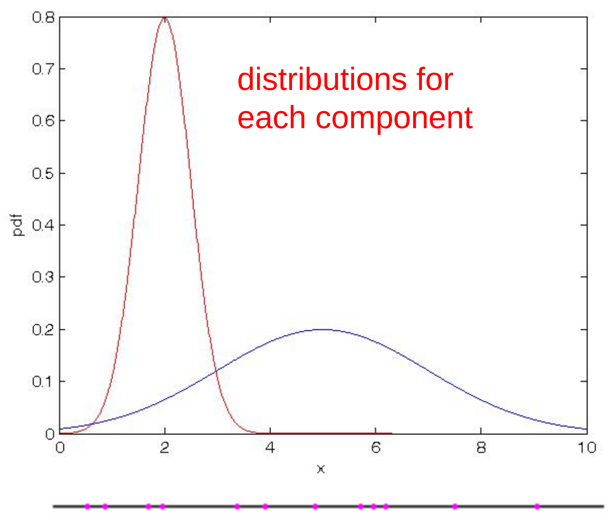
CS395T
Computational Statistics with Application to Bioinformatics

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

Lecture 15

Mixture Models



observed events (unknown mixture)

Suppose we have N independent events, i=1…N. Each might be from distribution 0 or distribution 1, but we don't know which (2-component mixture)

But we do know the respective probabilities for each i

$p_{0i} \equiv P(\mathbf{x}_i|0) \qquad p_{1i} \equiv P(\mathbf{x}_i|1)$

We want a (probabilistic) assignment of each event to 0 or 1.

Suppose $\mathbf{v} = (v_1, v_2, v_3, \dots, v_N)$ is an assignment of each event to a distribution

e.g., $\mathbf{v} = (1,1,0,1,0,0,0,1,\dots,1)$

Suppose s is the fraction of events in distribution 1, $s = P(v_i = 1)$

That is everything we need to know to write down a "forward" model for the probability of the data, given the (known and unknown) quantities:

$$P(\text{data}|\mathbf{v}, s) = \prod_{v_i=1} p_{1i} \times \prod_{v_i=0} p_{0i}$$

s doesn't enter directly, but it is a "hyperparameter" that affects the distribution of **v**'s

Now do the Bayes thing!

$$
\begin{aligned}
P(\mathbf{v}, s|\text{data}) &\propto P(\text{data}|\mathbf{v}, s)P(\mathbf{v}, s) \\
&= P(\text{data}|\mathbf{v}, s)P(\mathbf{v}|s)P(s) \\
&= \prod_{v_i=1} p_{1i} \times \prod_{v_i=0} p_{0i} \times s^{\#(v_i=1)}(1-s)^{\#(v_i=0)} P(s) \\
&= \prod_{v_i=1} p_{1i}s \times \prod_{v_i=0} p_{0i}(1-s) \times P(s)
\end{aligned}
$$

That is the complete model, usually too much to comprehend directly. Instead, we are usually interested in various marginalizations. For example:

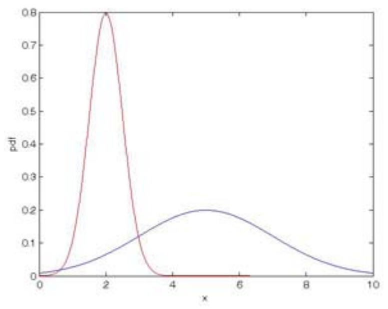
$$
P(s|\text{data}) \propto \sum_{v \in \mathbf{2}^N} \left[\prod_{v_i=1} p_{1i}s \times \prod_{v_i=0} p_{0i}(1-s) \times P(s) \right]
$$

key step is here:

$$
= \prod_i \underbrace{[p_{1i}s + p_{0i}(1-s)]}_{\text{prob of } i \text{ in the mixture distribution}} P(s)
$$

(multiply it out!)

prior on the mixture



Even more interesting is the marginalization that gives the assignment of each data point to one distribution or the other:

$$
\begin{aligned}
P(v_j = 1|\text{data}) &\propto \int \sum_{v \in \mathbf{2}^{N-1}} p_{1j} s \prod_{v_i=1, i\neq j} p_{1i} s \prod_{v_i=0, i \neq j} p_{0i}(1-s) P(s)\, ds \\
&= \int p_{1j} s \frac{P(s|\text{data})}{p_{1j}s + p_{0j}(1-s)} ds \\
&= \int \frac{p_{1j}s}{p_{1j}s + p_{0j}(1-s)} P(s|\text{data}) ds
\end{aligned}
$$

$$
\boxed{
\begin{aligned}
P(s|\text{data}) &\propto \sum_{v\in\mathbf{2}^N} \left[\prod_{v_i=1} p_{1i}s \times \prod_{v_i=0} p_{0i}(1-s) \times P(s)\right] \\
&= \prod_i [p_{1i}s + p_{0i}(1-s)] P(s)
\end{aligned}
}
$$

and similarly

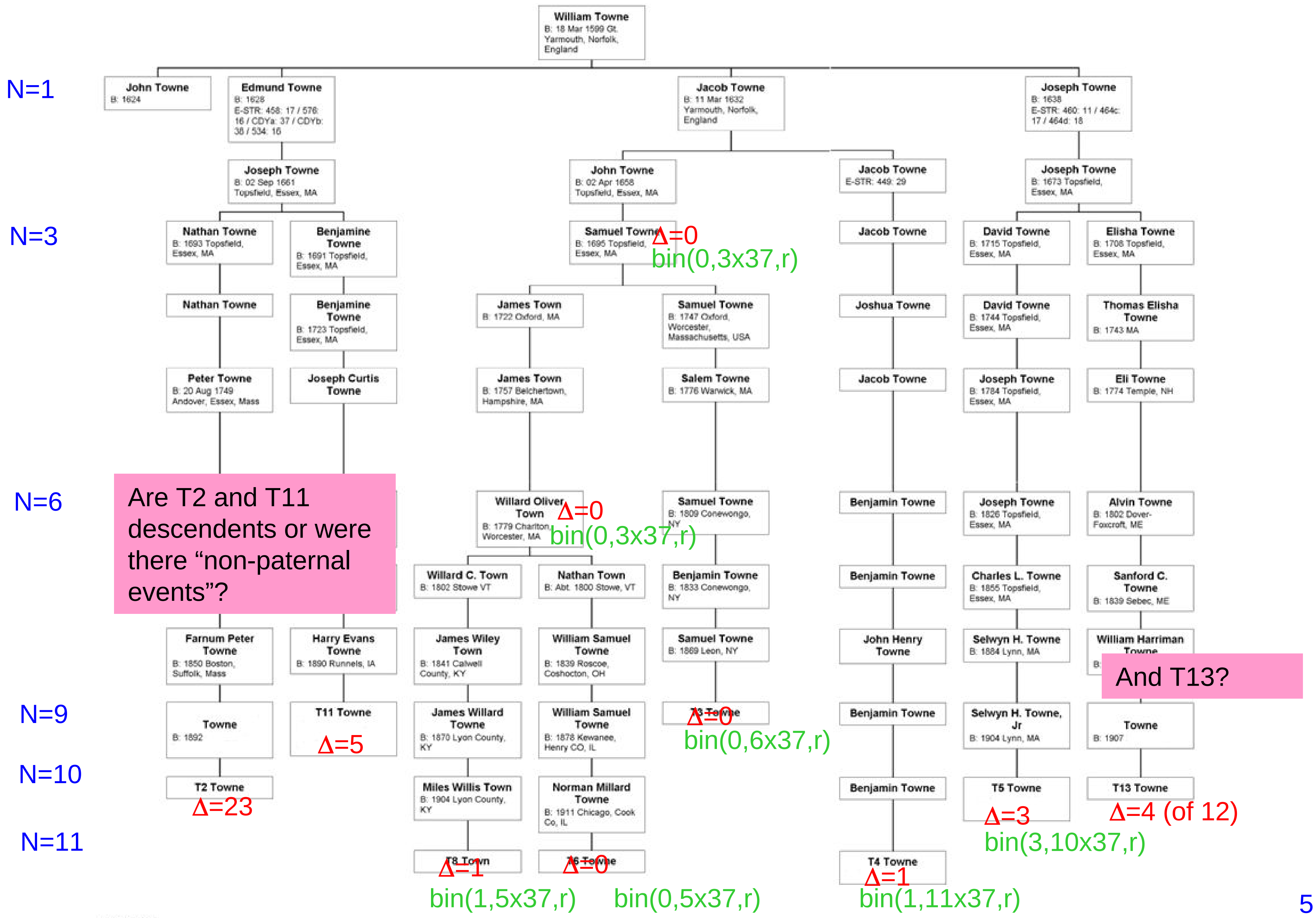
$$
P(v_j = 0|\text{data}) \propto \int \frac{p_{0j}(1-s)}{p_{1j}s + p_{0j}(1-s)} P(s|\text{data}) ds
$$

it’s just that data point’s relative probabilities in the two distributions, weighted by the mix probability

and then averaged over the mix probabilities

This is a very general idea, which can occur with many useful variations. Let’s apply to Towne…

Hi, guys! Remember us?


William Towne B: 18 Mar 1599 Gt. Yarmouth, Norfolk, England
N=1
John Towne B: 1624
Edmund Towne B: 1628 E-STR: 458: 17 / 576: 16 / CDYa: 37 / CDYb: 38 / 534: 16
Jacob Towne B: 11 Mar 1632 Yarmouth, Norfolk, England
Joseph Towne B: 1638 E-STR: 460: 11 / 464c: 17 / 464d: 18
Joseph Towne B: 02 Sep 1661 Topsfield, Essex, MA
John Towne B: 02 Apr 1658 Topsfield, Essex, MA
Jacob Towne E-STR: 449: 29
Joseph Towne B: 1673 Topsfield, Essex, MA
N=3
Nathan Towne B: 1693 Topsfield, Essex, MA
Benjamin Towne B: 1691 Topsfield, Essex, MA
Samuel Towne B: 1695 Topsfield, Essex, MA
Δ=0
bin(0,3x37,r)
Jacob Towne
David Towne B: 1715 Topsfield, Essex, MA
Elisha Towne B: 1708 Topsfield, Essex, MA
Nathan Towne
Benjamin Towne B: 1723 Topsfield, Essex, MA
James Town B: 1722 Oxford, MA
Samuel Towne B: 1747 Oxford, Worcester, Massachusetts, USA
Joshua Towne
David Towne B: 1744 Topsfield, Essex, MA
Thomas Elisha Towne B: 1743 MA
Peter Towne B: 20 Aug 1749 Andover, Essex, Mass
Joseph Curtis Towne
James Town B: 1757 Belchertown, Hampshire, MA
Salem Towne B: 1776 Warwick, MA
Jacob Towne
Joseph Towne B: 1784 Topsfield, Essex, MA
Eli Towne B: 1774 Temple, NH
N=6
Are T2 and T11 descendents or were there “non-paternal events”?
Willard Oliver Town B: 1779 Charlton, Worcester, MA
Δ=0
bin(0,3x37,r)
Samuel Towne B: 1809 Conewongo, NY
Benjamin Towne
Joseph Towne B: 1826 Topsfield, Essex, MA
Alvin Towne B: 1802 Dover-Foxcroft, ME
Willard C. Town B: 1802 Stowe VT
Nathan Towne B: Abt. 1800 Stowe, VT
Benjamin Towne B: 1833 Conewongo, NY
Benjamin Towne
Charles L. Towne B: 1855 Topsfield, Essex, MA
Sanford C. Towne B: 1839 Sebec, ME
Farnum Peter Towne B: 1850 Boston, Suffolk, Mass
Harry Evans Towne B: 1890 Runnels, IA
James Wiley Town B: 1841 Calwell County, KY
William Samuel Towne B: 1839 Roscoe, Coshocton, OH
Samuel Towne B: 1869 Leon, NY
John Henry Towne
Selwyn H. Towne B: 1884 Lynn, MA
William Harriman Towne
And T13?
N=9
Towne B: 1892
T11 Towne
Δ=5
James Willard Towne B: 1870 Lyon County, KY
William Samuel Towne B: 1878 Kewanee, Henry CO, IL
T3 Towne
Δ=0
bin(0,6x37,r)
Benjamin Towne
Selwyn H. Towne, Jr B: 1904 Lynn, MA
Towne B: 1907
N=10
T2 Towne
Δ=23
Miles Willis Town B: 1904 Lyon County, KY
Norman Millard Towne B: 1911 Chicago, Cook Co, IL
Benjamin Towne
T5 Towne
Δ=3
bin(3,10x37,r)
T13 Towne
Δ=4 (of 12)
N=11
T8 Town
Δ=1
bin(1,5x37,r)
T6 Towne
Δ=0
bin(0,5x37,r)
T4 Towne
Δ=1
bin(1,11x37,r)

Bayes and Bar Sinister

We can now understand that the Towne family problem is really a mixture model problem: Each VLSTR sample is either from a descendent of William Towne or from the descendent of a "non-paternal event". We are given an unknown mixture of such samples.

Arms of Sir Charles Beauclerk, 1st Duke of St Albans, bastard son of King Charles II by Nell Gwynn

Our model will have 3 unknown parameters:

r	mutation probability per locus per generation
c	non-paternal probability per generation
L	if non-paternal, number of generations back to LCA

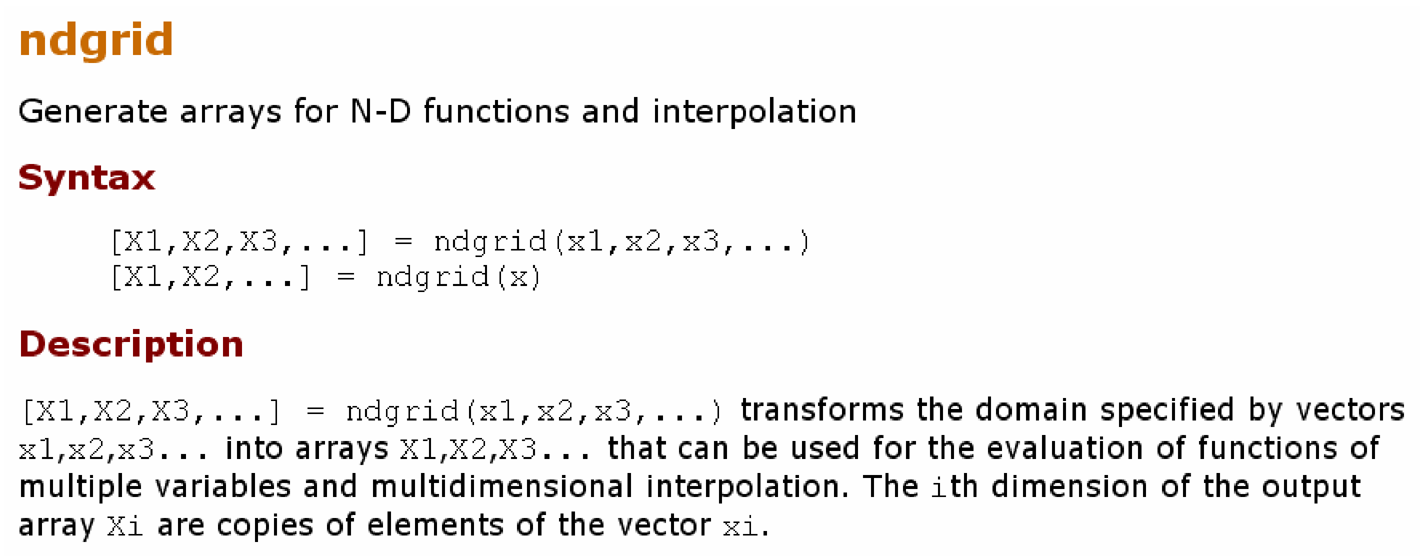
Modeling L as a constant is rather crude, but will illustrate the point. If this really mattered, we'd need to do a better job here.

The model is:

```
pmix = @(k,n,loci,r,c,lca) (1-c).^n * bin(k,n*loci,r) ...
          + (1-(1-c).^n) * bin(k,(n+lca)*loci,r);
model2 = @(r,c,lca) pmix(23,10,37,r,c,lca) .* pmix(5,9,37,r,c,lca)...
           .* pmix(0,3,37,r,c,lca).* pmix(0,3,37,r,c,lca)...
          .* pmix(1,5,37,r,c,lca) .* pmix(0,5,37,r,c,lca)...
          .* pmix(0,6,37,r,c,lca).* pmix(1,11,37,r,c,lca)...
          .* pmix(3,10,37,r,c,lca) .* pmix(4,10,12,r,c,lca) ./ r;
```

Notice that we now include all the data, especially clearly non-paternal T2.

So that we don't get lost in MATLAB semantics…



ndgrid

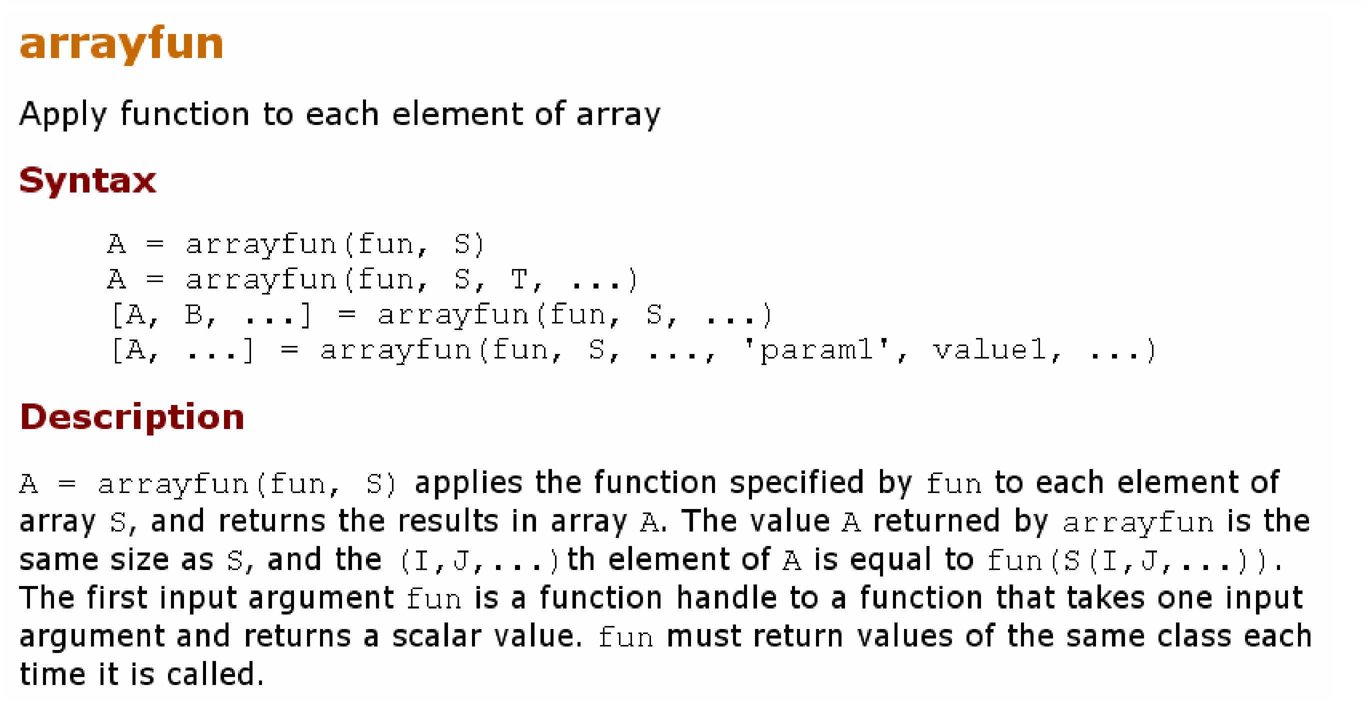
Generate arrays for N-D functions and interpolation

Syntax

```
[X1,X2,X3,...] = ndgrid(x1,x2,x3,...)
[X1,X2,...] = ndgrid(x)
```

Description

`[X1,X2,X3,...] = ndgrid(x1,x2,x3,...)` transforms the domain specified by vectors `x1,x2,x3...` into arrays `X1,X2,X3...` that can be used for the evaluation of functions of multiple variables and multidimensional interpolation. The `i`th dimension of the output array `Xi` are copies of elements of the vector `xi`.



arrayfun

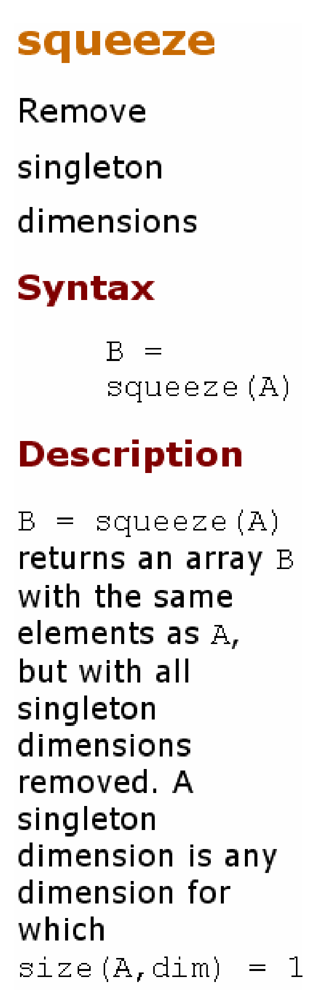
Apply function to each element of array

Syntax

```
A = arrayfun(fun, S)
A = arrayfun(fun, S, T, ...)
[A, B, ...] = arrayfun(fun, S, ...)
[A, ...] = arrayfun(fun, S, ..., 'param1', value1, ...)
```

Description

`A = arrayfun(fun, S)` applies the function specified by `fun` to each element of array `S`, and returns the results in array `A`. The value `A` returned by `arrayfun` is the same size as `S`, and the `(I,J,...)`th element of `A` is equal to `fun(S(I,J,...))`. The first input argument `fun` is a function handle to a function that takes one input argument and returns a scalar value. `fun` must return values of the same class each time it is called.



squeeze

Remove singleton dimensions

Syntax

```
B = squeeze(A)
```

Description

`B = squeeze(A)` returns an array `B` with the same elements as `A`, but with all singleton dimensions removed. A singleton dimension is any dimension for which `size(A,dim) = 1`

We evaluate the model over a 3-dimensional grid of parameters, and then normalize it.

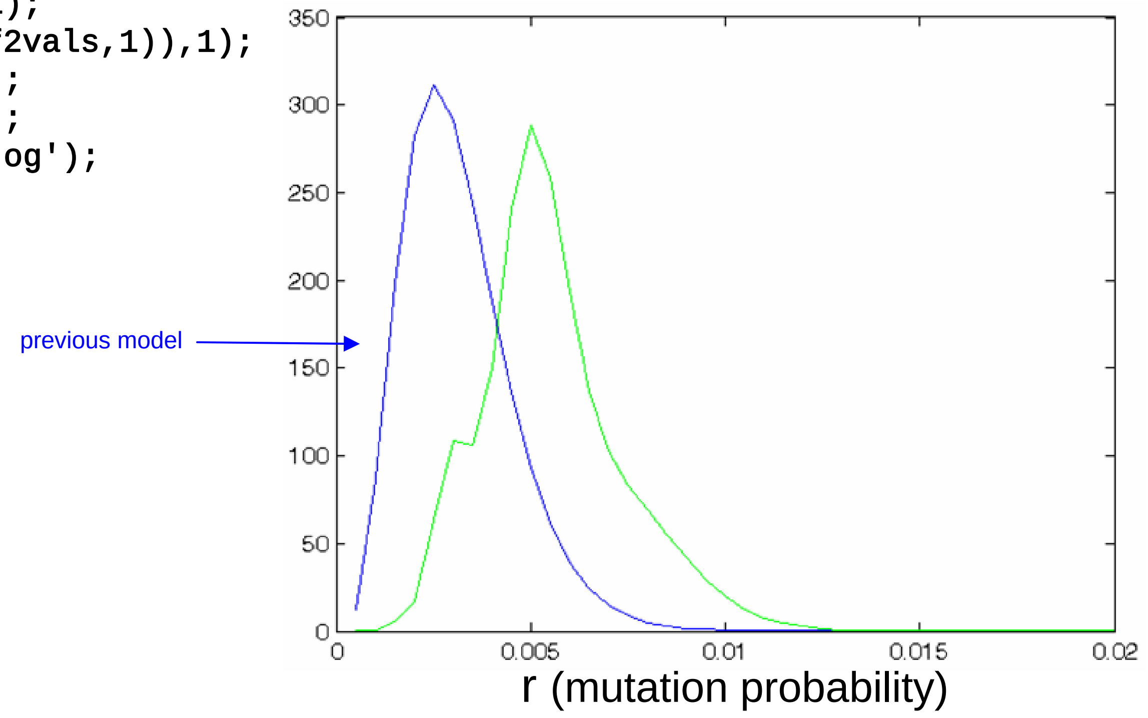
```
rvals = del:del:0.02;
cvals = [.002 .005 .01 .02 .03 .06 .1 .2]
lcavals = [25 50 100 200]
[rgrid cgrid lcagrid] = ndgrid(rvals,cvals,lcavals);
f2vals = arrayfun(model2,rgrid,cgrid,lcagrid);
f2vals = f2vals ./ sum(f2vals(:))
```

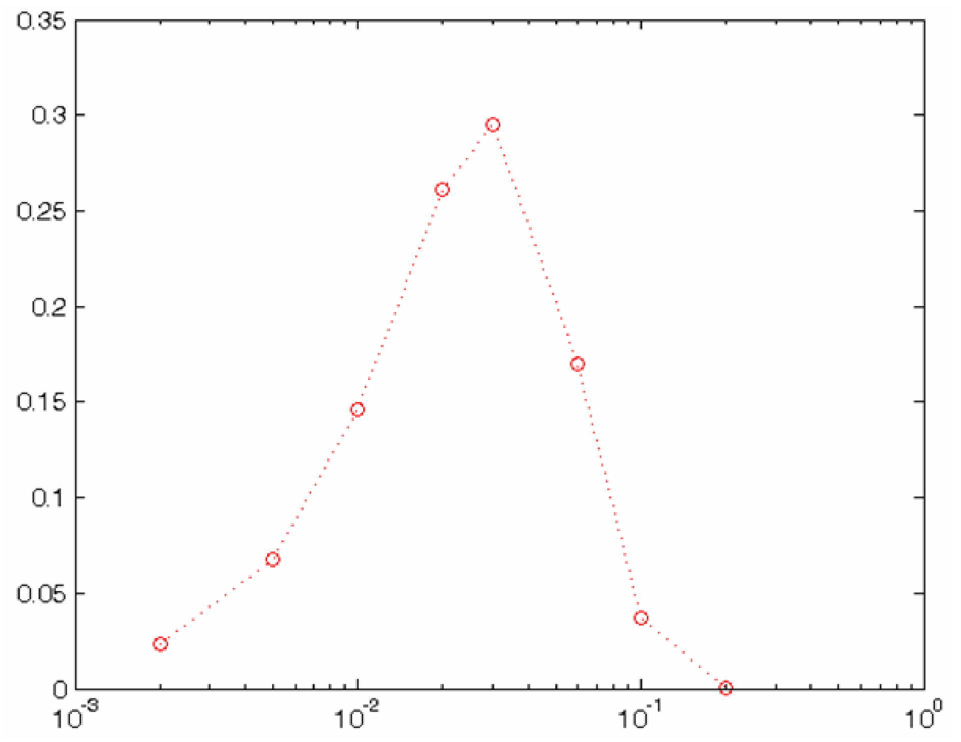
priors are implicit in the spacing of the grids, here approximately logarithmic; each grid cell is taken as equiprobable

We get individual parameter distributions by marginalization

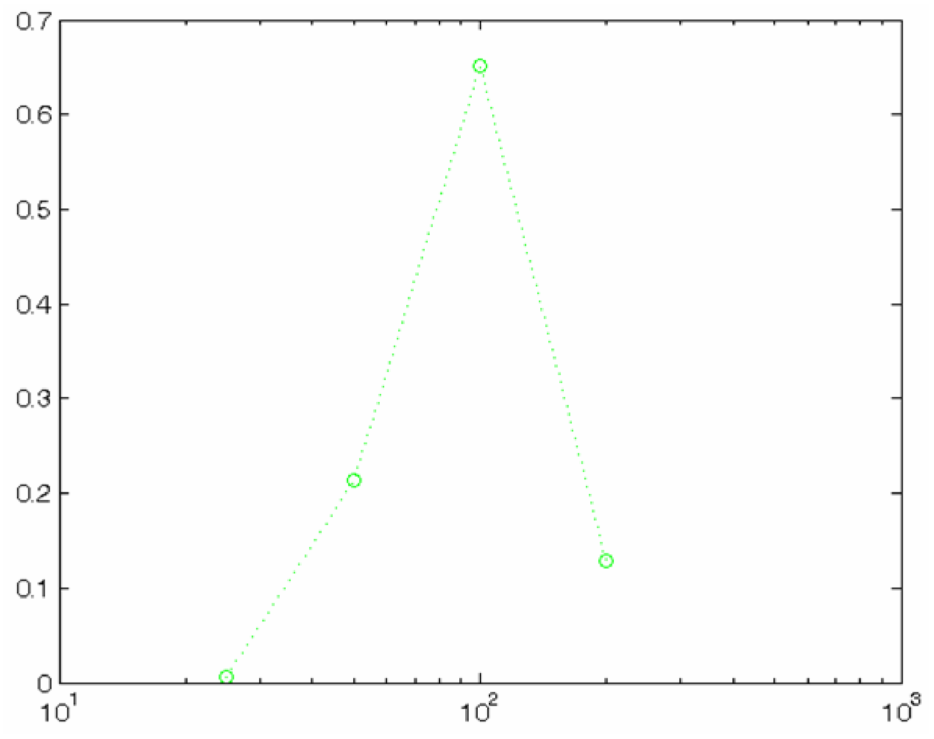
```
f2r = sum(sum(f2vals,3),2);
f2c = sum(sum(f2vals,3),1);
f2lca = sum(squeeze(sum(f2vals,1)),1);
plot(rvals,f2r./del,'-g');
semilogx(cvals,f2c,':or');
semilogx(lcavals,f2lca,':og');
```

Hint: use size() to debug this kind of stuff!





c (non-paternal probability per generation)



L (generations to LCA)

Calculate mixture probabilities by

$$P(v_j = 0|\text{data}) \propto \int \frac{p_{0j}(1-s)}{p_{1j}s + p_{0j}(1-s)} P(s|\text{data}) ds$$

now with additional marginalizations over r,c,L:

father was a sailor!

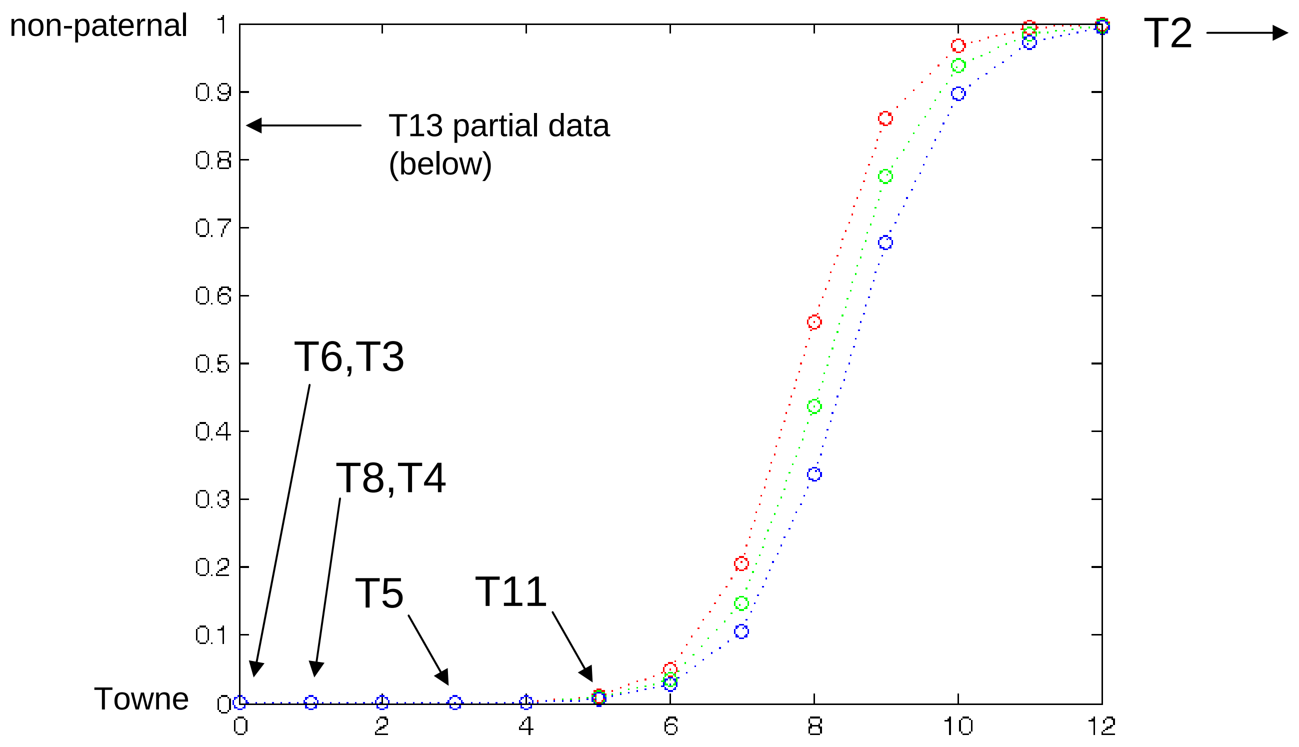
```
function p = nonpatprob(k,n,loci,rgrid,cgrid,lcagrid,f2vals)
    p = squeeze(sum(sum(sum(  arrayfun(@ppat,rgrid,cgrid,lcagrid) .* f2vals  ,3),2),1));

    function p = ppat(r,c,lca)
        p1 = (1-c).^n * poisspdf(k,n*loci*r);
        p2 = (1-(1-c).^n) * poisspdf(k,(n+lca)*loci*r);
        p = p2/(p1+p2)
    end

end
```

```
 for k=0:12, gen9(k+1) = nonpatprob(k,9,37,rgrid,cgrid,lcagrid,f2vals); end
 for k=0:12, gen10(k+1) = nonpatprob(k,10,37,rgrid,cgrid,lcagrid,f2vals); end
 for k=0:12, gen11(k+1) = nonpatprob(k,11,37,rgrid,cgrid,lcagrid,f2vals); end
 plot([0:12],gen9,':or')
 plot([0:12],gen10,':og')
 plot([0:12],gen11,':ob')
```

And the answers are…

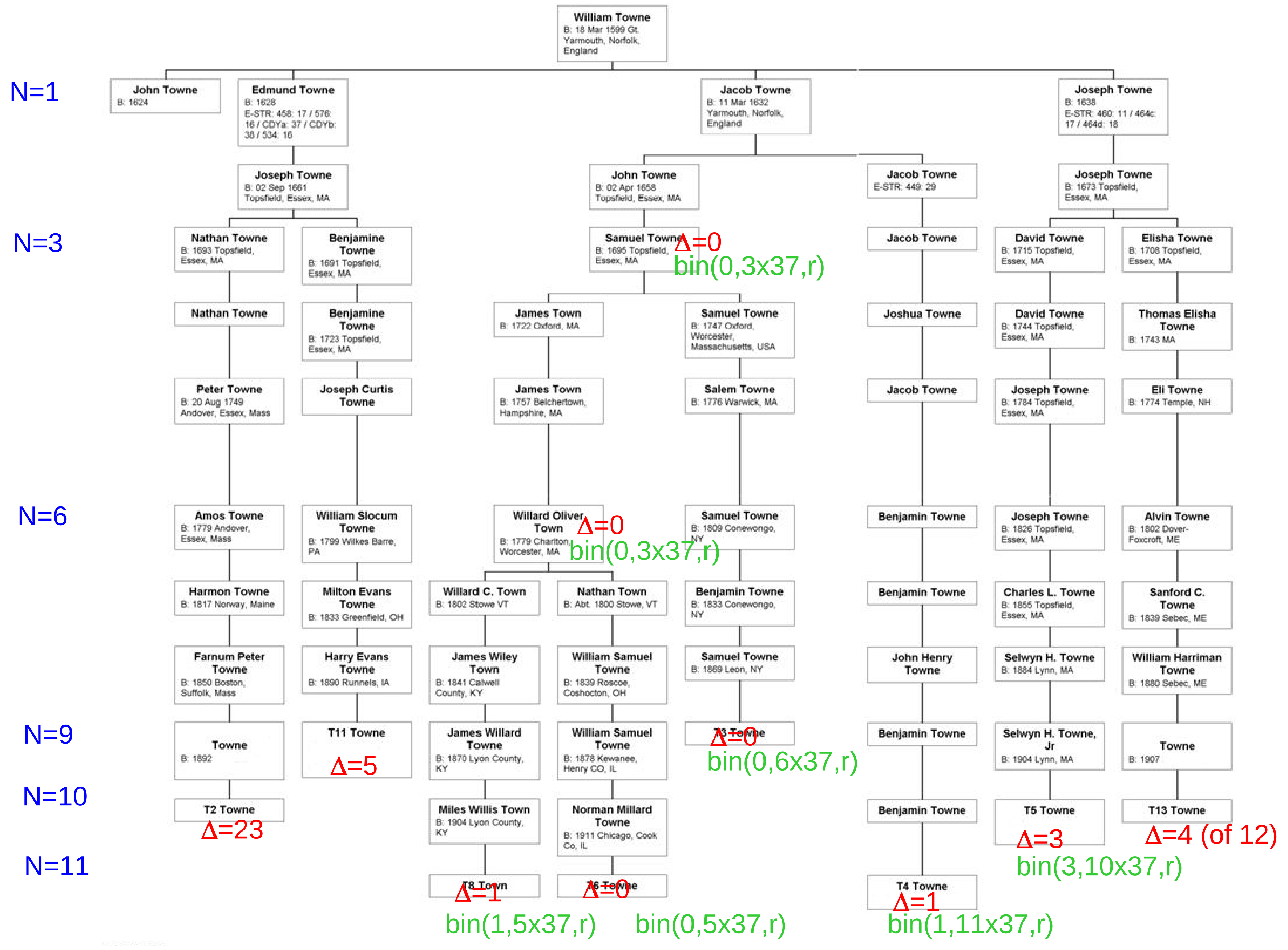


```
p13 = nonpatprob(4,10,12,rgrid,cgrid,lcagrid,f2vals)
p13 =
    0.8593
```

So, by Bayesian statistical modeling, T11 fought his way back to legitimacy. I guess this a happy ending.

Confession: the above picture is not quite right, because I found a bug in the code and didn't redo the picture. Could somebody try to redo this and post it on the course website?

Hi, guys! Remember us?


William Towne B: 18 Mar 1599 Gt. Yarmouth, Norfolk, England
N=1
John Towne B: 1624
Edmund Towne B: 1628 E-STR: 458: 17 / 576: 16 / CDYa: 37 / CDYb: 38 / 534: 16
Jacob Towne B: 11 Mar 1632 Yarmouth, Norfolk, England
Joseph Towne B: 1638 E-STR: 460: 11 / 464c: 17 / 464d: 18
Joseph Towne B: 02 Sep 1661 Topsfield, Essex, MA
John Towne B: 02 Apr 1658 Topsfield, Essex, MA
Jacob Towne E-STR: 449: 29
Joseph Towne B: 1673 Topsfield, Essex, MA
N=3
Nathan Towne B: 1693 Topsfield, Essex, MA
Benjamine Towne B: 1691 Topsfield, Essex, MA
Samuel Towne B: 1695 Topsfield, Essex, MA
Δ=0
bin(0,3x37,r)
Jacob Towne
David Towne B: 1715 Topsfield, Essex, MA
Elisha Towne B: 1708 Topsfield, Essex, MA
Nathan Towne
Benjamine Towne B: 1723 Topsfield, Essex, MA
James Town B: 1722 Oxford, MA
Samuel Towne B: 1747 Oxford, Worcester, Massachusetts, USA
Joshua Towne
David Towne B: 1744 Topsfield, Essex, MA
Thomas Elisha Towne B: 1743 MA
Peter Towne B: 20 Aug 1749 Andover, Essex, Mass
Joseph Curtis Towne
James Town B: 1757 Belchertown, Hampshire, MA
Salem Towne B: 1776 Warwick, MA
Jacob Towne
Joseph Towne B: 1784 Topsfield, Essex, MA
Eli Towne B: 1774 Temple, NH
N=6
Amos Towne B: 1779 Andover, Essex, Mass
William Slocum Towne B: 1799 Wilkes Barre, PA
Willard Oliver Town B: 1779 Charlton, Worcester, MA
Δ=0
bin(0,3x37,r)
Samuel Towne B: 1809 Conewongo, NY
Benjamin Towne
Joseph Towne B: 1826 Topsfield, Essex, MA
Alvin Towne B: 1802 Dover-Foxcroft, ME
Harmon Towne B: 1817 Norway, Maine
Milton Evans Towne B: 1833 Greenfield, OH
Willard C. Town B: 1802 Stowe VT
Nathan Towne B: Abt. 1800 Stowe, VT
Benjamin Towne B: 1833 Conewongo, NY
Benjamin Towne
Charles L. Towne B: 1855 Topsfield, Essex, MA
Sanford C. Towne B: 1839 Sebec, ME
Farnum Peter Towne B: 1850 Boston, Suffolk, Mass
Harry Evans Towne B: 1890 Runnels, IA
James Wiley Town B: 1841 Calwell County, KY
William Samuel Towne B: 1839 Roscoe, Coshocton, OH
Samuel Towne B: 1869 Leon, NY
John Henry Towne
Selwyn H. Towne B: 1884 Lynn, MA
William Harriman Towne B: 1880 Sebec, ME
N=9
Towne B: 1892
T11 Towne
Δ=5
James Willard Towne B: 1870 Lyon County, KY
William Samuel Towne B: 1878 Kewanee, Henry CO, IL
T3 Towne
Δ=0
bin(0,6x37,r)
Benjamin Towne
Selwyn H. Towne, Jr B: 1904 Lynn, MA
Towne B: 1907
N=10
T2 Towne
Δ=23
Miles Willis Town B: 1904 Lyon County, KY
Norman Millard Towne B: 1911 Chicago, Cook Co, IL
Benjamin Towne
T5 Towne
Δ=3
bin(3,10x37,r)
T13 Towne
Δ=4 (of 12)
N=11
T8 Town
Δ=1
bin(1,5x37,r)
T6 Towne
Δ=0
bin(0,5x37,r)
T4 Towne
Δ=1
bin(1,11x37,r)

Hierarchical Bayesian models (just a mention here):

Actually, I'd guess that our LCA model is too crude: no single L is consistent with both T2 and T11, so our model "promoted" T11 to legitimacy. I bet that T11 is a non-paternal event with a distant cousin!
What is really needed is a distribution of L's.

Old model: L is a fixed parameter to be estimated.

$$p_{\text{mix}}(k,n,M|r,c,L) \equiv (1-c)^n p_{\text{Bin}}(k,nM,r) \\ + [1-(1-c)^n] p_{\text{Bin}}(k,(n+L)M,r)$$

$$p(r,c,L|\text{data}) \propto \prod_{\text{Townes}} p(k_i,n_i,M_i|r,c,L)P(r,c,L)$$

Hierarchical model: L is drawn from a distribution, separately for each Towne

$$L \sim \text{Gamma}(\alpha,\beta)$$

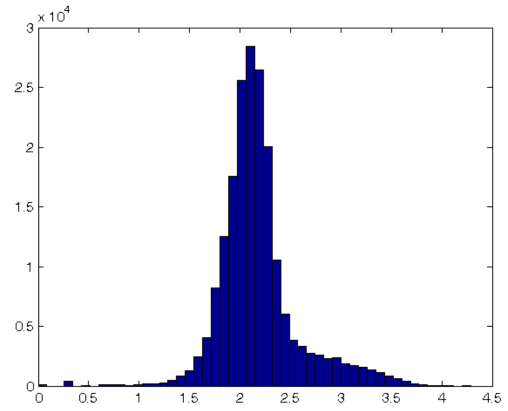
$$p(r,c,\alpha,\beta|\text{data}) \propto \prod_{\text{Townes}} \left[\int p(k_i,n_i,M_i|r,c,L_i)p_{\text{Gamma}}(L_i|\alpha,\beta)dL_i\right] \\ \times P(r,c,\alpha,\beta)$$

What makes this "hierarchical" is that *L_i*, a parameter in one piece of the model is an RV (dependent on "hyper-parameters") in another piece.

Gaussian Mixture Models (GMMs)

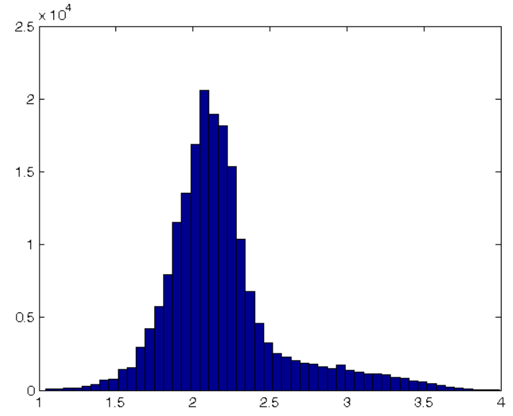
- Yet another method for fitting multiple Gaussians to (possibly) a set of multi-dimensional data points
  - properties of Gaussians are used in detail: doesn't easily generalize to other fitting functions
- But uses the raw data points
  - no binning!
  - hence useful in multidimensions
- Exemplifies Expectation Maximization (EM) methods
  - an important class of methods (Dempster, Laird, & Rubin)
  - we'll show some theory later
- Let's first try it in 1-D on the data set of exon lengths

```
g = readgenestats('genestats.dat');
exons = cell2mat(g.exonlen);
hist(log10(exons),50)
```



```
data = log10(exons(log10(exons)>1
& log10(exons)<4));
hist(data,50)
```

(Let's trim the outliers.)



Key to the notational thicket:

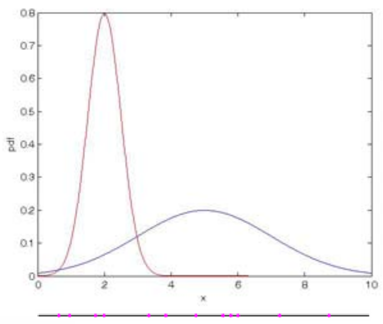
M dimensions

$k = 1 \dots K$ Gaussians “components”

$n = 1 \dots N$ data points

$P(k)$ population fraction in k

$P(\mathbf{x}_n)$ model probability at $\mathbf{x}_n$



$\boldsymbol{\mu}_k$	(the K means, each a vector of length M)
$\boldsymbol{\Sigma}_k$	(the K covariance matrices, each of size $M \times M$)
$P(k\|n) \equiv p_{nk}$	(the K probabilities for each of N data points)

“probabilistic assignment” of a data point to a component!

$$\mathcal{L} = \prod_n P(\mathbf{x}_n)$$ overall likelihood of the model

$$P(\mathbf{x}_n) = \sum_k N(\mathbf{x}_n \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) P(k)$$ specify the model as a mixture of Gaussians

$$N(\mathbf{x} \mid \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{(2\pi)^{M/2} \det(\boldsymbol{\Sigma})^{1/2}} \exp[-\tfrac{1}{2}(\mathbf{x} - \boldsymbol{\mu}) \cdot \boldsymbol{\Sigma}^{-1} \cdot (\mathbf{x} - \boldsymbol{\mu})]$$

Goal is to find *all* of the above, starting with only the $\mathbf{x}_n$

(So far this could be frequentist or Bayesian, although it was invented by frequentists.)