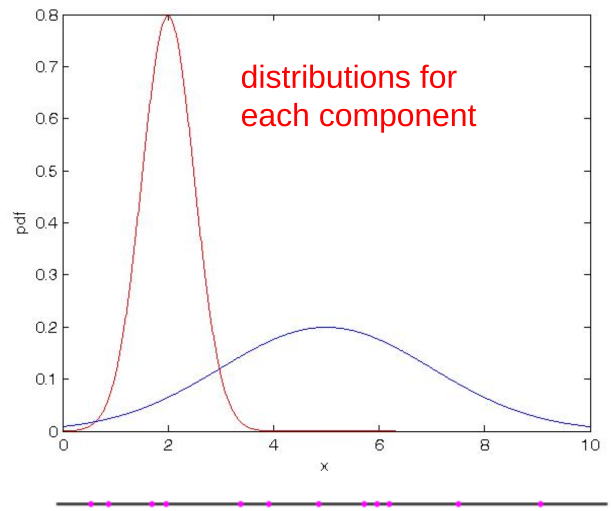
Opinionated Lessons in Statistics

by Bill Press

#27 Mixture Models

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

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

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!

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

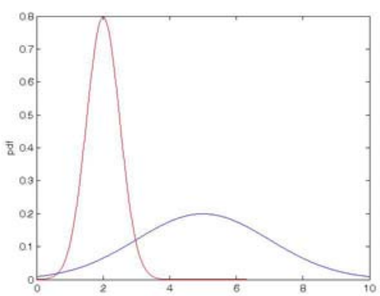
That is the complete model, usually too much to comprehend directly. Instead, we are usually interested in various marginalizations. For example:

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

key step is here: $= \prod_i [p_{1i}s + p_{0i}(1-s)] P(s)$ (multiply it out!)

prob of i in the mixture distribution

prior on the mixture



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

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

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

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

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

$$= \prod_i [p_{1i}s + p_{0i}(1-s)] P(s)$$

and similarly

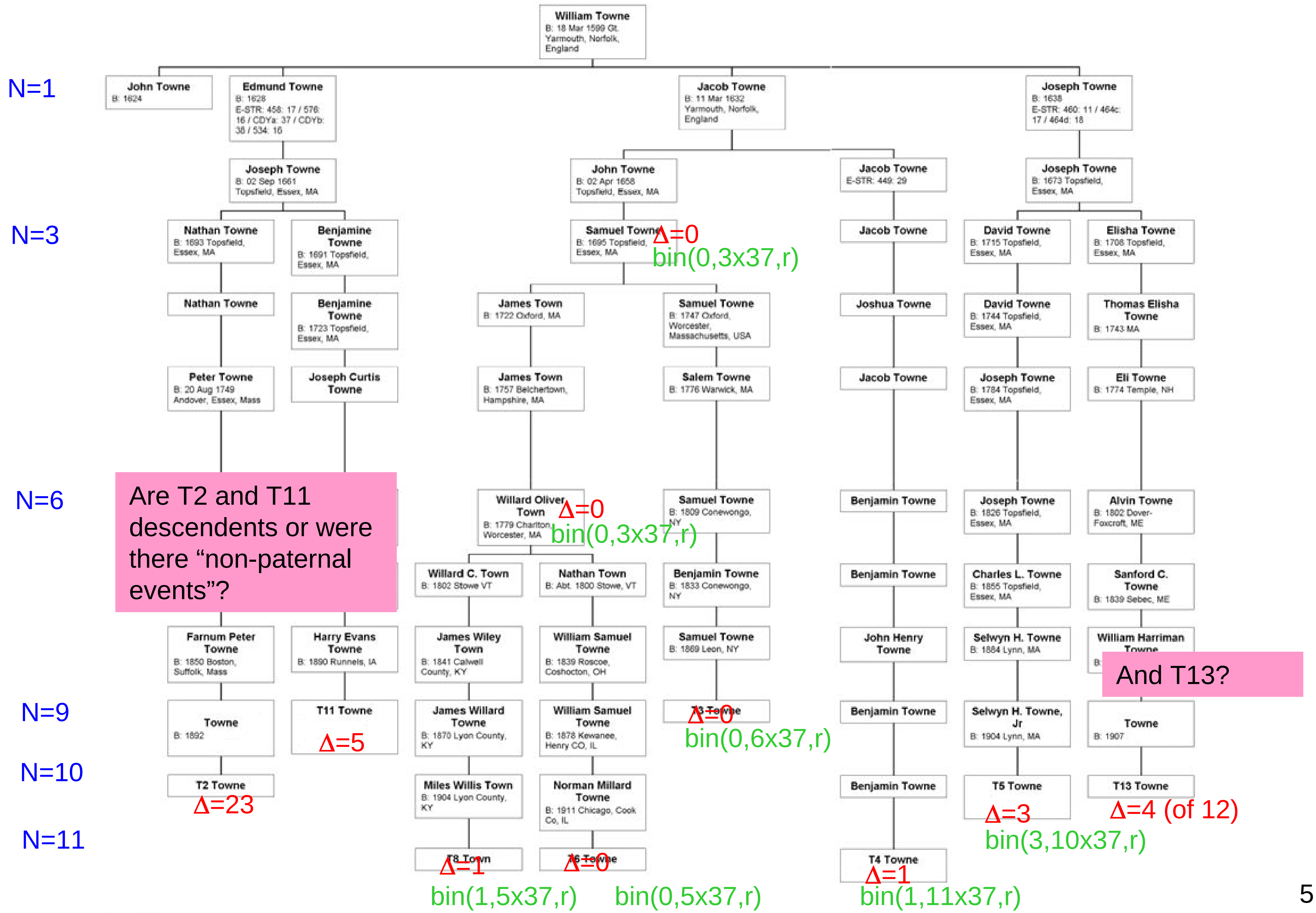
$$P(v_j = 0|\text{data}) \propto \int \frac{p_{0j}(1-s)}{p_{1j}s + p_{0j}(1-s)} P(s|\text{data}) ds$$

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?



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.

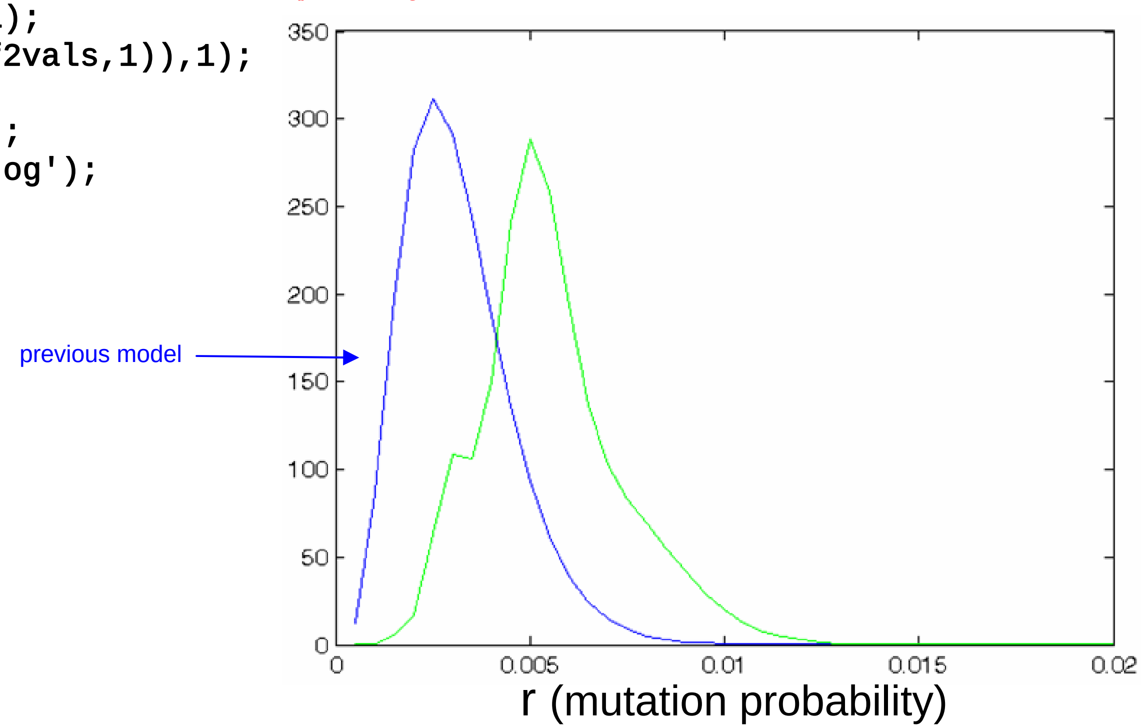
```
rvals = 0.0005:0.0005:0.02;
cvals = [.002 .005 .01 .02 .03 .06 .1 .2]
lcavals = [25 50 100 200]
[rgrid cgrid lcagrid] = ndgrid(rvals,cvals,lcavals);
f2vals = arrayfun(model2,rgrid,cgrid,lcagrid);
f2vals = f2vals ./ sum(f2vals(:))
```

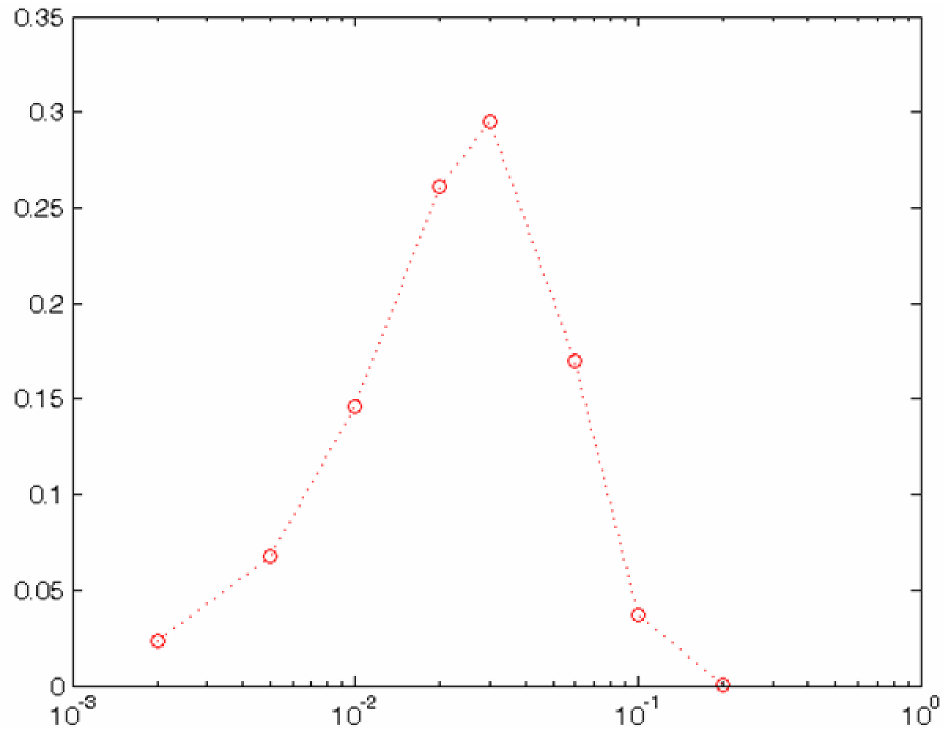
priors are implicit in the spacing of the grids, here approximately logarithmic; each grid cell is taken as equiprobable

We get individual parameter distributions by marginalization

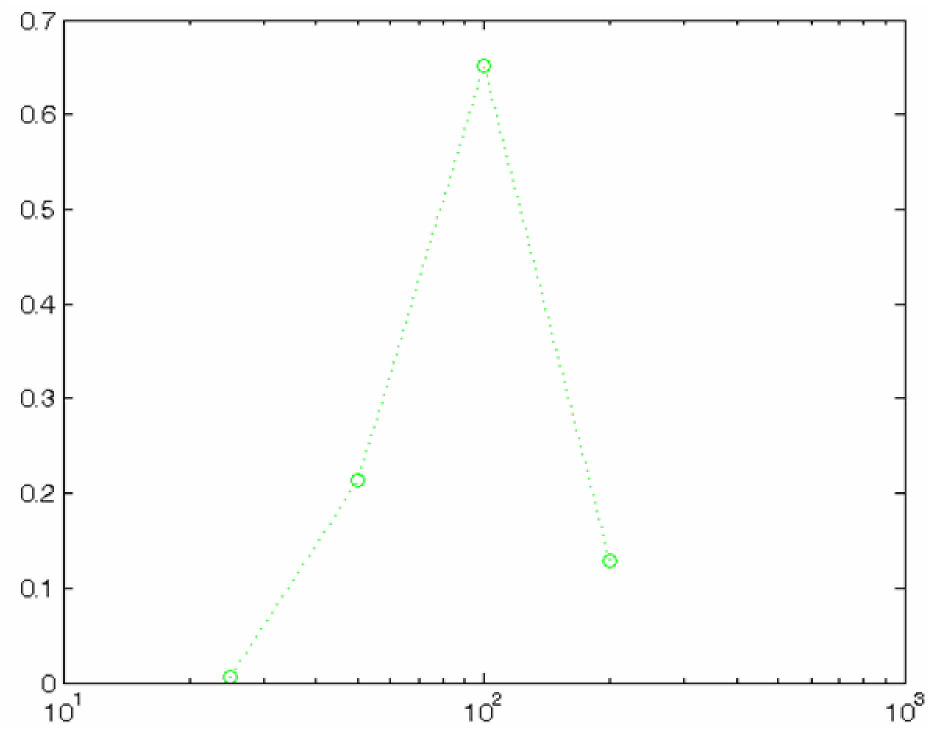
```
f2r = sum(sum(f2vals,3),2);
f2c = sum(sum(f2vals,3),1);
f2lca = sum(squeeze(sum(f2vals,1)),1);
plot(rvals,f2r,'-g');
semilogx(cvals,f2c,':or');
semilogx(lcavals,f2lca,':og');
```

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





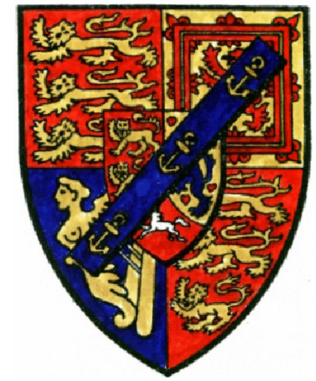
c (non-paternal probability per generation)



L (generations to LCA)

Calculate mixture probabilities by

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



father was a sailor!

now with additional marginalizations over r,c,L:

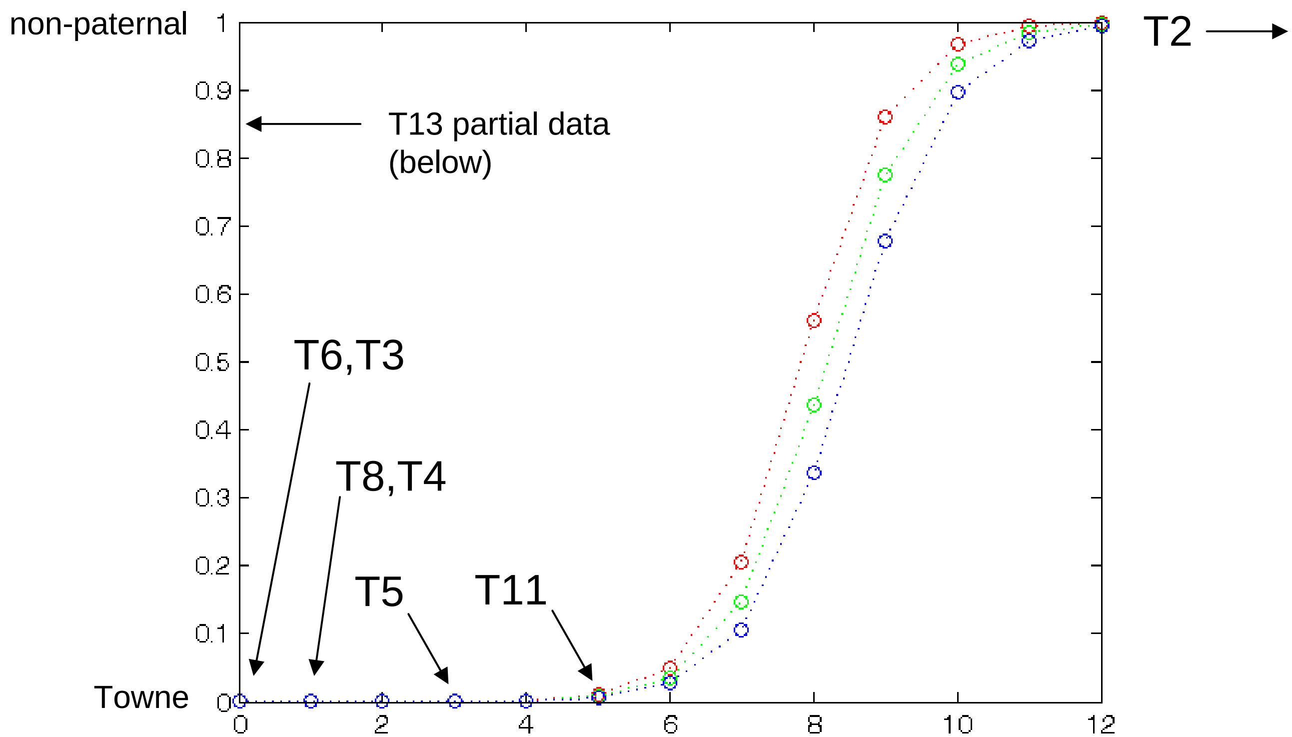
```
function p = nonpatprob(k,n,loci,rgrid,cgrid,lcagrid,f2vals)
    p = squeeze(sum(sum(sum(  arrayfun(@ppat,rgrid,cgrid,lcagrid) .* f2vals  ,3),2),1));

    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 close, but not quite right, because I found a bug in the code and didn't redo the picture. Challenge: redo the calculation and see how different your answer is!

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.

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

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

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

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

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

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.