

Computational Statistics with Application to Bioinformatics

Prof. William H. Press
Spring Term, 2008
The University of Texas at Austin

Unit 6: Understanding Distributions Known
Only Empirically

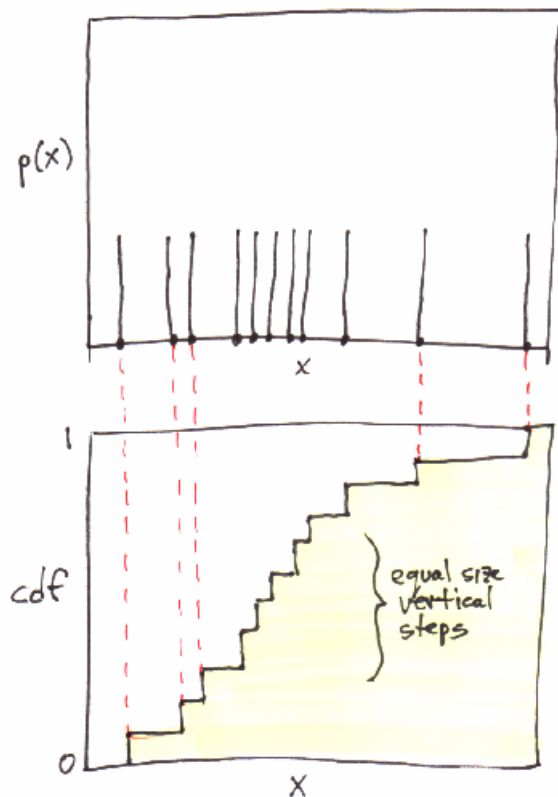
Unit 6: Understanding Distributions Known Only Empirically (Summary)

- Empirical distributions based on sample data
 - PDF is delta functions, CDF is staircase
 - sometimes this is all we know!
- Kolmogorov-Smirnov tests for difference between distributions
 - p-value test rejects hypothesis that they are the same
- The IQagent data structure
 - efficient for keeping an accurate approximation to empirical distributions
 - can be incrementally updated as new data comes along
- How to read the genestats.dat data file
 - containing data on intron and exon lengths by individual gene
 - in both C++ and Matlab
- How to digest the distributions with IQagent
 - .add method
- How to plot PDFs so that the statistical error is uniform
 - equal bins in ΔP , not in x
 - can easily get from IQagent method .report
- When to plot $x p(x)$ instead of $p(x)$
 - when abscissa is a log scale
 - point is to make (visual) area under the curve reflect the actual sample distribution

continues

- Some biological features in the distributions
 - intron lengths pile up at the lower limit of 150 (splice mechanics)
 - first and last exons have a different distribution from middle ones
 - single exon olfactory genes are a big feature
- C++ wrapper to use IQagent from within Matlab
- Some of the length distributions are clearly different between AT-rich genes and CG-rich genes
 - intron and gene lengths in particular
- Kolmogorov-Smirnov test whether intron and exon length distributions for AT rich and CG rich genes are identical
 - They're not – highly statistically significantly
 - But in one case they are nevertheless virtually identical
 - can happen when lots of data
 - editorial on when this is and isn't good science
- Simulate by re-sampling to see how the statistical significance would have been different with various amounts of data
 - basically, no significance up to a certain data size
 - thereafter exponential increase in significance

Distributions are often known only empirically as lists of observed events (values of x , e.g.)



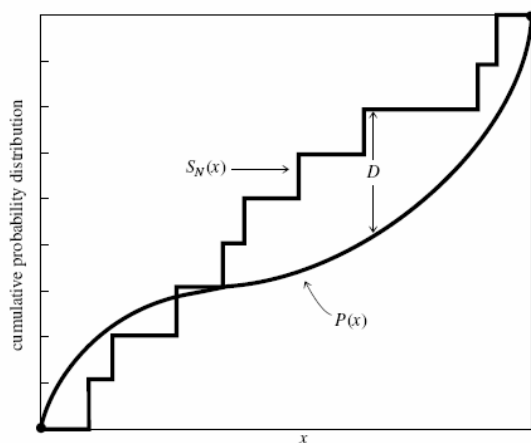
The pdf is a bunch of Dirac delta functions. Although this doesn't "look like" the smooth distribution from which it's drawn, it can sometimes be used "as is", for example in bootstrapping (to be discussed later). Basically it is the only available non-parametric estimate of the distribution.

The cdf is a staircase with a vertical step of $1/N$ at each observed data value. This has enough (stepwise) continuity that we can do useful things with it, for example Kolmogorov-Smirnov (KS) tests or smoothing and interpolation ("slightly parametric").

To generate random numbers from an empirically known distribution, we can use the transformation method. But we need a data structure for quick inverting of the (staircase) CDF.



Kolmogorov-Smirnov (KS) tests (quick summary):



$$D = \max_{-\infty < x < \infty} |S_N(x) - P(x)|$$

$$D = \max_{-\infty < x < \infty} |S_{N_1}(x) - S_{N_2}(x)|$$

$$N_e = \frac{N_1 N_2}{N_1 + N_2}$$

} original
KS

$$\text{Probability } (D > \text{observed}) = Q_{KS} \left(\left[\sqrt{N_e} + 0.12 + 0.11/\sqrt{N_e} \right] D \right)$$

Variants (to increase power of test near the edges):

Anderson-Darling (but has no easy tail prob.)

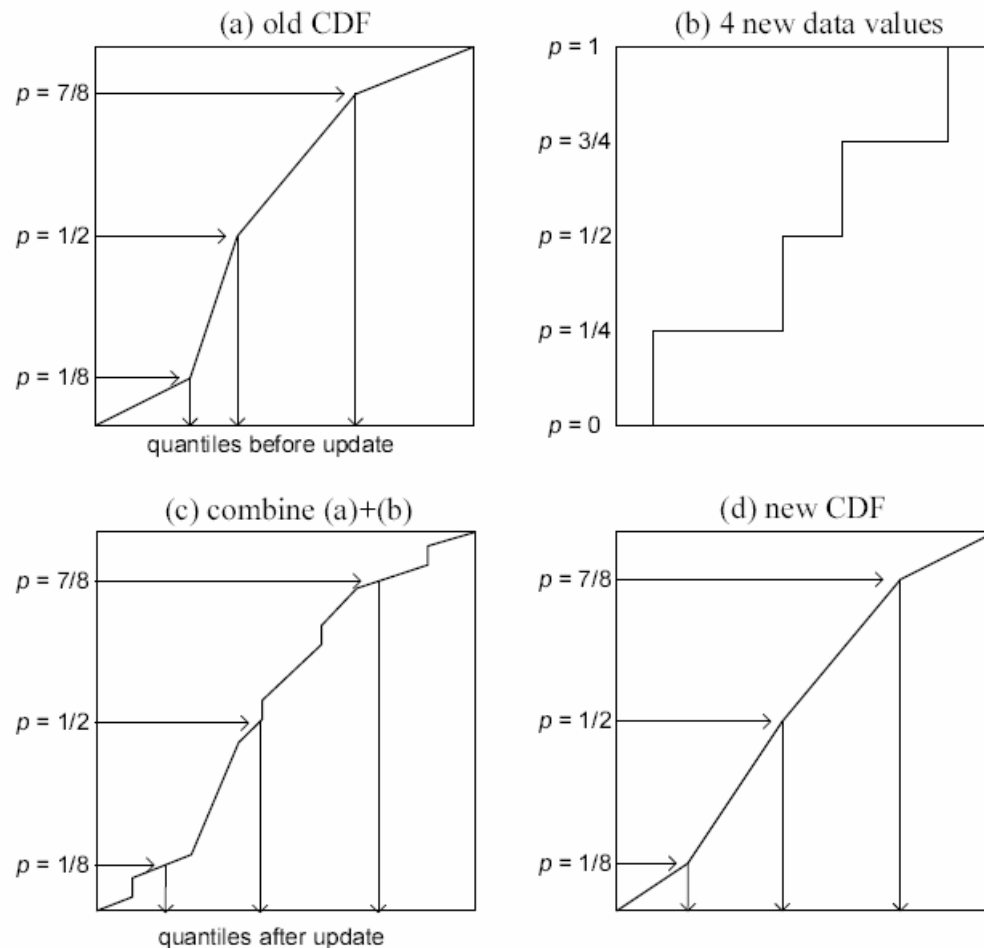
$$D^* = \max_{-\infty < x < \infty} \frac{|S_N(x) - P(x)|}{\sqrt{P(x)[1 - P(x)]}}$$

Kuiper: circular symmetry and easy tail prob.

$$V = D_+ + D_- = \max_{-\infty < x < \infty} [S_N(x) - P(x)] + \max_{-\infty < x < \infty} [P(x) - S_N(x)]$$



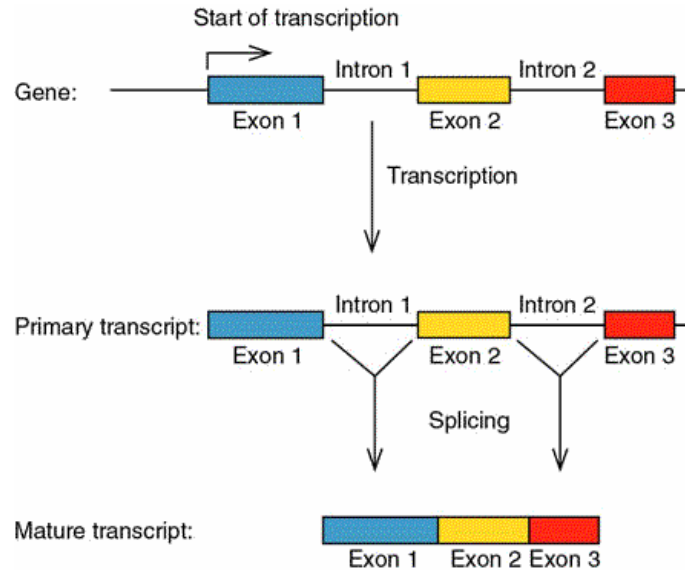
“IQagent” (Chambers et al.) is a great data structure for empirical pdf’s



- Maintain piecewise linear estimate of cdf at (e.g.) 250 points
 - include “popular” quantiles from 10^{-6} to $1-10^{-6}$
 - also include every 1% step
- Update from data in batches of (e.g.) 1000
 - but can also update whenever output is needed
- Easy to pick parameters that make it “accurate enough”
 - meaning: interpolation and statistical errors much smaller than sampling error intrinsic in the finite data available
 - $10^{-2} \Delta p \Rightarrow \sim 10^{-4}$ interp error $\Rightarrow \sim 100 \times 10^8$ sample needed to see the effect
 - with sample size N , quantile p is knowable only up to

$$\sim \sqrt{Np(1-p)}/N$$

Let's use IQagent to play around with a fundamental biological data set (“exploratory statistics”)



file **genestats.dat** (on course web site) contains 20694 lines like this:

					N exon lengths				N-1 intron lengths					
		number of exons N												
ENST00000341866	17470	3262	0.00002	4	1290	349	1412	211	169	678	13361	<EOL>		
ENST00000314348	22078	1834	0.00001	7	100	166	113	178	165	262	850	5475		
385 3273 1149 2070 7892														
ENST00000313081	13858	1160	0.00001	6	496	150	107	85	151	171	2068	76	2063	
674 7817														
ENST00000298622	80000	6487	0.00001	24	135	498	216	120	147	132	36	60	129	
129 84 63 99 99 54 66 69 78 204 66 73 1081 397 2452 12133 15737 1513 769 942														
103 829 2272 1340 3058 327 2371 1361 471 2922 735 85 9218 1257 2247 897 822														
12104														
gene name	total length	total length of exons	ignore for now											

Read the data file and digest it with IQagent structs:

```
FILE *IN = fopen("D:\\MyCompStatCourse\\genestats.dat", "rb");
IQagent intronlen, exonlen, messlen, totlen;
while (fscanf(IN, "%s", gname) == 1) {
    fscanf(IN, "%d%dLf%d", &len, &elen, &pisto, &ne);
    if (1) { // put further selections here
        for (j=0; j<ne; j++) {
            fscanf(IN, "%d", &k);
            exonlen.add(Doub(k));
        }
        for (j=0; j<ne-1; j++) {
            fscanf(IN, "%d", &k);
            intronlen.add(Doub(k));
        }
        totlen.add(Doub(len));
        messlen.add(Doub(elen));
    } else {
        for (j=0; j<ne+ne-1; j++) fscanf(IN, "%d", &k);
    }
}
fclose(IN);
xlo=10.; xhi=1.e6;
plotadist(plot, intronlen, xlo, xhi, 255, 0, 0);
plotadist(plot, exonlen, xlo, xhi, 0, 200, 0);
plotadist(plot, messlen, xlo, xhi, 0, 0, 128);
plotadist(plot, totlen, xlo, xhi, 0, 128, 128);
```

(in readgenestats_snip.cpp on course web site)

Guide:

There are multiple pedagogical goals in the next 10 or so slides, so don't get too confused:

1. How to read the genestats.dat file in both C++ and MATLAB
2. How to use IQagent in both C++ and MATLAB
3. Gotcha's about plotting distribution functions on log scales
4. **Learn some genomics**
5. In case you don't have other plotting software, learn to use NR3's postscriptplot3.h (on course web site)

Once digested, call IQagent for any quantile values, e.g., to plot the distributions

```
void plotadist(PSplot &plot, IQagent &iq, Doub xlo,
  Doub xhi, Int r, Int g, Int b) {
  Int N=500,j;
  Doub ymax,p,smooth=5.;
  VecDoub xx(N),yy(N);
  ymax = 0.;
  for (j=0;j<N;j++) {
    p = (j+0.5)/(N+1.);
    xx[j] = iq.report(p);
    yy[j] = iq.report(p) /
      (iq.report(p+smooth/(N+1.))-iq.report(p-smooth/(N+1.)));
    if (yy[j]>ymax) ymax = yy[j];
  }
  plot.setcolor(r,g,b);
  plot.setlimits(xlo,xhi,0.,1.2*ymax);
  plot.lineplot(xx,yy);
}
```

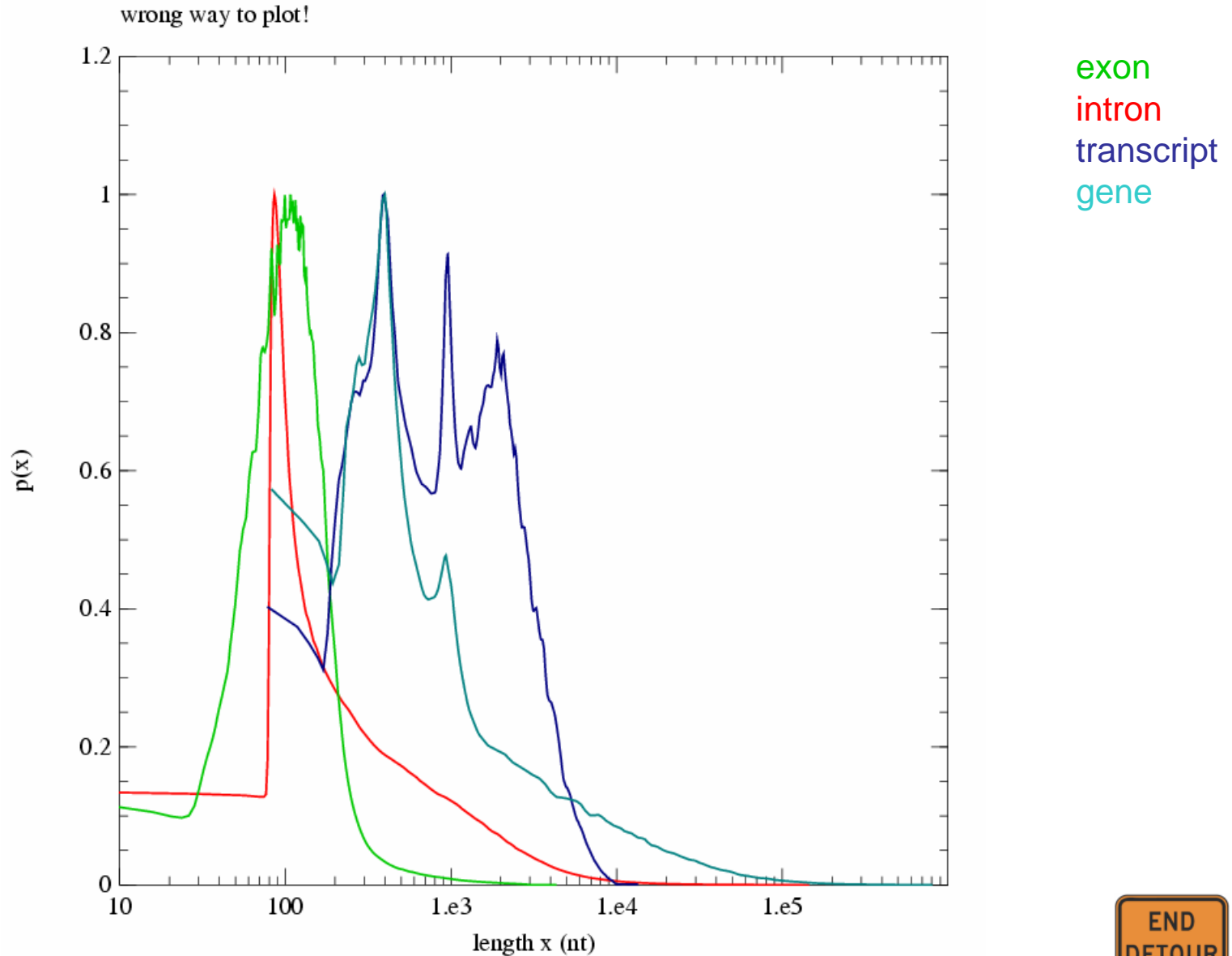
Why not 1? (Next slide.)

(in readgenestats_snip.cpp on course web site)

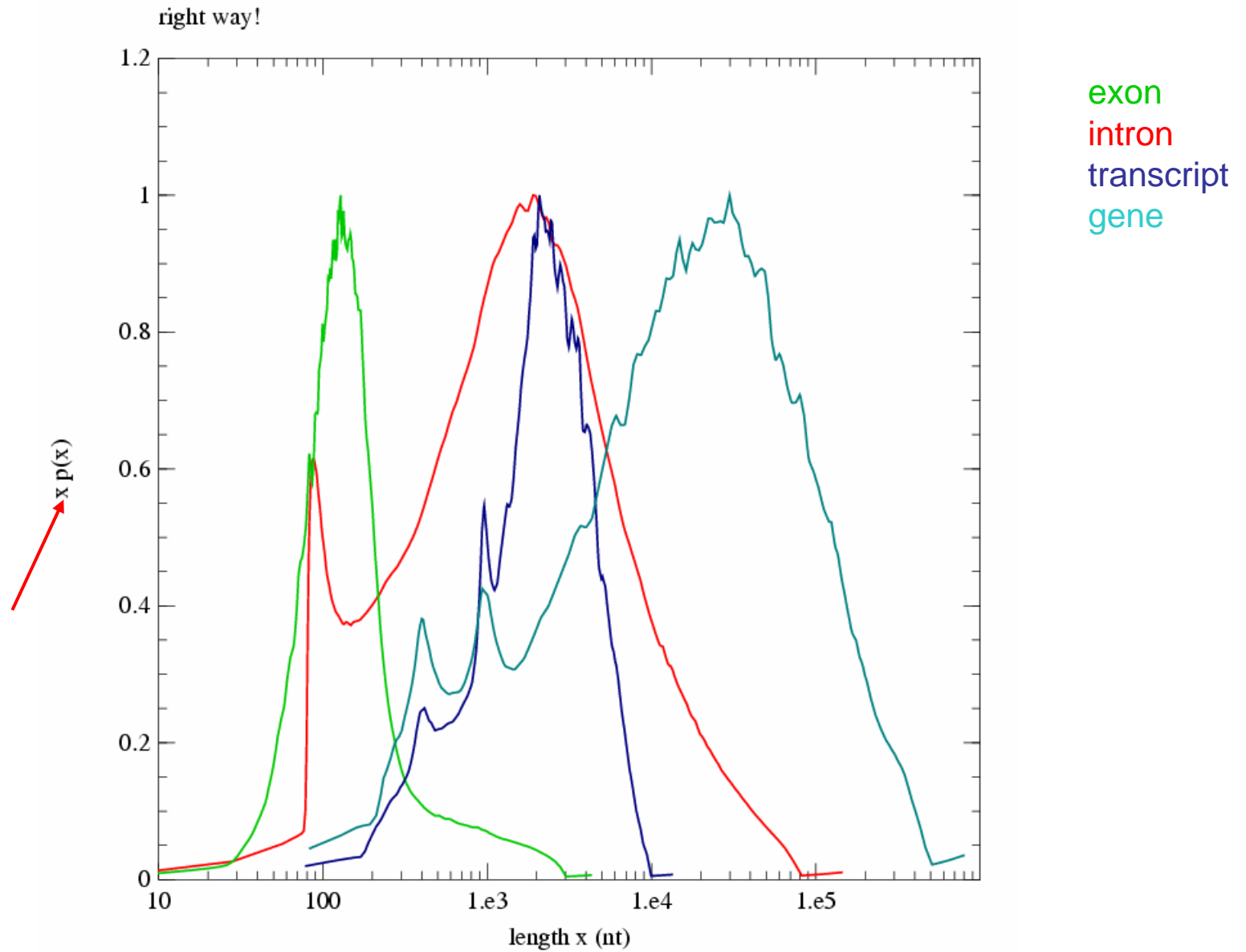
Since there are equal counts in each yy bin, the uncertainty of the plotted distribution is constant. This makes interpretation of significant features easier. It's not true for conventionally plotted histograms.



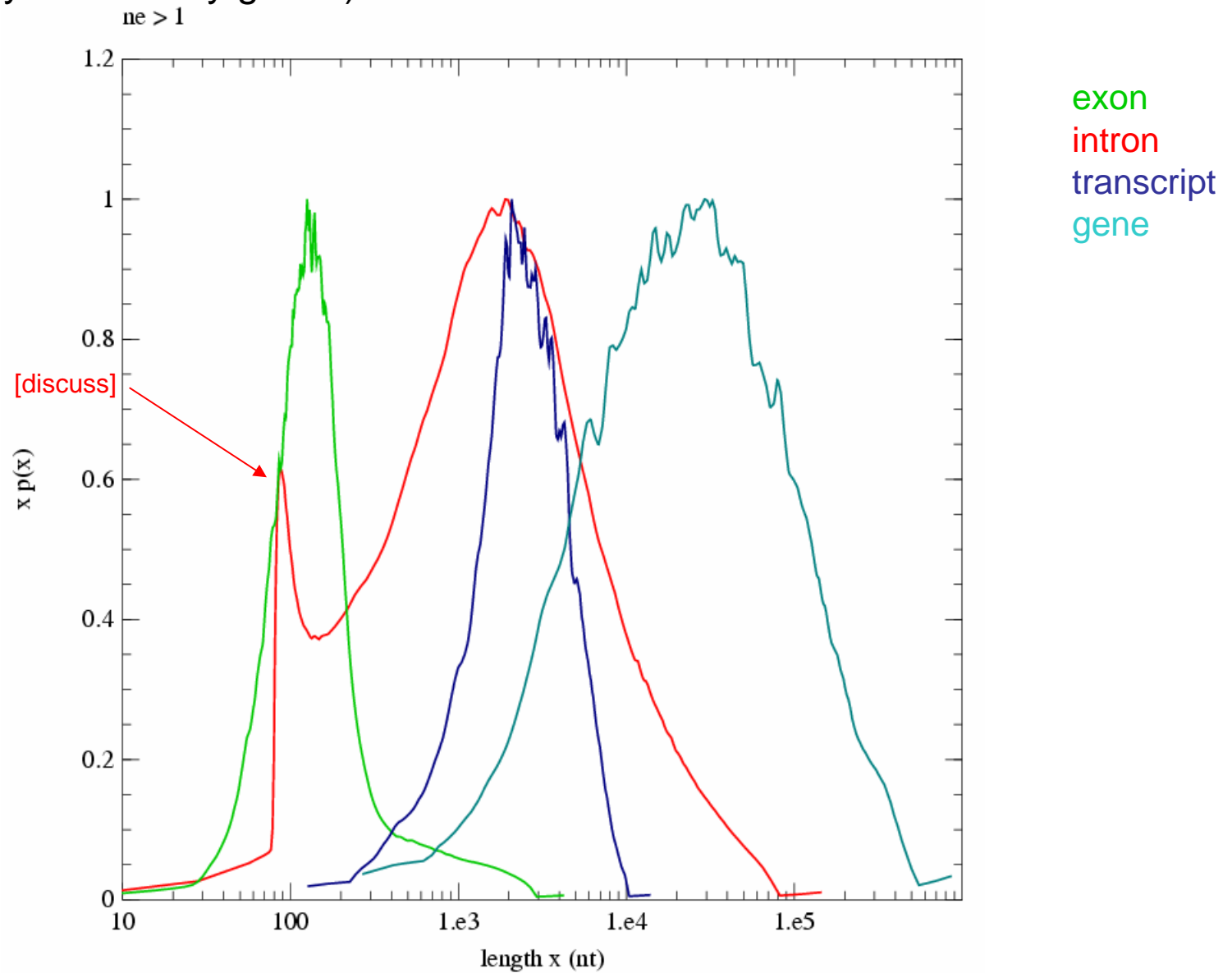
Log scales gotcha: area under the curve doesn't equal probability, and visual impression is not meaningful



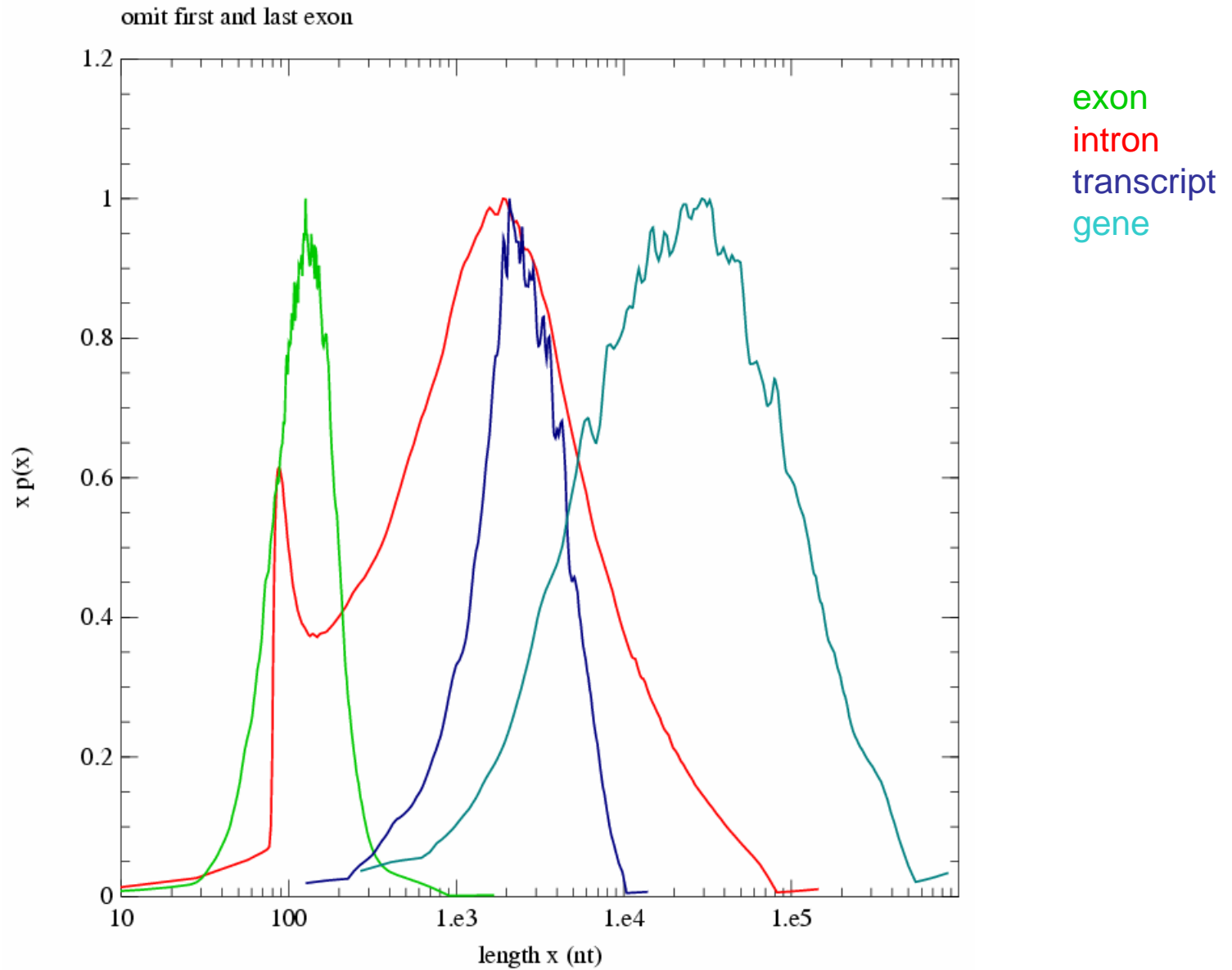
Plotting $x p(x)$ against $\log x$ shows what the distribution really is like



Eliminate genes with a single exon (the peaks that disappear are, e.g., a large family of olfactory genes)



Omitting first and last exons yields distributions that are roughly log-normal.
Deviations from log-normal are biologically interesting.



Easy, now, to generate (e.g.) random intron lengths:

```
Ran ran(112874);  
for (j=0;j<20;j++) printf("%6.0f\n",intronlen.report(ran.doub()));
```

```
687  
13043  
446  
4419  
3196  
1562  
1356  
16826  
2848  
85  
1507  
1546  
59203  
3415  
178  
35130  
5959  
642  
5967  
853
```

```

function genestats = readgenestats(filename)
maxlines = 25000; % cheating, sort of

fid = fopen(filename,'rt'); % text mode
if fid == -1
    error('Unable to open file ''%s''.', filename);
end

gname = repmat(' ',maxlines,15);
len = zeros(maxlines,1);
elen = zeros(maxlines,1);
piso = zeros(maxlines,1);
ne = zeros(maxlines,1);
exonlen = cell(maxlines,1);
intronlen = cell(maxlines,1);
for nlines = 1:maxlines
    C = textscan(fid,'%s %n %n %n %n',1);
    if feof(fid), break; end
    gname(nlines,:) = C{1}{1};
    len(nlines) = C{2};
    elen(nlines) = C{3};
    piso(nlines) = C{4};
    ne(nlines) = C{5};
    exonlen(nlines) = textscan(fid,'%n',ne(nlines));
    if ne(nlines) > 1
        intronlen(nlines) = textscan(fid,'%n',ne(nlines)-1);
    end
end
fclose(fid);
nlines = nlines - 1;
gname = gname(1:nlines,:);
len = len(1:nlines);
elen = elen(1:nlines);
piso = piso(1:nlines);
ne = ne(1:nlines);
exonlen = exonlen(1:nlines);
intronlen = intronlen(1:nlines);
genestats = dataset(gname,len,elen,piso,ne,exonlen,intronlen);

```

(readgenestats.m on course web site)

Matlab function to read the data set is somewhat messy because of the variable number of exons/introns on each data line. Returned dataset is a cell array, with exonlen and intronlen containing arrays of variable length. Don't worry too much about this, now. We'll see by example how to use it. (Thanks to Peter Perkins.)

```

g = readgenestats('genestats.dat');
summary(g)
gname: [20694x15 char]
len: [20694x1 double]
      min      1st Q      median      3rd Q      max
      60      3976      16221      49058      2298740
elen: [20694x1 double]
      min      1st Q      median      3rd Q      max
      45      1107      2082      3344      72066
pi so: [20694x1 double]
      min      1st Q      median      3rd Q      max
      0      1.0000e-005      1.3000e-004      0.2843      1.0000
ne: [20694x1 double]
      min      1st Q      median      3rd Q      max
      1      3      7      13      184
exonlen: [20694x1 cell]
intronlen: [20694x1 cell]

```

Wrapper for accessing NR3's IQagent from Matlab (NRiqagent.cpp on course web site):

```
#include "mex.h"
// NR3 includes:
#include "nr3.h"
#include "sort.h"
#include "iqagent.h"
```

see http://nr.com/nr3_matlab.html for tutorial

```
IQagent *agent = NULL;
```

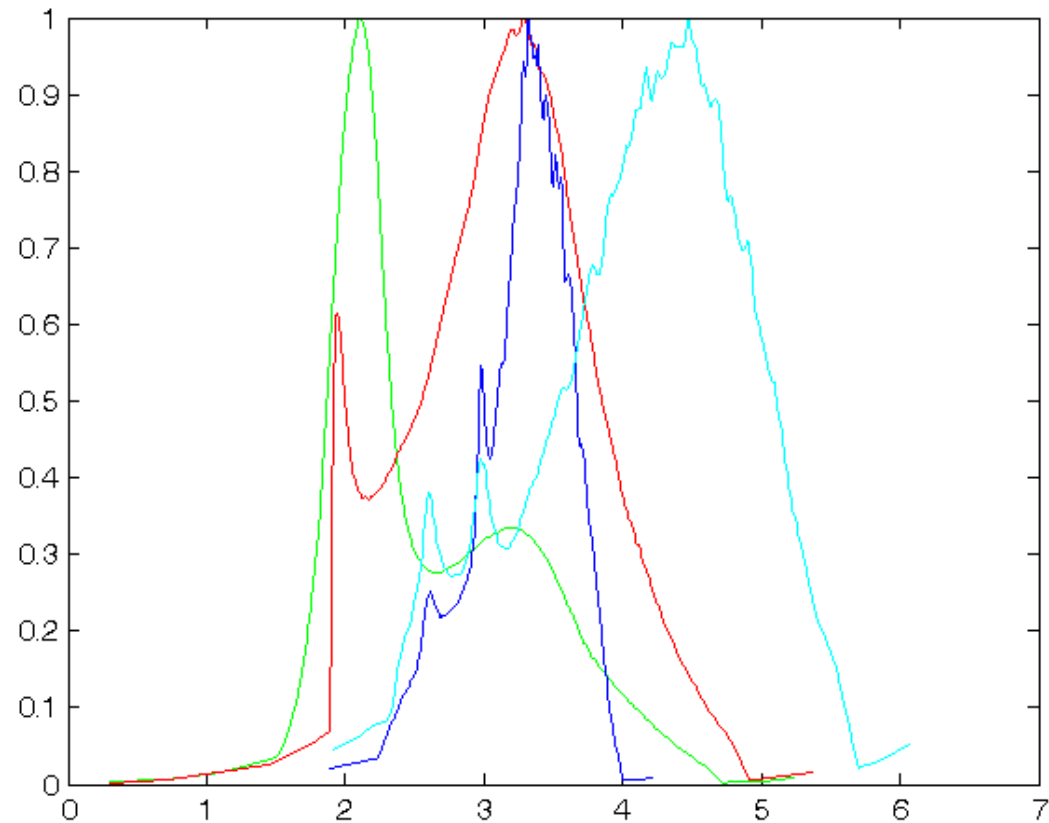
```
void mexFunction(int nlhs, mxArray *plhs[], int nrhs, const mxArray *prhs[]) {
    int i,M,N,numel;
    double *x,*y;
    if (nrhs==0) { // destructor
        if (agent) delete agent;
        agent = NULL;
        return;
    }
    if (!agent) agent = new IQagent;
    M = mxGetM(prhs[0]);
    N = mxGetN(prhs[0]);
    x = (double *)mxGetData(prhs[0]);
    numel = M*N;
    if (nlhs==0) { // assimilate data
        for (i=0;i<numel;i++) agent->add(*x++);
    } else { // report quantiles
        plhs[0] = mxCreateDoubleMatrix(M,N,mxREAL);
        y = (double *)mxGetData(plhs[0]);
        for (i=0;i<numel;i++) *y++ = agent->report(*x++);
    }
    return;
}
```

```

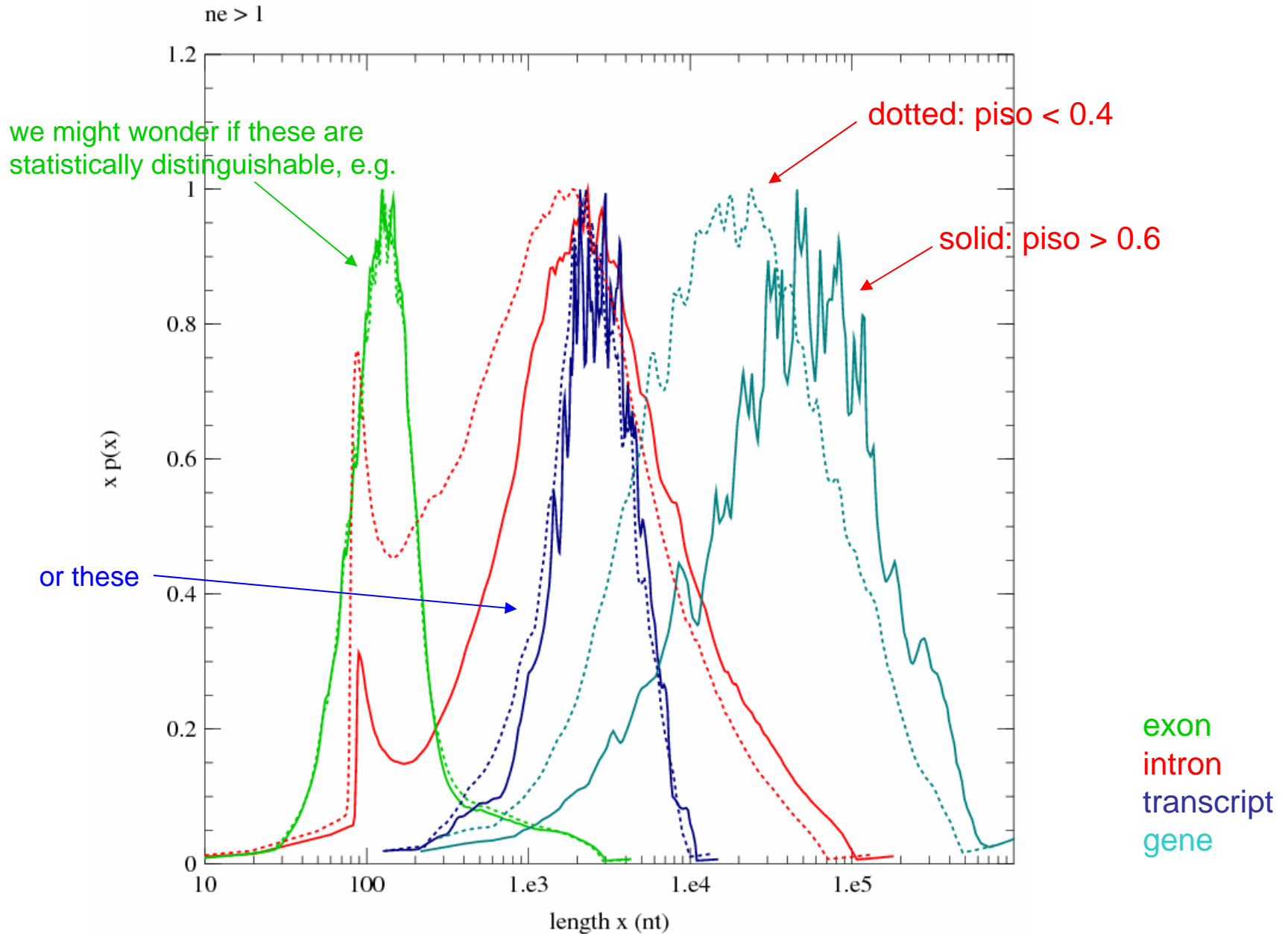
p = (. 5: 1: 499. 5)/500;
exons = cell2mat(g.exonlen);
iqagent(exons);
xx = iqagent(p);
rlo = iqagent(p-. 01);
rhi = iqagent(p+. 01);
yy = xx ./ (rhi -rlo);
yy = yy ./ max(yy);
plot(log10(xx), yy, 'g')
hold on
iqagent();
introns = cell2mat(g.intronlen);
iqagent(introns);
xx = iqagent(p);
rlo = iqagent(p-. 01);
rhi = iqagent(p+. 01);
yy = xx ./ (rhi -rlo);
yy = yy ./ max(yy);
plot(log10(xx), yy, 'r')
iqagent();
iqagent(g.elen);
xx = iqagent(p);
rlo = iqagent(p-. 01);
rhi = iqagent(p+. 01);
yy = xx ./ (rhi -rlo);
yy = yy ./ max(yy);
plot(log10(xx), yy, 'b')
iqagent();
iqagent(g.len);
xx = iqagent(p);
rlo = iqagent(p-. 01);
rhi = iqagent(p+. 01);
yy = xx ./ (rhi -rlo);
yy = yy ./ max(yy);
plot(log10(xx), yy, 'c')
hold off;

```

exon
intron
transcript
gene

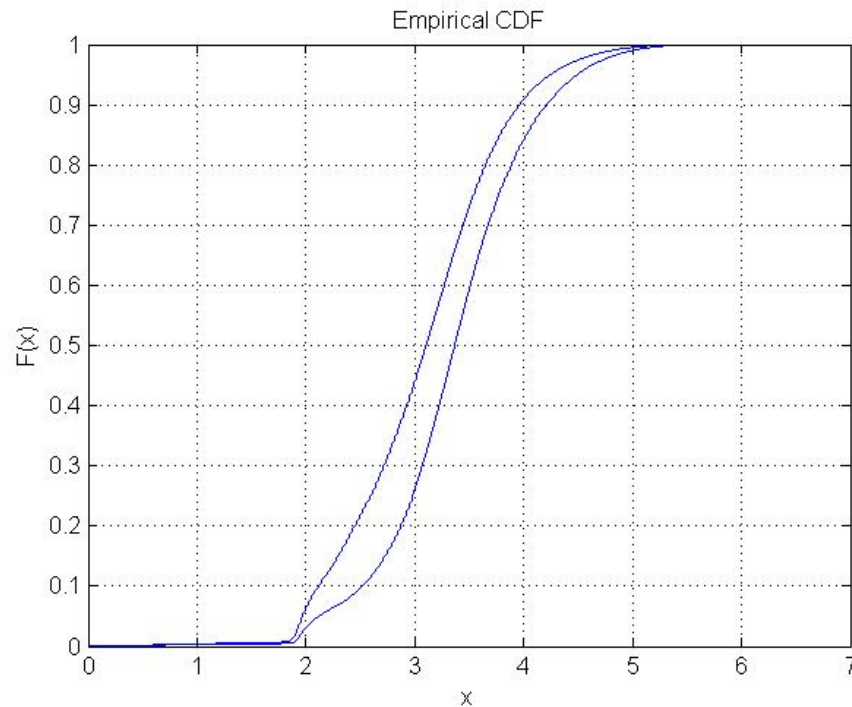


The extra variable " p_{iso} " is "probability that gene is in an AT isochore"



Let's look at the intron CDFs and then try a KS test

```
i l ena = cell2mat(g.intronlen(g.piso>0.6));  
i l enb = cell2mat(g.intronlen(g.piso<0.4));  
cdfplot(log10(i l ena))  
hold on  
cdfplot(log10(i l enb))  
hold off
```



```
[h p k] = kstest2(i l ena, i l enb)
```

```
h =
```

```
1
```

```
p =
```

```
0
```

```
k =
```

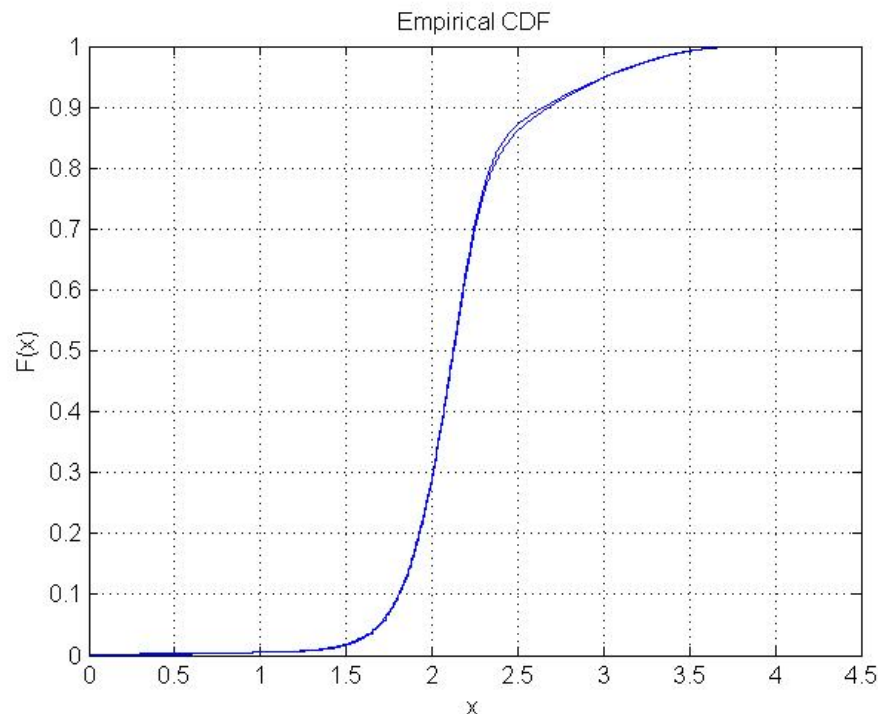
```
0.1807
```

null hypothesis so strongly ruled out that it displays as zero

How about exons?

```

el ena = cel l 2mat (g. exonl en(g. pi so>0. 6));
el enb = cel l 2mat (g. exonl en(g. pi so<0. 4));
cdfplot(log10(el ena))
hold on
cdfplot(log10(el enb))
hold off
    
```



```
[h p k] = kstest2(el ena, el enb)
```

```
h =
```

```
1
```

```
p =
```

```
8.6614e-006
```

```
k =
```

```
0.0131
```

```
[h p k] = kstest2(log10(el ena), log10(el enb))
```

```
h =
```

```
1
```

```
p =
```

```
8.6614e-006
```

```
k =
```

```
0.0131
```

KS test independent of parameterization, as it should be

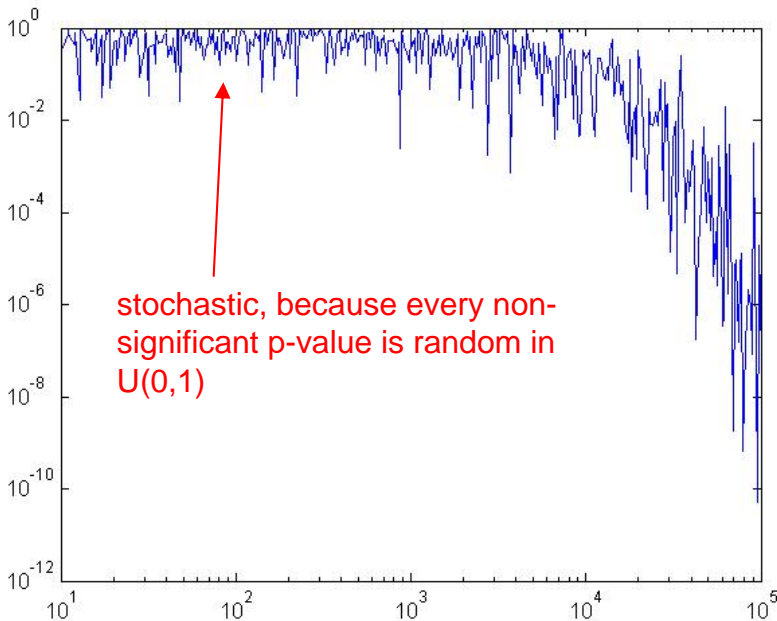
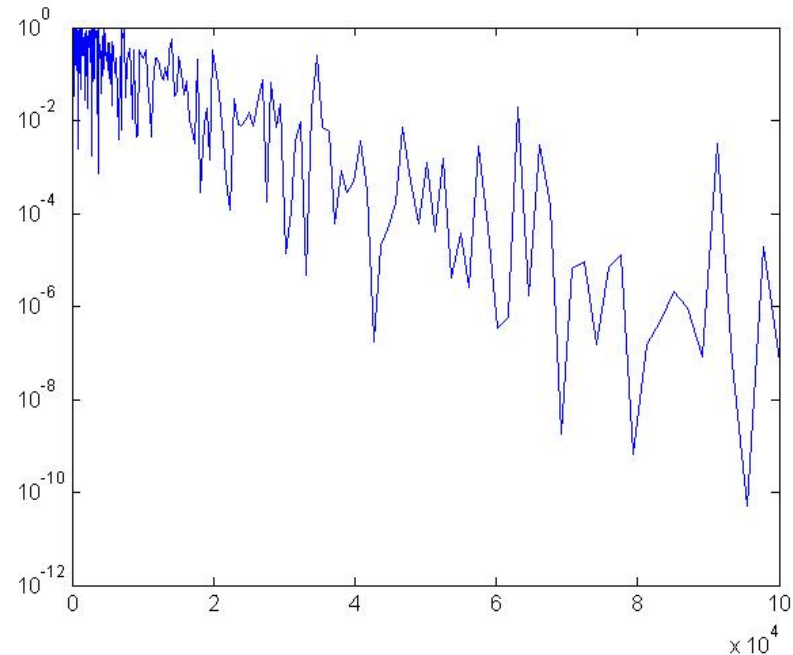
So these are “highly significantly different”. But also “virtually identical”. You could make both claims in a paper.

Editorial: It’s easy in science to find highly statistically significant, but negligibly unimportant, effects. Whether these are good science or not depends on (i) are they the faint (e.g., diluted) echo of an effect that you could show to be strong (and scientifically important) in some other setup, or (ii) is the field so exact (e.g., atomic physics) that tiny statistically significant effects can make or break a theory. If neither (i) nor (ii) is true, then good advice is not to waste a career on small “significant” effects! Biologists, this means you!

We got a highly significant result, because we had a lot of data.
 As an exercise, let's see how the significance would have deteriorated with sparser data.

```
function p = ksboot(dat1, dat2, nsamp)
samp1 = randsample(dat1, nsamp, true);
samp2 = randsample(dat2, nsamp, true);
[h p k] = kstest2(samp1, samp2);
```

```
ns = int32(10.^(1:0.01:5));
ps = arrayfun(@(x) ksboot(elena, elenb, x), ns);
loglog(ns, ps)
semilogx(ns, ps)
```



You get basically no significance up to a certain data size (here $\sim 1 \times 10^4$). Thereafter, the p-value decreases exponentially!

Editorial: Discoveries get made by increased data, rarely by fancy statistical analysis on existing data. Sadly (for us theorists).