

**CS395T**  
**Computational Statistics with**  
**Application to Bioinformatics**

Prof. William H. Press  
Spring Term, 2009

The University of Texas at Austin

**Unit 9: Multivariate Normal Distributions**

## Multivariate Normal Distributions

Generalizes Normal (Gaussian) to M-dimensions

Like 1-d Gaussian, completely defined by its mean and (co-)variance

Mean is a M-vector, covariance is a M x M matrix

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

Because mean and covariance are easy to estimate from a data set, it is easy – perhaps too easy – to fit a multivariate normal distribution to data.

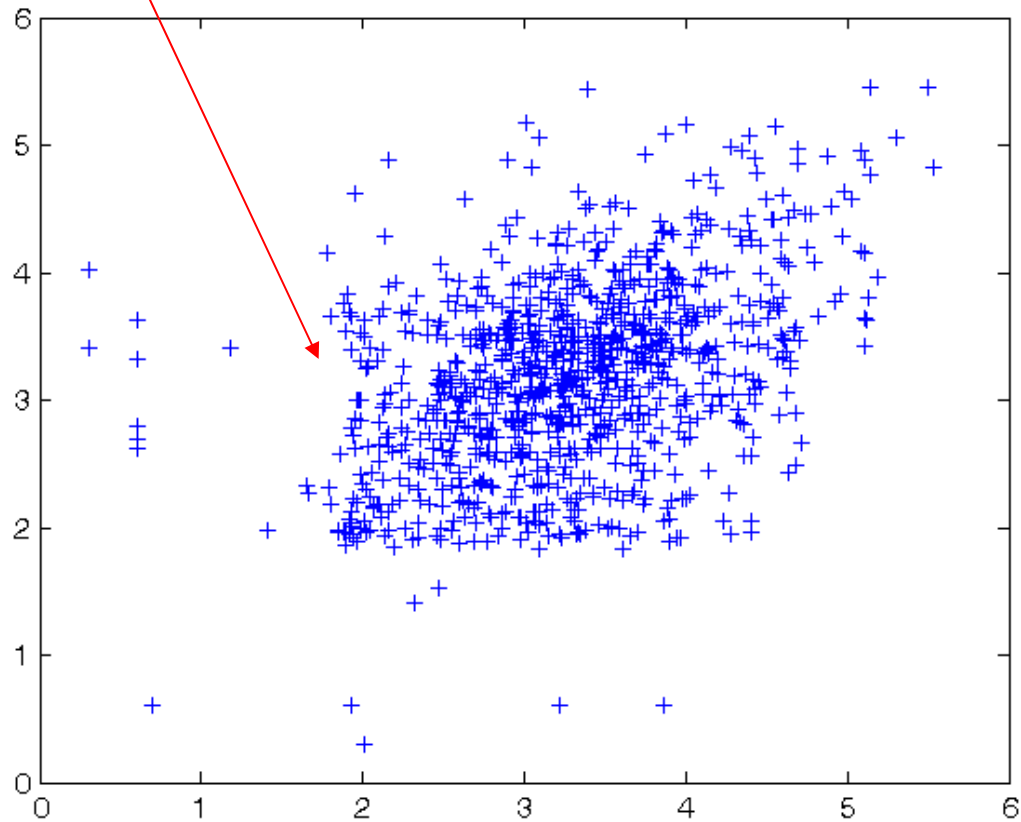
$$\boldsymbol{\mu} = \langle \mathbf{x} \rangle \qquad \boldsymbol{\Sigma} = \langle (\mathbf{x} - \boldsymbol{\mu}) \otimes (\mathbf{x} - \boldsymbol{\mu}) \rangle$$

Sample the sizes of 1<sup>st</sup> and 2<sup>nd</sup> introns for 1000 genes:

```
g = readgenestats('genestats.dat');
ggg = g(g.ne>2, :);
which = randsample(size(ggg, 1), 1000);
i1len = ggg.intronlen(which);
i1len = zeros(size(which));
i2len = zeros(size(which));
for j=1: numel(i1len), i1len(j) = log10(i1len{j}(1)); end;
for j=1: numel(i2len), i2len(j) = log10(i1len{j}(2)); end;
plot(i1len, i2len, '+')
hold on
```

This is kind of fun, because it's not just the usual featureless scatter plot

notice the biology!



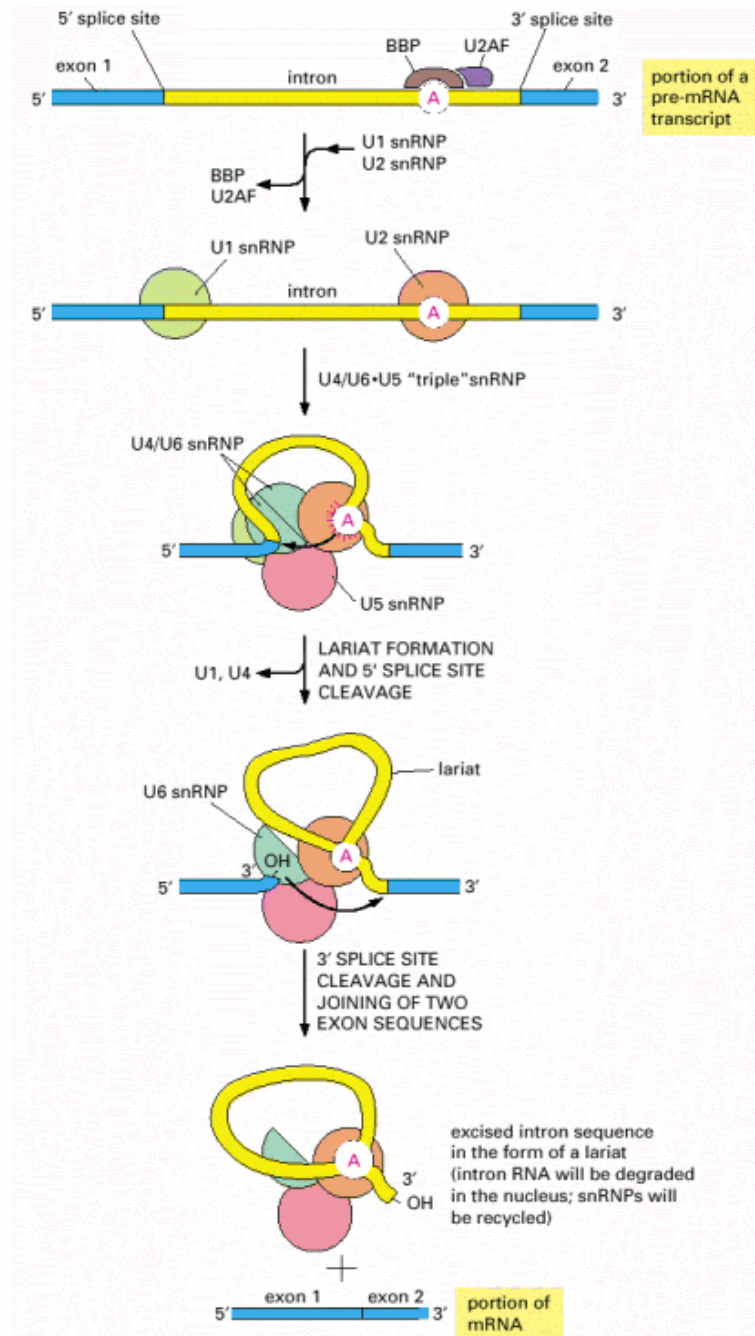
Is there a significant correlation here? (Yes, we'll see.)

## Biology:

The hard lower bounds on intron length are because the intron has to fit around the “big” spliceosome machinery!

It's all carefully arranged to allow exons of any length, even quite small.

Why? Could the spliceosome have evolved to require a minimum exon length, too? Are we seeing chance early history, or selection?



credit: Alberts et al.  
*Molecular Biology of the Cell*

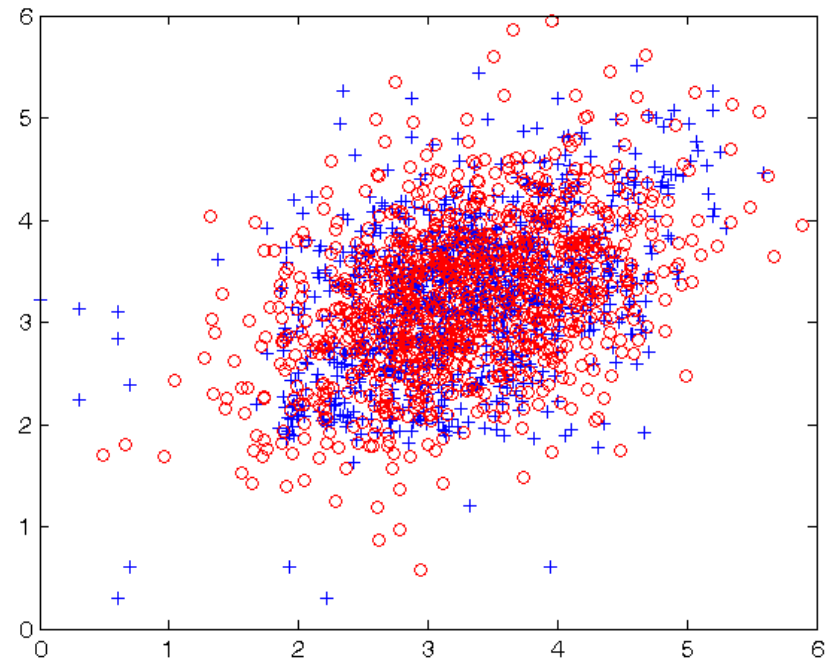
“Fitting” a multivariate normal just consists of computing the mean and covariance. Once fitted, one can also easily sample from it.

```
mu = mean([i 1 | en; i 2 | en], 2)
sig = cov(i 1 | en, i 2 | en)
mu =
    3.2844
    3.2483
sig =
    0.6125    0.2476
    0.2476    0.5458
rsamp = mvnrnd(mu, sig, 1000);
plot(rsamp(:, 1), rsamp(:, 2), 'or')
hold off
```

If renormalized, the covariance matrix is exactly the linear correlation matrix:

```
r = sig ./ sqrt(diag(sig) * diag(sig)')
tval = sqrt(numel(i 1 | en)) * r
r =
    1.0000    0.3843
    0.3843    1.0000
tval =
    31.6228    12.1511
    12.1511    31.6228
[rr p] = corrcoef(i 1 | en, i 2 | en)
rr =
    1.0000    0.3843
    0.3843    1.0000
p =
    1.0000    0.0000
    0.0000    1.0000
```

Matlab has built-ins



statistical significance of the correlation in standard deviations (but note: uses CLT)

$$r = \frac{\sum_i (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_i (x_i - \bar{x})^2} \sqrt{\sum_i (y_i - \bar{y})^2}}$$

## How to generate multivariate normal deviates:

There is a quite general way to construct a vector deviate  $\mathbf{x}$  with a specified covariance  $\Sigma$  and mean  $\mu$ , starting with a vector  $\mathbf{y}$  of independent random deviates of zero mean and unit variance: First, use Cholesky decomposition (§2.9) to factor  $\Sigma$  into a left triangular matrix  $\mathbf{L}$  times its transpose,

$$\Sigma = \mathbf{L}\mathbf{L}^T \quad (7.4.2)$$

This is always possible because  $\Sigma$  is positive-definite, and you need do it only once for each distinct  $\Sigma$  of interest. Next, whenever you want a new deviate  $\mathbf{x}$ , fill  $\mathbf{y}$  with independent deviates of unit variance and then construct

$$\mathbf{x} = \mathbf{L}\mathbf{y} + \mu \quad (7.4.3)$$

The proof is straightforward, with angle brackets denoting expectation values: Since the components  $y_i$  are independent with unit variance, we have

$$\langle \mathbf{y} \otimes \mathbf{y} \rangle = \mathbf{1} \quad (7.4.4)$$

where  $\mathbf{1}$  is the identity matrix. Then,

$$\begin{aligned} \langle (\mathbf{x} - \mu) \otimes (\mathbf{x} - \mu) \rangle &= \langle (\mathbf{L}\mathbf{y}) \otimes (\mathbf{L}\mathbf{y}) \rangle \\ &= \langle \mathbf{L}(\mathbf{y} \otimes \mathbf{y})\mathbf{L}^T \rangle = \mathbf{L} \langle \mathbf{y} \otimes \mathbf{y} \rangle \mathbf{L}^T \\ &= \mathbf{L}\mathbf{L}^T = \Sigma \end{aligned} \quad (7.4.5)$$

**So, just take:  $y_i \sim \mathcal{N}(0, 1)$  and you get a multivariate normal deviate!**

A related, useful, Cholesky trick is to draw error ellipses (ellipsoids, ...)

$$\Sigma = \mathbf{L} \cdot \mathbf{L}^T$$

So, locus of points at 1 standard deviation is

$$1 = (\mathbf{x} - \boldsymbol{\mu}) \cdot \boldsymbol{\Sigma}^{-1} \cdot (\mathbf{x} - \boldsymbol{\mu}) \quad \Rightarrow \quad |\mathbf{L}^{-1} \cdot (\mathbf{x} - \boldsymbol{\mu})| = 1$$

If  $\mathbf{z}$  is on the unit circle (sphere, ...) then

$$\mathbf{x} = \mathbf{L} \cdot \mathbf{z} + \boldsymbol{\mu}$$

```
function [x y] = errorellipse(mu, sigma, stdev, n)
L = chol(sigma, 'lower');
circle = [cos(2*pi*(0:n)/n); sin(2*pi*(0:n)/n)].*stdev;
ellipse = L*circle + repmat(mu, [1, n+1]);
x = ellipse(1, :);
y = ellipse(2, :);
```

```
plot(i1len, i2len, '+b');
hold on
[xx yy] = errorellipse(mu, sig, 1, 100);
plot(xx, yy, '-r');
[xx yy] = errorellipse(mu, sig, 2, 100);
plot(xx, yy, '-r');
```

