B I O I N F O R M A T I C S

Kristel Van Steen, PhD[2]

Montefiore Institute - Systems and Modeling

GIGA - Bioinformatics

ULg

kristel.vansteen@ulg.ac.be

CHAPTER 5: DNA SEQUENCE ANALYSIS

1 Introduction

1.a The biological problem

Relevant questions

Given the DNA sequence

AATCGGATGCGCGTAGGATCGGTAGGGTAGGCTTTAAGATCATGCTATTTTCGAGA
TTCGATTCTAGCTAGGTTTAGCTTAGCTTAGTGCCAGAAATCGGATGCGCGTAGGAT
CGGTAGGGTAGGCTTTAAGATCATGCTATTTTCGAGATTCGATTCTAGCTAGGTTTT
TAGTGCCAGAAATCGTTAGTGCCAGAAATCGATT

..

many questions arise

Relevant questions

- What sort of sequence might this be?

- Is it likely to be a gene?
    - What is the possible expression level?
    - What is the possible protein product?
    - Can we get the protein product?
    - Can we figure out the key residue in the protein product?
- What sort of statistics to be used for *describing* this sequence?
- Do parameters describing the sequence differ from those describing bulk DNA in that organism?

- Can we determine the organism from which this sequence came?

What will WE learn from sequence data?

- Recognizing **motifs, sites, signals, domains**
  - Sequence motifs are extremely convenient descriptors of conserved, functionally important short portions of proteins
  - A conserved motif can be represented by a *consensus sequence*
  - Often the above words (in bold) are used interchangeably to describe recurring elements of interest (patterns – **pattern recognition**)

- Why **(probability) models** for biomolecular motifs?
  - To characterize them
  - To help identify them
  - For incorporation into larger models (e.g., an entire gene)

How does DNA sequence motif discovery work?

Patrik D'haeseleer

How can we computationally extract an unknown motif from a set of target sequences? What are the principles behind the major motif discovery algorithms? Which of these should we use, and how do we know we've found a 'real' motif?

Extracting regulatory motifs[1] from DNA sequences seems to be all the rage these days. Take your favorite cluster of coexpressed genes, and with some luck you might hope to find a short pattern of nucleotides upstream of the transcription start sites of these genes, indicating a common transcription factor binding site responsible for their coordinate regulation. Easier said than done—the hunt for such a

of occurrences of all n-mers in the target sequences, and calculate which ones are most overrepresented. A motif description based on exact occurrence of specific words is too rigid for most real-world binding sites, but a number of similar overrepresented words may be combined into a more flexible motif description. Alternatively, one can search the space of all degenerate consensus

3-nucleotide degenerate positions in the motif[2]. Another enumerative approach describes a motif as a consensus sequence and an allowed number of mismatches, and uses an efficient suffix tree representation to find all such motifs in the target sequences[3]. Enumerative methods cover the entire search space, and therefore do not run the risk of getting stuck in a local optimum. On

(D'Haeseleer2006)

Towards a theoretical understanding of false positives in DNA motif finding

Amin Zia and Alan M Moses*

Abstract

Background: Detection of false-positive motifs is one of the main causes of low performance in *de novo* DNA motif-finding methods. Despite the substantial algorithm development effort in this area, recent comprehensive benchmark studies revealed that the performance of DNA motif-finders leaves room for improvement in realistic scenarios.

Results: Using large-deviations theory, we derive a remarkably simple relationship that describes the dependence of false positives on dataset size for the one-occurrence per sequence motif-finding problem. As expected, we predict that false-positives can be reduced by decreasing the sequence length or by adding more sequences to the dataset. Interestingly, we find that the false-positive strength depends more strongly on the number of sequences in the dataset than it does on the sequence length, but that the dependence on the number of sequences diminishes, after which adding more sequences does not reduce the false-positive rate significantly. We compare our theoretical predictions by applying four popular motif-finding algorithms that solve the one-occurrence-per -sequence problem (MEME, the Gibbs Sampler, Weeder, and GIMSAN) to simulated data that contain no motifs. We find that the dependence of false positives detected by these softwares on the motif-finding parameters is similar to that predicted by our formula.

(Zia and Moses 2012)

What will WE learn from sequence data?

- **Comparative genomics**:
  the study of the genomic sequence of organisms that are related to humans – could ultimately help to identify targets for drug development

LEADING EDGE

SHAKING THE TREE OF LIFE

Comparative genomics – the study of the genomic sequence of organisms that are related to humans – could ultimately help to identify targets for drug development. **BY JACK MCCAIN,** Contributing Editor

Confucius said that the measure of man is man, but curious creatures may be useful yardsticks in determining the workings of the human body. Careful comparisons of the tree (Figures 1–4). Note that the tree's true shape is unknown in many instances and is subject to substantial ongoing revision.

The National Human Genome Research Institute (NHGRI), part of the National Institutes of Health, is Genome Research, Cambridge, Mass.; The Institute for Genomic Research [TIGR], Rockville, Md.; Washington University Medical Center, St. Louis). Organisms selected for sequencing include many with a long history of use as models

(McCain 2004)

What will WE learn from sequence data?

- Identifying relevant mutations:
  - a germline mutation is one the was passed on to offspring because the egg or sperm cell was mutated.
  - a somatic mutation is a mutation of the somatic cells (all cells except sex cells) that cannot be passed on to offspring.

A survey of tools for variant analysis of next-generation genome sequencing data

Stephan Pabinger, Andreas Dander, Maria Fischer, Rene Snajder, Michael Sperk, Mirjana Efremova, Birgit Krabichler, Michael R. Speicher, Johannes Zschocke and Zlatko Trajanoski

Submitted: 20th August 2012; Received (in revised form): 4th December 2012

Abstract

Recent advances in genome sequencing technologies provide unprecedented opportunities to characterize individual genomic landscapes and identify mutations relevant for diagnosis and therapy. Specifically, whole-exome sequencing using next-generation sequencing (NGS) technologies is gaining popularity in the human genetics community due to the moderate costs, manageable data amounts and straightforward interpretation of analysis results. While whole-exome and, in the near future, whole-genome sequencing are becoming commodities, data analysis still poses significant challenges and led to the development of a plethora of tools supporting specific parts of the analysis workflow or providing a complete solution. Here, we surveyed 205 tools for whole-genome/whole-exome sequencing data analysis supporting five distinct analytical steps: quality assessment, alignment, variant identification, variant annotation and visualization. We report an overview of the functionality, features and specific requirements of the individual tools. We then selected 32 programs for variant identification, variant annotation and

(Pabinger et al 2013)

Recall – Chapter 4:

Shaking the tree: mapping complex disease genes with linkage disequilibrium

Lyle J Palmer, Lon R Cardon

Much effort and expense are being spent internationally to detect genetic polymorphisms contributing to susceptibility to complex human disease. Concomitantly, the technology for detecting and genotyping single nucleotide polymorphisms (SNPs) has undergone rapid development, yielding extensive catalogues of these polymorphisms across the genome. Population-based maps of the correlations amongst SNPs (linkage disequilibrium) are now being developed to accelerate the discovery of genes for complex human diseases. These genomic advances coincide with an increasing recognition of the importance of very large sample sizes for studying genetic effects. Together, these new genetic and epidemiological data hold renewed promise for the identification of susceptibility genes for complex traits. We review the state of knowledge about the structure of the human genome as related to SNPs and linkage disequilibrium, discuss the potential applications of this knowledge to mapping complex disease genes, and consider the issues facing whole genome association scanning using SNPs.

Genomic approaches to disease association mapping

Genomics is transforming epidemiology, medicine, and drug discovery,[1–7] and attention is being directed towards population-based genetic association studies for complex phenotypes.[3,8–12] For many complex conditions, the genetic expediting the discovery of complex human disease genes. We review knowledge about the human genome as related to SNPs and linkage disequilibrium (LD), discuss the potential applications of this knowledge to mapping complex disease genes, and look at the feasibility of whole genome association using SNPs.

(Palmer and Cardon 2005)

Recall – Chapter 5:

Public resources that help in mapping complex diseases in humans

http://www.ncbi.nlm.nih.gov/ - The National Center for Biotechnology Information (**NCBI**) advances science and health by providing access to biomedical and genomic information.
http://www.ensembl.org/index.html – The **Ensembl** project produces genome databases for vertebrates and other eukaryotic species, and makes this information freely available online.
http://hapmap.ncbi.nlm.nih.gov/ – **HapMap** – multi-country effort to identify and catalog genetic similarities and differences in human beings.
http://lynx.ci.uchicago.edu/ - **LYNX** – Gene Annotations, Enrichment Analysis and Genes Prioritization.
http://www.genemania.org/ – **GeneMANIA** - Indexing 1,421 association networks containing 266,984,699 interactions mapped to 155,238 genes from 7 organisms.

ARTICLE SERIES: **Applications of next-generation sequencing**

Sequencing studies in human genetics: design and interpretation

David B. Goldstein, Andrew Allen, Jonathan Keebler, Elliott H. Margulies, Steven Petrou, Slavé Petrovski & Shamil Sunyaev

Nature Reviews Genetics **14**, 460–470 (2013) doi:10.1038/nrg3455

Published online 11 June 2013

Abstract

Next-generation sequencing is becoming the primary discovery tool in human genetics. There have been many clear successes in identifying genes that are responsible for Mendelian diseases, and sequencing approaches are now poised to identify the mutations that cause undiagnosed childhood genetic diseases and those that predispose individuals to more common complex diseases. There are, however, growing concerns that the complexity and magnitude of complete sequence data could lead to an explosion of weakly justified claims of association between genetic variants and disease. Here, we provide an overview of the basic workflow in next-generation sequencing studies and emphasize, where possible, measures and considerations that facilitate accurate inferences from human sequencing studies.

Recall: Different cell types

- *Eukaryotes*: organisms with a rather complex cellular structure.

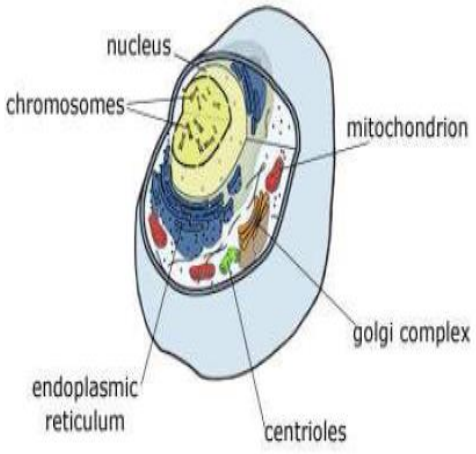
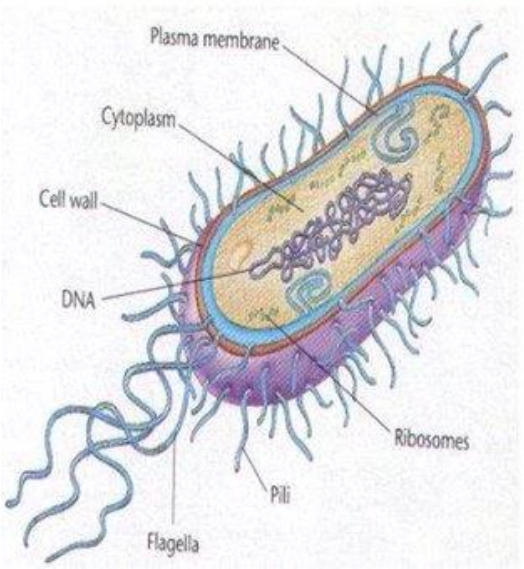


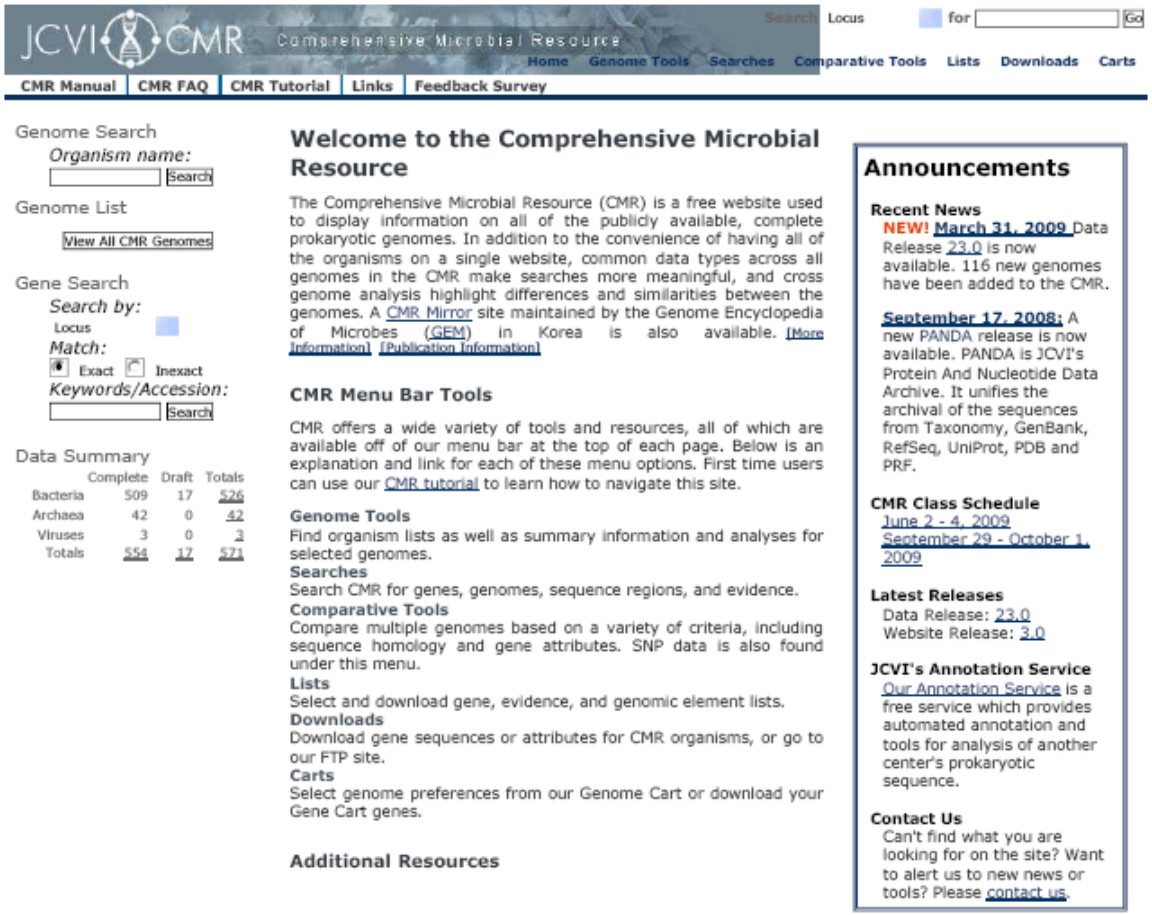
Image adapted from: National Human Genome Research Institute.

A typical eukaryotic cell.



- *Prokaryotes*: cells without organelles where the genetic information floats freely in the cytoplasm

Despite the importance of "knowing" bacterial compositions



(http://cmr.jcvi.org/tigr-scripts/CMR/CmrHomePage.cgi)

we will focus on human data and Bioconductor / R Environment

Methods to answer questions related to pattern recognition

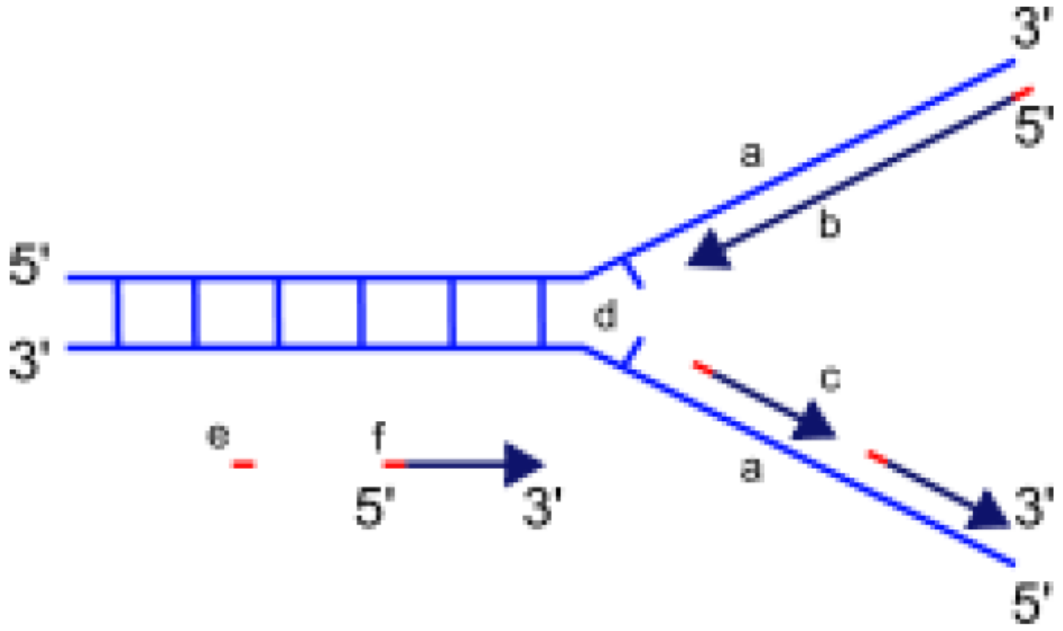
For instance by investigating frequencies of occurrences of words

Words

- Words are short strings of letters drawn from an alphabet
- In the case of DNA, the set of letters is A, C, T, G
- A word of length k is called a k-word or k-tuple
- Differences in word frequencies help to differentiate between different DNA sequence sources or regions
- Examples:
  - 1-tuple: individual nucleotide
  - 2-tuple: dinucleotide
  - 3-tuple: codon

1.b Biological words of length 1 or base composition

Recall – Chapter 1 - DNA replication

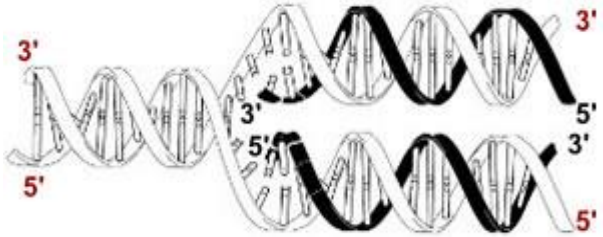


Scheme of the replication fork

a: template, b: leading strand, c: lagging strand, d: replication fork, e: primer, f: Okazaki fragments (short, newly synthesized DNA fragments that are formed on the lagging template strand during DNA replication)

Biological words of length 1

- There are constraints on base composition imposed by the genetic code
- The distribution of individual bases within a DNA molecule is not ordinarily uniform
  - There may be an excess of G over C on the leading strands



  - This can be described by the “GC skew”, characterized by:
    - (#G - #C) / (#G + #C)
    - # = nr of

2 Probability theory revisited

2.a Probability distributions

Introduction

- Probability distributions are a fundamental concept in statistics. They are used both on a theoretical level and a practical level.
- Some practical uses of probability distributions are:
  - to calculate confidence intervals for parameters and
  - to calculate critical regions for hypothesis tests.

Introduction

- Statistical intervals and hypothesis tests are often based on specific distributional assumptions.
- Before computing an interval or test based on a distributional assumption, we need to verify that the assumption is justified for the given data set.
- For this chapter, the distribution does not always need to be the best-fitting distribution for the data, but an adequate enough model so that the statistical technique yields valid conclusions.
- Simulation studies: one way to obtain empirical evidence for a probability model

Assumptions

- Simple rules specifying a probability model:
  - First base in sequence is either A, C, T or G with prob p_A, p_C, p_T, p_G
  - Suppose the first r bases have been generated, while generating the base at position r+1, no attention is paid to what has been generated before.
- A, C, T or G is generated with the probabilities above
- Notation for the output of a random string of *n* bases may be: $L_1, L_2, \ldots, L_n$ (L_i = base inserted at position *i* of the sequence)
- Whatever we would like to do with such strings, we will need to introduce the concept of a random variable

Probability distributions

- Suppose the "machine" we are using produces an output X that takes exactly 1 of the J possible values in a set $\chi = \{x_1, x_2, \ldots, x_n\}$
  - In the DNA sequence J=4 and $\chi = \{A, C, T, G\}$
  - X is a discrete random variables (since its values are uncertain)
  - If p_j is the prob that the value (realization of the random variable *X*) x_j occurs, then
    - $p_1, \ldots, p_J \geq O$ and $p_1 + \ldots + p_J = 1$
- The probability distribution (probability mass function) of X is given by the collection $p_1, \ldots, p_J$
  - $P(X=x_j) = p_j$, j=1, …, J
- The probability that an event S occurs (subset of χ) is P(X ∈ S) = $\sum_{j:x_j \in S} (p_j)$

Probability distributions

- What is the probability distribution of the number of times a given pattern occurs in a random DNA sequence $L_1, \ldots, L_n$?
  - New sequence $X_1, \ldots, X_n$:

$$X_i=1 \text{ if } L_i=A \text{ and } X_i=0 \text{ else}$$

  - The number of times N that A appears is the sum

$$N=X_1+\ldots+X_n$$

  - The prob distr of each of the X_i:

$$P(X_i=1) = P(L_i=A)=p_A$$

$$P(X_i=0) = P(L_i=C \text{ or } G \text{ or } T) = 1 - p_A$$

- What is a "typical" value of N?
  - Depends on how the individual X_i (for different *i*) are interrelated

Independence

- Discrete random variables X_1, …, X_n are said to be independent if for any subset of random variables and actual values, the joint distribution equals the product of the component distributions
- According to our simple model, the L_i are independent and hence

$$P(L_1=l_1,L_2=l_2, \ldots,L_n=l_n)=P(L_1=l_1)\ P(L_2=l_2)\ \ldots P(L_n=l_n)$$

Expected values and variances

- Mean and variance are two important properties of real-valued random variables and corresponding probability distributions.
- The "mean" of a discrete random variable X taking values x_1, x_2, . . . (denoted EX (or E(X) or E[X]), where E stands for expectation, which is another term for mean) is defined as:

$$EX = \sum_i x_i P(X = x_i).$$

  - $EX_i = 1 \times p_A + 0 \times (1 - p_A)$
  - If Y=cX, then EY = cEX
  - $E(X_1 + \ldots + X_n) = EX_1 + \ldots + EX_n$
- Because X_i are assumed to be independent and identically distributed (iid):

$$E(X_1 + \ldots + X_n) = nEX_1 = np_A$$

Expected values and variances

- The idea is to use squared deviations of X from its center (expressed by the mean). Expanding the square and using the linearity properties of the mean, the Var(X) can also be written as:

$$\mathrm{Var}(X) = E\left[X^2\right] - [EX]^2$$

  - If Y=cX then VarY = c^2VarX
  - The variance of a sum of independent random variables is the sum of the individual variances

- For the random variables X_i:
  $\mathrm{Var}X_i = [1^2 \times p_A + 0^2 \times' (1 - p_A)] - p_A^2 = p_A(1 - p_A)$
  $\mathrm{Var}N = n\mathrm{Var}X_1 = np_A(1 - p_A)$

Expected values and variances

- The expected value of a random variable X gives a measure of its location. Variance is another property of a probability distribution dealing with the spread or variability of a random variable around its mean.

$$\mathrm{Var}(X) = E\left[X - EX\right]^2$$

  - The positive square root of the variance of X is called its standard deviation sd(X)

The binomial distribution

- The binomial distribution is used when there are exactly two mutually exclusive outcomes of a trial. These outcomes are appropriately labeled "success" and "failure". The binomial distribution is used to obtain the probability of observing x successes in a fixed number of trials, with the probability of success on a single trial denoted by p. The binomial distribution assumes that p is fixed for all trials.
- The formula for the binomial probability mass function is :

$$P(N = j) = \binom{n}{j} p^j (1-p)^{n-j}, \text{ j = 0,1, ...,n}$$

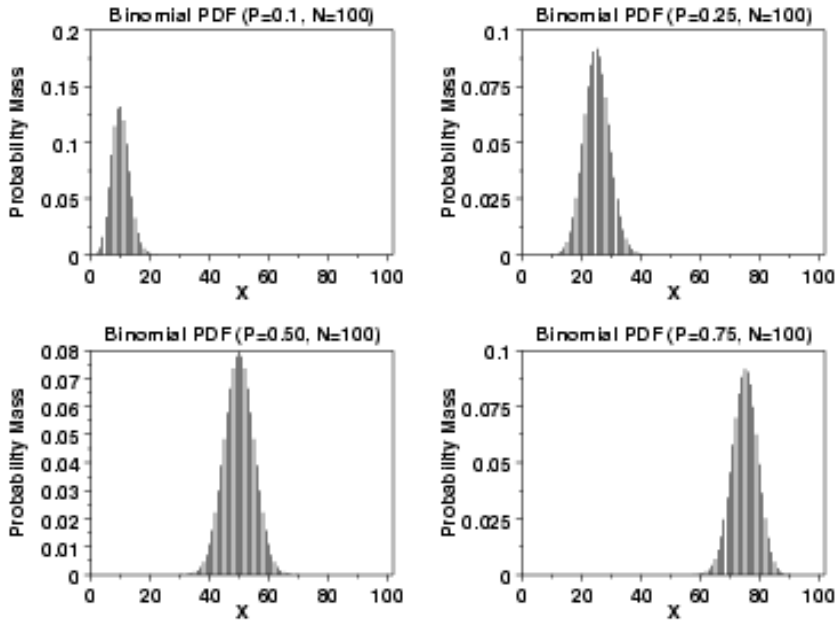
with the binomial coefficient $\binom{n}{j}$ determined by

$$\binom{n}{j} = \frac{n!}{j!\,(n-j)!},$$

and j!=j(j-1)(j-2)…3.2.1, 0!=1

The binomial distribution

- The mean is np and the variance is np(1-p)
- The following is the plot of the binomial probability density function for four values of p and n = 100.



2.b Simulating from probability distributions

- The idea is that we can study the properties of the distribution of N when we can get our computer to output numbers N_1, …, N_n having the same distribution as N
  - We can use the sample mean to estimate the expected value EN:

$$\bar{N} = (N_1 + \ldots + N_n)/n$$

  - Similarly, we can use the sample variance to estimate the true variance of N:

$$s^2 = \frac{1}{n-1}\sum_{i=1}^{n}(N_i - \bar{N})^2$$

Why do we use (n-1) and not n in the denominator?

Simulating from probability distributions

- What is needed to produce such a string of observations?
  - Access to pseudo-random numbers: random variables that are uniformly distributed on (0,1): any number between 0 and 1 is a possible outcome and each is equally likely
- In practice, simulating an observation with the distribution of X_1:
  - Take a uniform random number u
  - Set X_1=1 if $U \leq p \equiv p_A$ and 0 otherwise.
  - Why does this work? … $P(X_1 = 1) = P(U \leq p_A) = p_A$
  - Repeating this procedure n times results in a sequence X_1, …, X_n from which N can be computed by adding the X's

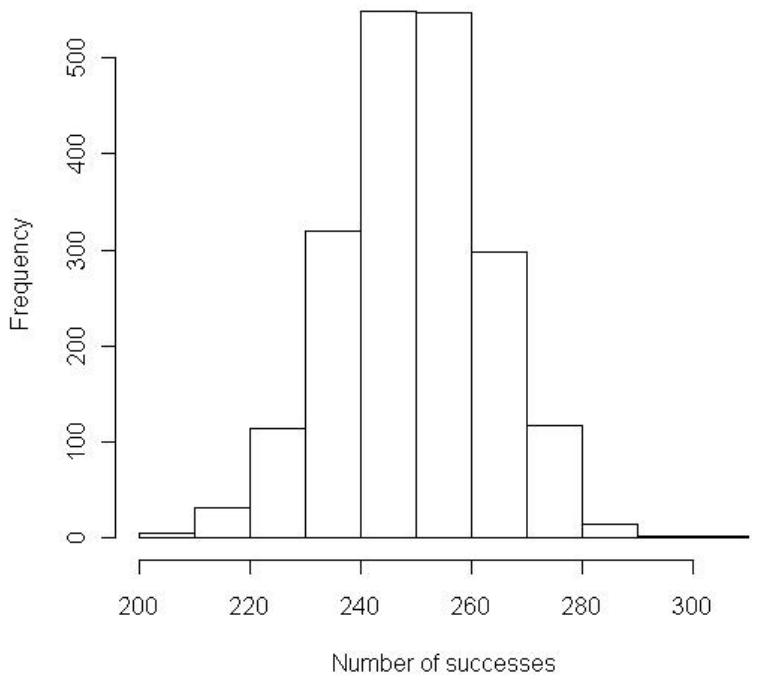
Simulating from probability distributions

- Simulate a sequence of bases L_1, …, L_n:
  - Divide the interval (0,1) in 4 intervals with endpoints
    $$p_A, p_A + p_C, p_A + p_C + p_G, 1$$
  - If the simulated u lies in the leftmost interval, L_1=A
  - If u lies in the second interval, L_1=C; if in the third, L_1=G and otherwise L_1=T
  - Repeating this procedure n times with different values for U results in a sequence L_1, …, L_n
- Use the "sample" function in R:

```
pi <- c(0.25,0.75)
x<-c(1,0)
set.seed(2009)
sample(x,10,replace=TRUE,pi)
```

Simulating from probability distributions

- By looking through a given simulated sequence, we can count the number of times a particular pattern arises (for instance, the base A)
- By repeatedly generating sequences and analyzing each of them, we can get a feel for whether or not our particular pattern of interest is unusual



Simulating from probability distributions

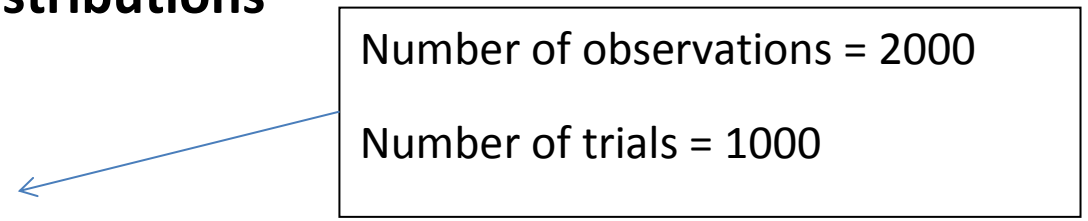
- Using R code:

```
x<- rbinom(2000,1000,0.25)
mean(x)
sd(x)^2
hist(x,xlab="Number of successes",main="")
```

What is the number of observations?

Simulating from probability distributions

- Using R code:




```
x<- rbinom(2000,1000,0.25)
mean(x)
sd(x)^2
hist(x,xlab="Number of successes",main="")
```

What is the number of observations?

- Suppose we have a sequence of 1000bp and assume that every base occurs with equal probability. How likely are we to observe at least 300 A's in such a sequence?
  - Exact computation using a closed form of the relevant distribution
  - Approximate via simulation
  - Approximate using the Central Limit Theory

Exact computation via closed form of relevant distribution

- The formula for the binomial probability mass function is :

$$P(N = j) = \binom{n}{j} p^j (1-p)^{n-j}, \text{ j = 0,1, …,n}$$

and therefore

$$P(N \geq 300) = \sum_{j=300}^{1000} \binom{1000}{j} (1/4)^j (1-1/4)^{1000-j}$$

$$= 0.00019359032194965841$$

	P: exactly 300 out of 1000
Method 1. exact binomial calculation	0.00004566114740576488
Method 2. approximation via normal	0.000038
Method 3. approximation via Poisson	------
	P: 300 or fewer out of 1000
Method 1. exact binomial calculation	0.9998520708293378
Method 2. approximation via normal	0.999885
Method 3. approximation via Poisson	------
	P: 300 or more out of 1000
Method 1. exact binomial calculation	0.00019359032194965841
Method 2. approximation via normal	0.000153
Method 3. approximation via Poisson	------

For hypothesis testing

	P: 300 or more out of 1000	
	One-Tail	Two-Tail
Method 1. exact binomial calculation	0.00019359032194965841	0.0003025705168772097
Method 2. approximation via normal	0.000153	0.000306
Method 3. approximation via Poisson	------	------

(http://faculty.vassar.edu/lowry/binomialX.html)

Approximate via simulation

- Using R code and simulations from the theoretical distribution, $P(N \geq 300)$ can be estimated as 0.000196 via.

```
x<- rbinom(1000000,1000,0.25)
sum(x>=300)/1000000
```

or 0.0001479292 via

```
1-pbinom(300,size=1000,prob=0.25)
pbinom(300,size=1000,prob=0.25,lower.tail=FALSE)
```

Approximate via Central Limit Theory

- The central limit theorem offers a 3rd way to compute probabilities of a distribution
- It applies to sums or averages of iid random variables
- Assuming that X_1, …, X_n are iid random variables with mean μ and variance σ^2, then we know that for the sample average

$$\bar{X}_n = \frac{1}{n}(X_1 + \ldots + X_n),$$

$$\mathrm{E}\bar{X}_n = \mu \text{ and } \mathrm{Var}\ \bar{X}_n = \frac{\sigma^2}{n}$$

- Hence,

$$E\left(\frac{\bar{X}_n - \mu}{\sigma/\sqrt{n}}\right) = 0, Var\left(\frac{\bar{X}_n - \mu}{\sigma/\sqrt{n}}\right) = 1$$

Approximate via Central Limit Theory

- The central limit theorem states that if the sample size n is large enough,

$$P\left(a \leq \frac{\bar{X}_n - \mu}{\frac{\sigma}{\sqrt{n}}} \leq b\right) \approx \phi(b) - \phi(a),$$

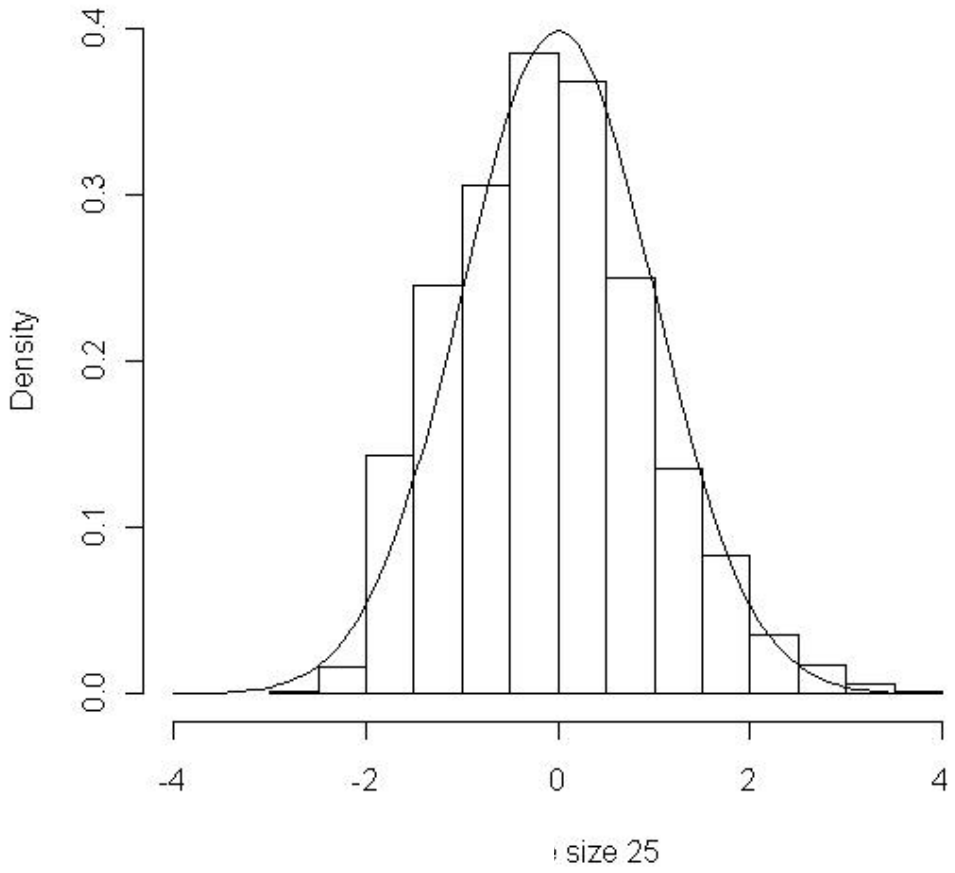
with $\phi(.)$ the standard normal distribution defined as

$$\phi(z) = P(Z \leq z) = \int_{-\infty}^{z} \phi(x)dx$$

- The central limit theorem in action using R code:

```
bin25<-rbinom(1000,25,0.25)
av.bin25 <- 25*0.25
stdev.bin25 <- sqrt(25*0.25*0.75)
bin25<-(bin25-av.bin25)/stdev.bin25
hist(bin25,xlim=c(-4,4),ylim=c(0.0,0.4),prob=TRUE,xlab="Sample size
25",main="")
x<-seq(-4,4,0.1)
lines(x,dnorm(x))
```

Approximate via Central Limit Theory



Approximate via Central Limit Theory

- Estimating the quantity $P(N \geq 300)$ when N has a binomial distribution with parameters n=1000 and p=0.25,

$$E(N) = n\mu = 1000 \times 0.25 = 250,$$

$$sd(N) = \sqrt{n}\,\sigma = \sqrt{1000 \times \frac{1}{4} \times \frac{3}{4}} \approx 13.693$$

$$P(N \geq 300) = P\left(\frac{N - 250}{13.693} > \frac{300 - 250}{13.693}\right)$$

$$\approx P(Z > 3.651501) = 0.0001303560$$

- R code:

```
pnorm(3.651501,lower.tail=FALSE)
```

How do the estimates of $P(N \geq 300)$ compare?

3 Biological words of length 2

Introduction

- Dinucleotides are important because physical parameters associated with them can describe the trajectory of the DNA helix through space (such as DNA bending). This may affect gene expression.
- Concentrating on abundances, and assuming the iid model for L_1, …, L_n:

$$P(L_i = l_i, L_{i+1} = l_{i+1}) = p_{l_i}\, p_{l_{i+1}}$$

- Has a given sequence an unusual dinucleotide frequency compared to the iid model?
  - Compare observed O with expected E dinucleotide numbers

$$\chi^2 = \frac{(\mathrm{O}-\mathrm{E})^2}{\mathrm{E}},$$

with $E = (n-1)p_{l_i}p_{l_{i+1}}$.

Why (n-1) as factor? How many df?

Comparing to the reference

- How to determine which values of χ^2 are unlikely or extreme?
  - Recipe:
    - Compute the number c given by

$$c = \begin{cases} 1 + 2p_{l_i} - 3p_{l_i}^2, & \text{if } l_i = l_{i+1} \\ 1 - 3p_{l_i}p_{l_{i+1}}, & \text{if } l_i \neq l_{i+1} \end{cases}$$

    - Calculate the ratio $\frac{\chi^2}{c}$, where χ^2 is given as before
    - If this ratio is larger than 3.84 then conclude that the iid model is not a good fit
    - Note: qchisq(0.95,1) = 3.84

4 Markov Chains

Introduction

- When moving from bacteria (such as **E**. **coli**, a common type of bacteria that can get into food, like beef and vegetables) to real genomes, a more complicated probabilistic model is required then the iid model before to capture the dinucleotide properties
- One approach is to use Markov chains.
- Markov chains are a direct generalization of independent trials, where the character at a position may depend on the characters of preceding positions, hence may be conditioned on preceding positions

Conditional probabilities

- If Ω refers to the set of all possible outcomes of a single experiment, A to a particular event, and A^c to the complement Ω-A of A, then

$$P(A) + P(A^c) = 1,$$

- The conditional probability of A given B is $P(A|B) = \frac{P(A \cap B)}{P(B)}, P(B) > 0$
- As a consequence: $P(B|A) = \frac{P(A|B)\,P(B)}{P(A)}$, also known as Bayes' Theorem
- For B_1, …, B_k forming a partition of Ω, this is the B_i are disjoint and the Bi are exhaustive (their union Is Ω), the law of total probability holds:

$$P(A) = \sum_{i=1}^{k} P(A \cap Bi) = \sum_{i=1}^{k} P(A|Bi)\,P(Bi)$$

The Markov property

- The property will be explained via studying a sequence of random variables X_t, t=0,1,… taking values in the state space {A,C,T,G}
- The sequence $\{X_t, t \geq 0\}$ is called a **first-order Markov chain** if only the previous neighbor influences the probability distribution of the character at any position, and hence satisfies the Markov property:

$$P(X_{t+1} = j \,|X_t = i, X_{t-1} = i_{t-1}, \dots , \quad X_0 = i_0) = P(X_{t+1} = j|X_t = i)$$

for $t \geq 0$ and for all $i, j, i_{t-1}, \dots, i_0$ in the state space

- We consider Markov chains that are **homogeneous**:
  $P(X_{t+1} = j|X_t = i) = p_{ij}$ (i.e. independent of the position t)

$$Note\ that\ I\ no\ longer\ used\ the\ notation\ "L"\ for\ sequences$$

Type equation here.

The Markov property

- The p_{ij} are the elements of a matrix P called the one-step transition matrix of the chain.
- Stepping from one position to the next is one issue, how to start is another issue
  - An initial probability distribution is needed as well
  - It is determined by a vector of probabilities corresponding to every possible initial state value i: $\pi_i^{(0)} = \pi_i \ = P(X_0 = i)$
- The probability distribution for the states at position 1 can be obtained as follows:

$$P(X_1 = j) = \sum_{i \in \chi} P(X_0 = i, X_1 = j)$$

$$= \sum_{i \in \chi} P(X_0 = i)\, P(X_1 = j | X_0 = i) \ = \ \sum_{i \in \chi} \pi_i p_{ij}$$

The Markov property

- To compute the probability distribution for the states at position 2, we first show that $P(X_2 = j|X_0 = i)$ is the ij-th element of PP=P^2

$$P(X_2 = j|X_0 = i) = \sum_{k \in \chi} P(X_2 = j, X_1 = k|\, X_0 = i)$$
$$= \sum_{k \in \chi} P(X_2 = j|X_1 = k, X_0 = i) P(X_1 = k|\, X_0 = i)$$
$$= \sum_{k \in \chi} P(X_2 = j|X_1 = k) P(X_1 = k|\, X_0 = i)$$
$$= \sum_{k \in \chi} p_{ik} p_{kj} = (PP)_{ij}$$

- Therefore

$$\pi_j^{(2)} = P(X_2 = j) = \sum_{i \in \chi} \pi_i \, P_{ij}^2$$

The Markov property

- In a similar way it can be shown that

$$\pi_j^{(t)} = P(X_t = j) = \sum_{i \in \chi} \pi_i \, P_{ij}^t$$

- In principle, it can happen that the distribution $\pi^{(t)}$ is independent of t. This event is then referred to as a **stationary distribution** of the chain.
  - It occurs when $\sum_{i \in \chi} \pi_i p_{ij} = \pi_{j,}$ for all j, or stated differently when $\pi = \pi P$

Creating our own Markov chain simulation in practice

- Assume the observed dinucleotide relative frequencies (each row specifies a base and each column specifies the following base):

	A	C	G	T
A	0.146	0.052	0.058	0.089
C	0.063	0.029	0.010	0.056
G	0.050	0.030	0.028	0.051
T	0.087	0.047	0.063	0.140

- How to compute the individual base frequencies?
- How to propose initial state parameters to build a Markov chain?
- How to compute the transition matrix?

```
markov1 <- function(x,pi,P,n){
mg <- rep(0,n)
mg[1] <- sample(x,1,replace=TRUE,pi)
for (k in 1:(n-1)){
  mg[k+1] <- sample(x,1,replace=TRUE,P[mg[k],])
  }
return(mg)
}

x<-c(1:4)
pi <- c(0.342,0.158, 0.158, 0.342)
# A C G T one-step transition matrix:
P <- matrix(scan(),ncol=4,nrow=4,byrow=T)
0.423 0.151 0.168 0.258
0.399 0.184 0.063 0.354
0.314 0.189 0.176 0.321
0.258 0.138 0.187 0.415
```

```
tmp <- markov1(x,pi,P,50000)
A<- length(tmp[tmp[]==1])
C<- length(tmp[tmp[]==2])
G<- length(tmp[tmp[]==3])
T<- length(tmp[tmp[]==4])
(C+G)/(A+C+G+T)   # fraction of G+C

count <-0
for (i in 1:49999){
  if (tmp[i]==2 && tmp[i+1]==3)
  count <- count+1
  }
count/49999 # abundance of CG dinucleotide as estimated by the model
```

CpG islands

- The CG island is a short stretch of DNA in which the frequency of the CG sequence is higher than other regions. It is also called the CpG island, where "p" simply indicates that "C" and "G" are connected by a phosphodiester bond.
- CpG islands are often located around the promoters of housekeeping genes (which are essential for general cell functions) or other genes frequently expressed in a cell.
- At these locations, the CG sequence is not methylated.

(http://www.web-books.com/MoBio/Free/Ch7F2.htm)

5 Biological words of length 3

Introduction

- There are 61 codons that specify amino acids and three stop codons.
- Since there are 20 common amino acids, this means that most amino acids are specified by more than one codon.
- This has led to the use of a number of statistics to summarize the "bias" in codon usage.
- Since there is variation in codon frequencies, it is interesting to investigate these frequencies in more detail

Predicted relative frequencies

- For a sequence of independent bases L_1, L_2, ... , L_n the expected 3-tuple relative frequencies can be found by using the logic employed for dinucleotides we derived before
- The probability of a 3-word can be calculated as follows:

$$\mathbb{P}(L_i = r_1, L_{i+1} = r_2, L_{i+2} = r_3) = \mathbb{P}(L_i = r_1)\mathbb{P}(L_{i+1} = r_2)\mathbb{P}(L_{i+2} = r_3).$$

- This provides the expected frequencies of particular codons, using the individual base frequencies
- It follows that among those codons making up the amino acid Phe, the expected proportion of TTT is

$$\frac{\mathrm{P(TTT)}}{\mathrm{P(TTT)} + \mathrm{P(TTC)}}$$

The codon adaptation index

- Consider a sequence of amino acids X = x_1, x_2, ... , x_L representing protein X, with x_k representing the amino acid residue corresponding to codon *k* in the gene.
- Question: How does the actual codon usage compare with a model that states that the codons employed are the most probable codons for highly expressed genes?
- For the codon corresponding to a particular amino acid at position *k* in protein *X*, let p_k be the probability that *this* particular codon is used to code for the amino acid
- Let q_k correspond to the probability for *the most frequently used* codon of the corresponding amino acid in highly expressed genes.

The codon adaptation index (Sharp and Li 1987)

- The CAI is defined as

$$\mathrm{CAI} = \left[\prod_{k=1}^{L} p_k / q_k \right]^{1/L}$$

- It is the geometric mean of the ratios of the probabilities for the codons *actually* used to the probabilities of the codons *most frequently* used in highly expressed genes.
- An alternative way of writing this is

$$\log(\mathrm{CAI}) = \frac{1}{L} \sum_{k=1}^{L} \log(p_k / q_k).$$

- This expression is in terms of a sum of the logarithms of probability ratios.

The codon adaptation index

- The CAI for a gene sequence in genomic DNA provides a first approximation of its expression level:
  if the CAI is relatively large, then we would predict that the expression level is also large.
- Hence, the CAI can be shown to be correlated with mRNA levels

The codon adaptation index

- Comparison of predicted and observed triplet frequencies in coding sequences for a subset of genes and codons from *E. coli.* Figures in parentheses below each gene class show the number of genes in that class. Table 2.3 from Deonier et al 2005.

			Observed	
	Codon	Predicted	Gene Class I (502)	Gene Class II (191)
Phe	TTT	0.493	0.551	0.291
	TTC	0.507	0.449	0.709
Ala	GCT	0.246	0.145	0.275
	GCC	0.254	0.276	0.164
	GCA	0.246	0.196	0.240
	GCG	0.254	0.382	0.323
Asn	AAT	0.493	0.409	0.172
	AAC	0.507	0.591	0.828

Class II : Highly expressed genes

Class I : Moderately expressed genes

The codon adaptation index

- An example:

  Consider the amino acid sequence from the amino terminal end of the *himA* gene of E. coli (which codes for one of the two subunits of the protein IHF: length L = 99).

```
M   A   L   T   K   A   E   M   S   E   Y   L   F   ...
ATG GCG CTT ACA AAA GCT GAA ATG TCA GAA TAT CTG TTT ...
```

The codon adaptation index

- Top lines: amino acid sequence and corresponding codons.
- Upper table: probabilities for codons in lower table.
- The probability of the most frequently used codon in highly expressed genes is underlined (Fig 2.2 – Deonier et al 2005).

M	A	L	T	K	A	E	M	S	E	Y	L	F	...
ATG	GCG	CTT	ACA	AAA	GCT	GAA	ATG	TCA	GAA	TAT	CTG	TTT	...

1.000	0.469	0.018	0.451	0.798	0.469	0.794	1.000	0.428	0.794	0.193	0.018	0.228	
	0.057	0.018	0.468	0.202	0.057	0.206		0.319	0.206	0.807	0.018	0.772	
	0.275	0.038	0.035		0.275			0.033			0.038		
	0.199	0.033	0.046		0.199			0.007			0.033		
		0.007						0.037			0.007		
		0.888						0.176			0.888		

ATG	GCT	TTA	ACT	AAA	GCT	GAA	ATG	TCT	GAA	TAT	TTA	TTT	
	GCC	TTG	ACC	AAG	GCC	GAG		TCC	GAG	TAC	TTG	TTC	
	GCA	CTT	ACA		GCA			TCA			CTT		
	GCG	CTC	ACG		GCG			TCG			CTC		
		CTA						AGT			CTA		
		CTG						AGC			CTG		

The codon adaptation index

- The CAI for this fragment of coding sequence is given by

$$\mathrm{CAI} = \left[\frac{1.000}{1.000} \times \frac{0.199}{0.469} \times \frac{0.038}{0.888} \times \frac{0.035}{0.468} \cdots\right]^{1/99}.$$

- If every codon in a gene corresponded to the most frequently used codon in highly expressed genes, then the CAI would be 1.0.
- In E. coli a sample of 500 protein-coding genes displayed CAI values in the range from 0.2 to 0.85 (Whittam, 1996)

A new biological problem

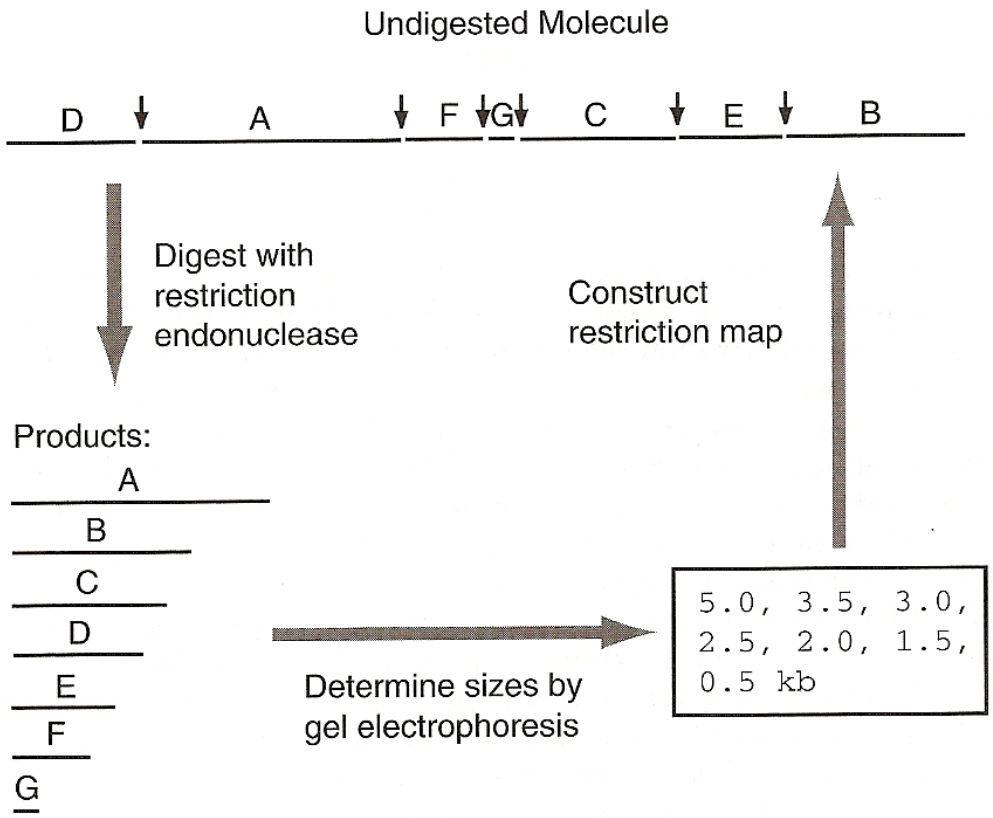
- Because DNA can be long but is very thin, it is easily broken during processing. Note that the DNA in human chromosome 1, at 245,000,000bp, is 8.33cm long and only 20 x10-^{8} cm thick
- Whereas restriction endonucleases provides the means for precisely and reproducibly cutting the DNA into fragments of manageable size (usually in the size range of 100s to 1000s of base pairs), and molecular cloning provides the method for amplifying the DNA of interest
- Note: Cloning puts DNA of manageable size into vectors that allow the inserted DNA to be amplified, and the reason for doing this is that large molecules cannot be readily manipulated without breakage

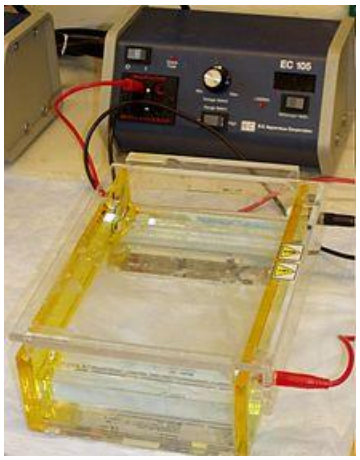
A new biological problem (continued)

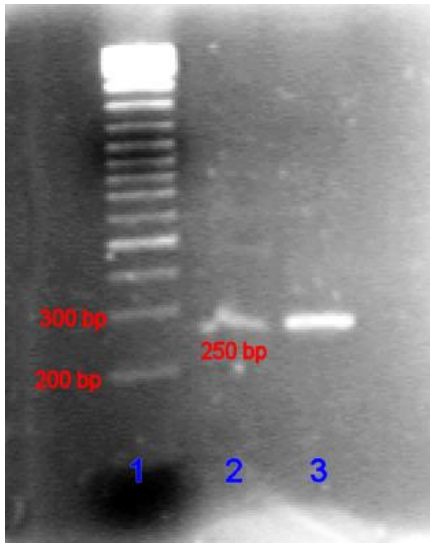
- A **restriction map** is a display of positions on a DNA molecule where cleavage by one or more restriction endonucleases can occur.
- It is created by determining the ordering of the DNA fragments generated after digestion with one or more restriction endonucleases.
- The restriction map is useful not only for dissecting a DNA segment for further analysis but also as a "fingerprint" or bar code that distinguishes that molecule from any other molecule.
- A graphical summary is given in the following figure (Figure 3.1 – Deonier et al 2005)

A new biological problem (continued)

- The order of fragments (D, A, F, G, C, E, B) is originally unknown. A variety of techniques may be employed to determine this order.







- DNA will migrate towards the positive electrode, which is usually colored red.
- Fragments of linear DNA migrate through agarose gels with a mobility that is inversely proportional to the log10 of their molecular weight.

(http://arbl.cvmbs.colostate.edu/hbooks/genetics/biotech/gels/agardna.html)

6 Modeling the number of restriction sites in DNA

Introduction

- Modelling the number of restriction sites in DNA is important when addressing the following questions:

If we were to digest the DNA with a restriction endonuclease such as EcoR1, 1) approximately how many fragments would be obtained, and 2) what would be their size distribution?

The number of restriction sites

- Restriction endonuclease recognition sequences have length *t* (4, 5, 6 or 8 typically), where *t* is much smaller than *n.*
- Our model assumes that cleavage can occur between any two successive positions on the DNA.
  - This is wrong in detail because, depending upon where cleavage occurs within the bases of the recognition sequence (which may differ from enzyme to enzyme), there are positions near the ends of the DNA that are excluded from cleavage.
  - However, since *t* is much smaller than *n,* the ends of the molecule do not affect the result too much

The number of restriction sites

- We again use X_i to represent the outcome of a trial occurring at position *i*, but this time X_i does not represent the identity of a base (one of four possible outcomes) but rather whether position i is or is not the beginning of a restriction site.
- In particular,

$$X_i = \begin{cases} 1, \text{ if base } i \text{ is the start of a restriction site,} \\ 0, \text{ if not.} \end{cases}$$

- We denote by *p* the probability that any position i is the beginning of a restriction site:

$$X_i = \begin{cases} 1, \text{ with probability } p, \\ 0, \text{ with probability } 1-p. \end{cases}$$

The number of restriction sites

- Unlike with tossing a fair coin, for the case of restriction sites on DNA, *p* depends upon
  - the base composition of the DNA and
  - the identity of the restriction endonuclease.
- For example:
  - Suppose that the restriction endonuclease is *Eco*RI, with recognition sequence 5'-GAATTC-3'.
    - The site really recognized is duplex DNA, with the sequence of the other strand determined by the Watson-Crick base-pairing rules.
  - Suppose furthermore that the DNA has equal proportions of A, C, G, and T.

The number of restriction sites

- The probability that any position is the beginning of a site is the probability that this first position is G, the next one is A, the next one is A, the next one is T, the next one is T, and the last one is C.
- Since, by the iid model, the identity of a letter at any position is independent of the identity of letters at any other position, we see from the multiplication rule that

$$p = \mathbb{P}(\mathtt{GAATTC}) = \mathbb{P}(\mathtt{G})\mathbb{P}(\mathtt{A})\mathbb{P}(\mathtt{A})\mathbb{P}(\mathtt{T})\mathbb{P}(\mathtt{T})\mathbb{P}(\mathtt{C}) = (0.25)^6 \sim 0.00024.$$

- Notice that *p* is small, a fact that becomes important later.

The number of restriction sites

- The appearance of restriction sites along the molecule is represented by the string $X_1, X_2, \ldots, X_n$,
- The number of restriction sites is $N = X_1 + X_2 + \ldots + X_m$, where $m = n - 5$.
    - The sum has m terms in it because a restriction site of length 6 cannot begin in the last five positions of the sequence, as there aren't enough bases to fit it in.
- For simplicity of exposition we take m = *n* in what follows.
- What really interests us is the number of "successes" (restriction sites) in *n* trials.

The number of restriction sites

- If $X_1, X_2, \ldots, X_n$ were independent of one another, then the probability distribution of *N* would be a binomial distribution with parameters n and *p;*
  - The expected number of sites would therefore be *np*
  - The variance would be *np(*1 - *p).*
- We remark that despite the X_i are not in fact independent of one another (because of overlaps in the patterns corresponding to X_i and X_{i+1}, for example), the binomial approximation usually works well.
- Computing probabilities of events can be cumbersome when using the probability distribution

$$P(N = j) = \binom{n}{j} p^j (1-p)^{n-j}, \text{ j = 0,1, ...,n}$$

Poisson approximation to the binomial distribution

- In what follows, we assume that *n* is large and *p* is small, and we set λ= *np.*
- We know that for *j* = 0, 1, ... , *n,*

$$P(N = j) = \binom{n}{j} p^j (1-p)^{n-j}$$

- Writing

$$\mathbb{P}(N = j) = \frac{n(n-1)(n-2)\cdots(n-j+1)}{j!(1-p)^j} p^j (1-p)^n.$$

and given that the number of restriction sites (j) is small compared to the length of the molecule (n), such that

$$n(n-1)(n-2)\ldots(n-j+1) \approx n^j, (1-p)^j \approx 1,$$

Poisson approximation to the binomial distribution

$$\mathbb{P}(N = j) \approx \frac{(np)^j}{j!}(1-p)^n = \frac{\lambda^j}{j!}\left(1 - \frac{\lambda}{n}\right)^n.$$

in which $\lambda = np$.

- From calculus, for any x,

$$\lim_{n\to\infty}\left(1 - \frac{x}{n}\right)^n = \mathrm{e}^{-x}.$$

- Since *n* is large (often more than 10^4), we replace $(1 - \frac{\lambda}{n})^n$ by $e^{-\lambda}$ to get our final approximation in the form

$$\mathbb{P}(N = j) \approx \frac{\lambda^j}{j!}\,\mathrm{e}^{-\lambda}, \quad j = 0, 1, 2, \ldots$$

- This is the formula for the Poisson distribution with parameter $\lambda = np$

Poisson approximation to the binomial distribution

- Example:
  - To show how this approximation can be used, we estimate the probability that there are no more than two *Eco*RI sites in a DNA molecule of length 10,000, assuming equal base frequencies
  - Earlier we obtained p=0.00024 for this setting.
  - The problem is to compute $P(N \leq 2)$
    - Therefore $\lambda = np = 2.4$
    - Using the Poisson distribution: $P(N \leq 2) \approx 0.570$
    - Interpretation: More than half the time, molecules of length 10,000 and uniform base frequencies will be cut by *Eco*RI two times or less
- R code:

```
ppois(2,2.4)
```

Distribution of restriction fragment lengths

- With this generalization, we assume that restriction sites now occur according to a Poisson process with rate λ per bp. Then the probability of *k* sites in an interval of length *l* bp is

$$\mathbb{P}(N = k) = \frac{\mathrm{e}^{-\lambda l}(\lambda l)^k}{k!}, \ k = 0, 1, 2, \ldots \ .$$

- We can also calculate the probability that a restriction fragment length X is larger than x. If there is a site at y, then the length of that fragment is greater than x if there are no events in the interval (y, y + x):

$$\mathbb{P}(X > x) = \mathbb{P}(\text{no events in } (y, y + x)) = \mathrm{e}^{-\lambda x}, \ x > 0.$$

Distribution of restriction fragment lengths

- The previous has some important consequences:

$$\mathbb{P}(X \leq x) = \int_0^x f(y)\mathrm{d}y = 1 - \mathrm{e}^{-\lambda x},$$

so that the density function for X is given by

$$f(x) = \lambda \mathrm{e}^{-\lambda x},\ x > 0.$$

- The distance between restriction sites therefore follows an exponential distribution with parameter λ
  - The mean distance between restriction sites is $1/\lambda$

Simulating restriction fragment lengths (sizes)

- If we *simulated* a sequence using the iid model, we could compute the fragment sizes in this simulated sequence and visualize the result
- R code simulating a DNA sequence having 48500 positions and uniform base probabilities:

```
x<-c(1:4)
propn <- c(0.25,0.25,0.25,0.25)
seq2 <- sample(x,48500,replace=TRUE,prob=propn)
seq2[1:15]
length(seq2[])
```

Simulating restriction fragment lengths

- R code identifying the restriction sites in a sequence string, with bases coded numerically:

```
rsite <- function(inseq, seq){
 # inseq: vector containing input DNA sequence,
 # A=1, C=2, G=3, T=4
 # seq: vector for the restriction site, length m
 # Make/initialize vector to hold site positions found in inseq
 xxx <- rep(0,length(inseq))
 m <-length(seq)
 # To record whether position of inseq matches seq
 truth <- rep(0,m)
```

```
# Check each position to see if a site starts there
  for (i in 1:(length(inseq) - (length(seq) -1))){
    for (j in 1:m){
      if (inseq[i+j-1]==seq[j]){
      truth[j] <- 1 # Record match to jth position
      }
    }
    if (sum(truth[]) ==m){ # Check whether all positions match
    xxx[i] <- i          # Record site if all positions match
    }
    truth <- rep(0,m)    # Reinitialize for next loop cycle
  }
  # Write vector of restriction sites positions stored in xxx
  L <- xxx[xxx>0]
  return(L)
}
```

Simulating restriction fragment lengths

- The restriction sites we look for are for A*lu*I, AGCT.
- R code envoking the appropriate function:

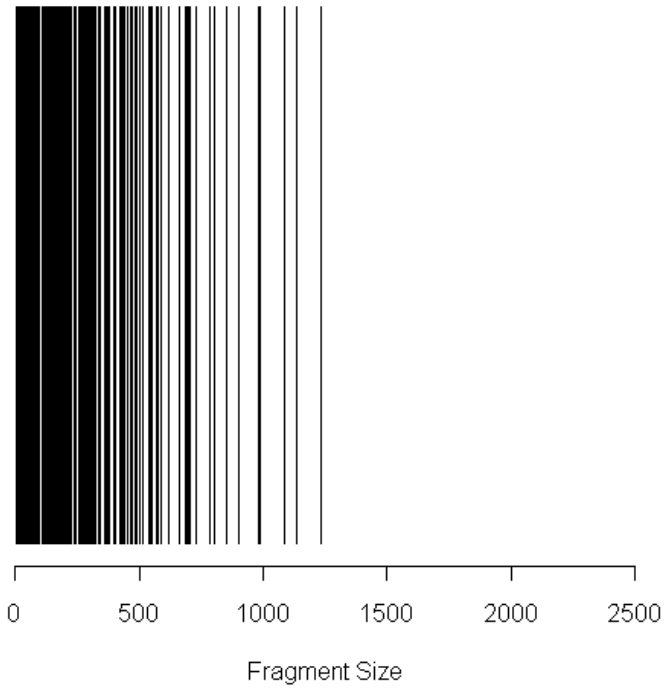
```
alu1 <- c(1,3,2,4)
alu.map <- rsite(seq2,alu1)
length(alu.map)
alu.map[1:10]
```

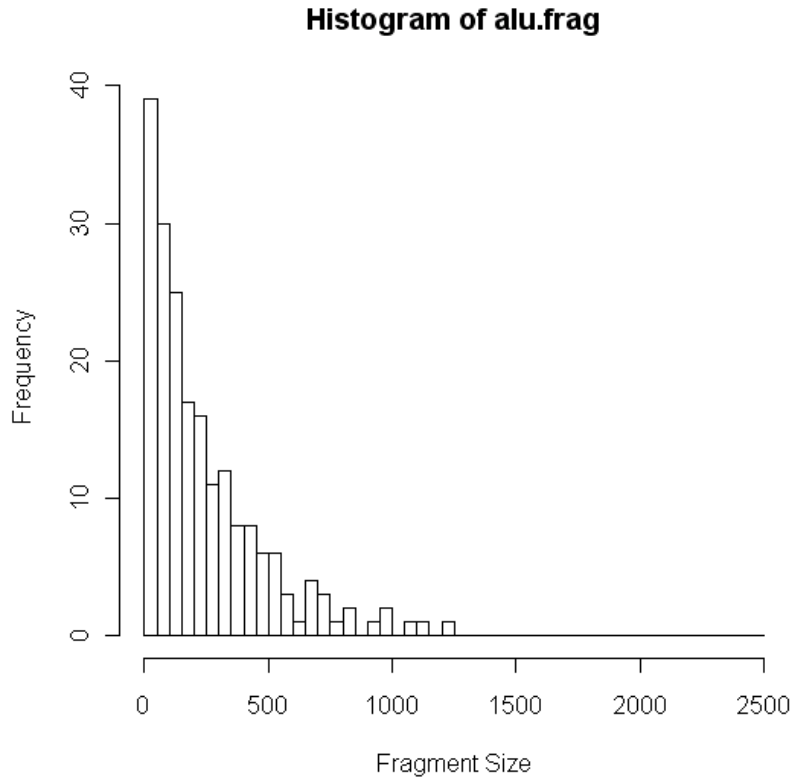
Simulating restriction fragment lengths

- The fragment lengths can be obtained by subtracting positions of successive sites
- R code doing it for you:

```
flengthr <- function(rmap,N){
  # rmap is a vector of restriction sites for a linear molecule
  # N is the length of the molecule
  frags <- rep(0,length(rmap))
  # Vector for substraction results: elements initialized to 0
  rmap <-c(rmap,N)
  # Adds length of molecule for calculation of end piece
  for(i in 1:(length(rmap)-1)){
  frags[i] <- rmap[i+1]-rmap[i]
  }
  frags <- c(rmap[1],frags) # First term is left end piece
  return(frags)
}
```

Simulating restriction fragment lengths



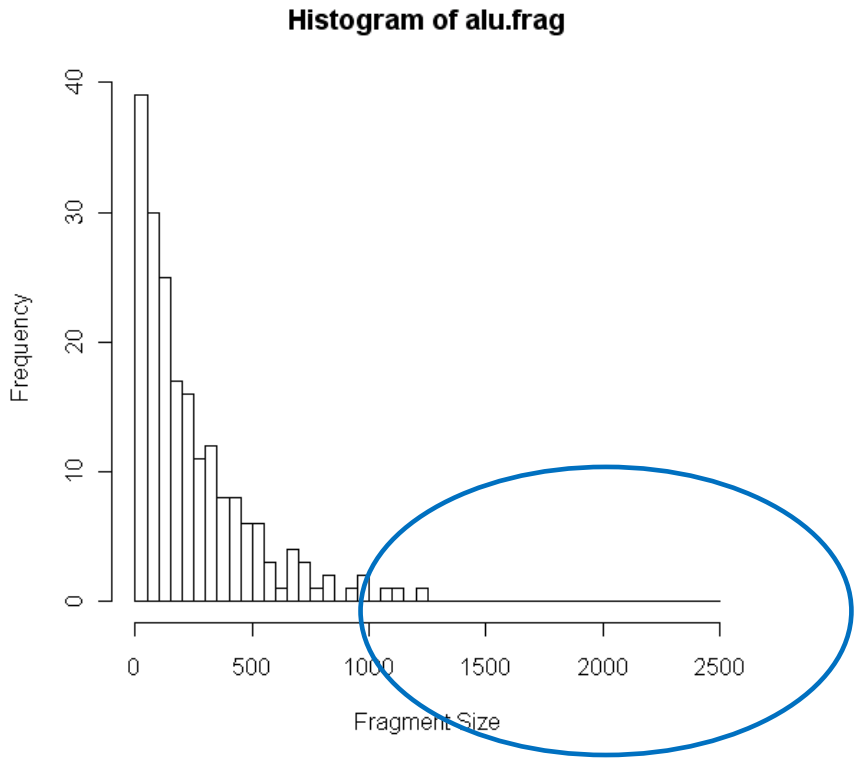


Simulating restriction fragment lengths

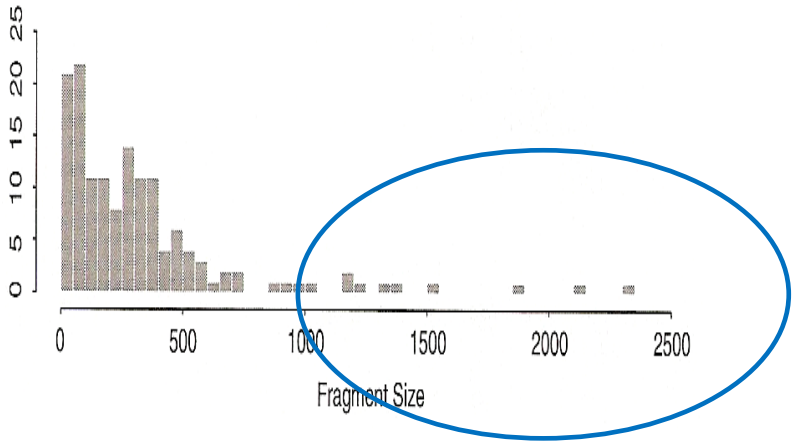
- R code corresponding to the figures:

```
plot(c(0,2500),c(3,1),xlab="Fragment Size",ylab="",type="n",axes=F)
axis(1,c(0,500,1000,1500,2000,2500))
for (i in 1:length(alu.frag)){
lines(c(alu.frag[i],alu.frag[i]),c(1,3))
}
hist(alu.frag,breaks=seq(0,2500,50), freq = TRUE,xlab="Fragment Size")
```

Is our theoretical model to simulate restriction fragment lengths valid?



Histogram based on theoretical model



Histogram of fragment sizes (bp) produced by AluI digestion of bacteriophage lambda DNA

Simulating restriction fragment lengths

- To determine whether the actual distribution differs significantly from the mathematical model (exponential distribution), we could break up the length axis into a series of "bins" and calculate the expected number of fragments in each bin by using the model-based (theoretical) density.
- We could then compare the observed with expected number of fragments (using the same bin boundaries) via for instance a χ^2 – test.

7 Comparing sequences

Introduction

- Sequence comparisons are important for a number of reasons.
  - First, they can be used to establish evolutionary relationships among organisms using methods analogous to those employed for anatomical characters.
  - Second, comparison may allow identification of functionally conserved sequences (e.g., DNA sequences controlling gene expression).
  - Finally, comparisons between humans and other species may identify corresponding genes in model organisms, which can be genetically manipulated to develop models for human diseases.

Evolutionary basis of alignment

- This lies in the fact that sequence alignment enables the researcher to determine if two sequences display sufficient similarity to justify the inference of homology.
- Understanding the difference between similarity and homology is of utmost importance:
  - Similarity is an observable quantity that may be expressed as a % identity or some other measure.
  - Homology is a conclusion drawn from the data that the two genes share a common evolutionary history.

(S-star Subbiah)

Evolutionary basis of alignment

- Genes are either homologous or not homologous.
- There are no degrees of homology as are there in similarity.
- While it is presumed that the homologous sequences have diverged from a common ancestral sequence through iterative molecular changes we do not actually know what the ancestral sequence was.

Homology versus similarity

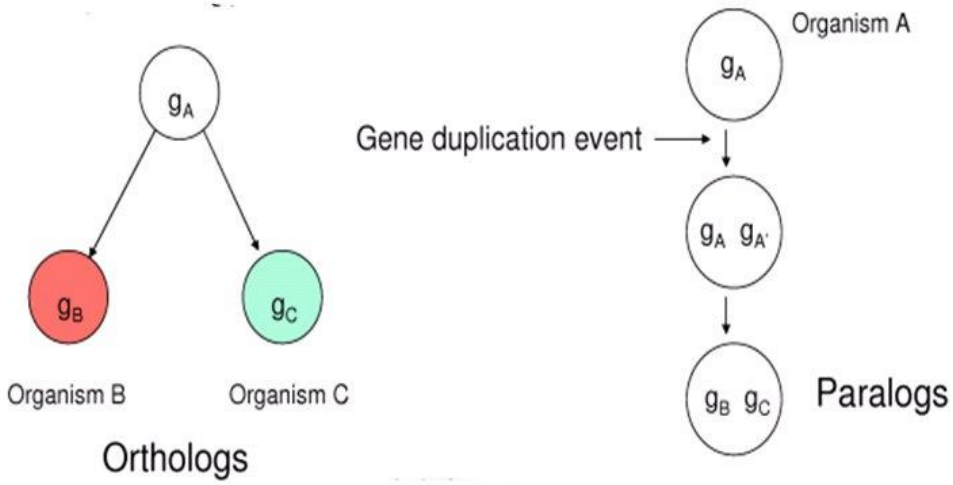
- Hence, sequence similarity is not sequence homology and can occur by chance ...
- If two sequences have accumulated enough mutations over time, then the similarity between them is likely to be low.
- Consequently, homology is more difficult to detect over greater evolutionary distances

#mutations	
0	agt gt ccgt t aagt gcgt t c
1	agt gt ccgt t at agt gcgt t c
2	agt gt ccgct t at agt gcgt t c
4	agt gt ccgct t aagggcgt t c
8	agt gt ccgct t caaggggcgt
16	gggccgt t cat gggggt
32	gcagggcgt cact gagggct

#mutations	
64	acagt ccgt t cgggct at t g
128	cagagcact accgc
256	cacgagt aagat at agct
512	t aat cgt gat a
1024	accct t at ct act t cct ggagt t
2048	agcgacct gcccaa
4096	caaac

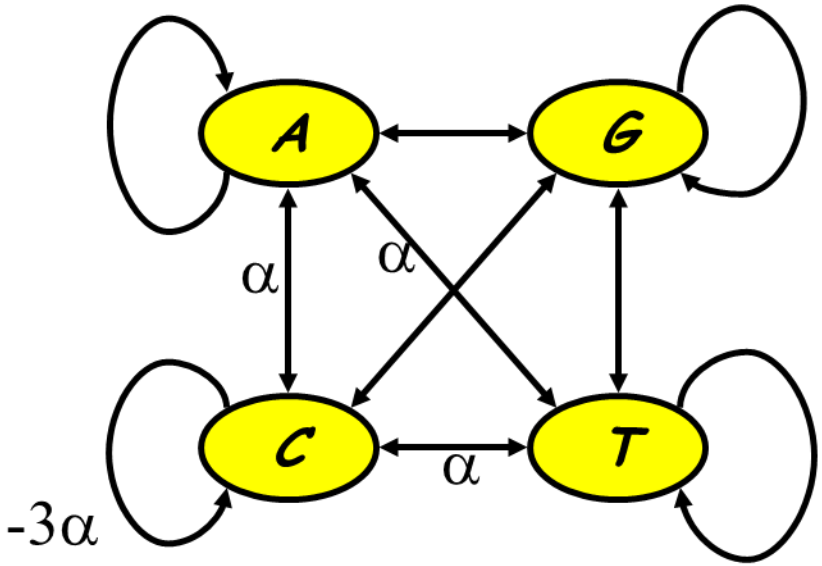
Orthologs and paralogs

- We distinguish between two types of homology
  - Orthologs: homologs from two different species, separated by a, what is called, a *speciation event.* Orthologs typically retain the original function
  - Paralogs: homologs within a species, separated by a *gene duplication event*. A copy is free to mutate and acquire new function



The Jukes-Cantor model (1969)

- We need to develop a formula for DNA evolution via Prob(Y | x, t) where x and y are taken from {A, C, G, T} and t is the time length.
- Jukes-Cantor assumes equal rate of change:



$$R = \begin{array}{c} \\ A \\ C \\ G \\ T \end{array} \begin{array}{c} \begin{array}{cccc} A & C & G & T \end{array} \\ \begin{bmatrix} -3\alpha & \alpha & \alpha & \alpha \\ \alpha & -3\alpha & \alpha & \alpha \\ \alpha & \alpha & -3\alpha & \alpha \\ \alpha & \alpha & \alpha & -3\alpha \end{bmatrix} \end{array}$$

8 Pairwise alignments

Introduction

- There are many different ways that two strings might be aligned. Ordinarily, we expect homologs to have more matches than two randomly chosen sequences.
- Example (matches are indicated by . and - is placed opposite bases not aligned):

```
ACGTCTAG    2 matches
..          5 mismatches
ACTCTAG-    1 not aligned
```

Introduction

- We might instead have written the sequences

```
ACGTCTAG     5 matches
  .....      2 mismatches
-ACTCTAG     1 not aligned
```

- We might also have written

```
ACGTCTAG     7 matches
..  .....    0 mismatches
AC-TCTAG     1 not aligned
```

Which alignment is better?
What does better mean?

Introduction

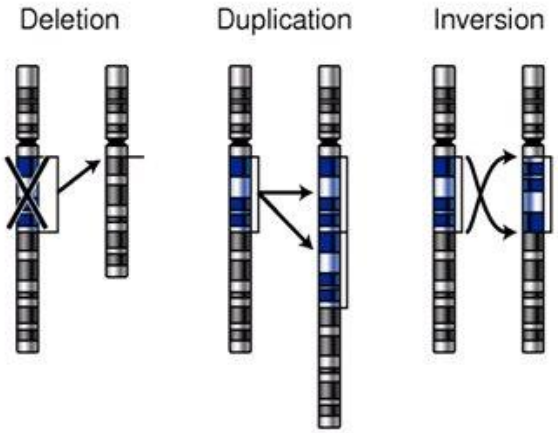
- Next, consider aligning the sequence TCTAG with a long DNA sequence:

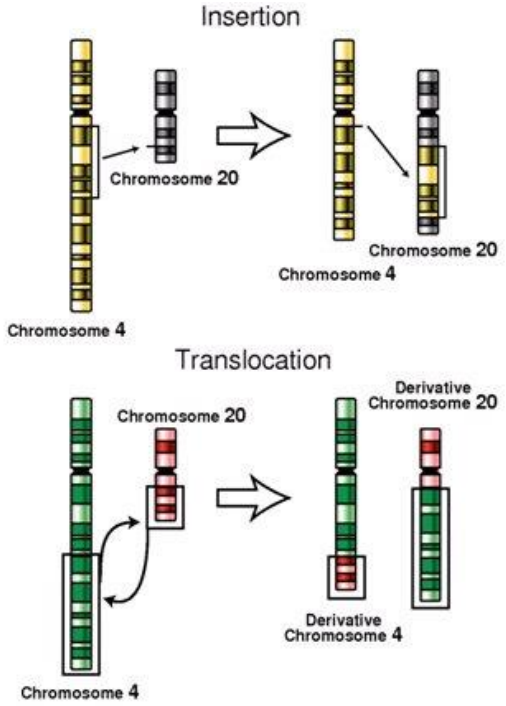
```
...AACTGAGTTTACGCTCATAGA...
      T---CT-A--G
```

- We might suspect that if we compared any string of modest length with another very long string, we could obtain perfect agreement if we were allowed the option of "not aligning" with a sufficient number of letters in the long string.
- Events such as point mutations, insertion or deletion of short segments will affect alignment efforts.

Recall – Chapter 1

Different types of mutations





NIH – National Human Genome Research Institute)

Toy example

- Suppose that we wish to align WHAT with WHY.
- Our goal is to find the highest-scoring alignment. This means that we will have to devise a scoring system to characterize each possible alignment.
- One possible alignment solution is

```
WHAT
WH-Y
```

- However, we need a rule to tell us how to calculate an alignment score that will, in turn, allow us to identify which alignment is best.
- Let's use the following scores for each instance of match, mismatch, or indel:
  - identity (match) +1
  - substitution (mismatch) -μ
  - indel - δ

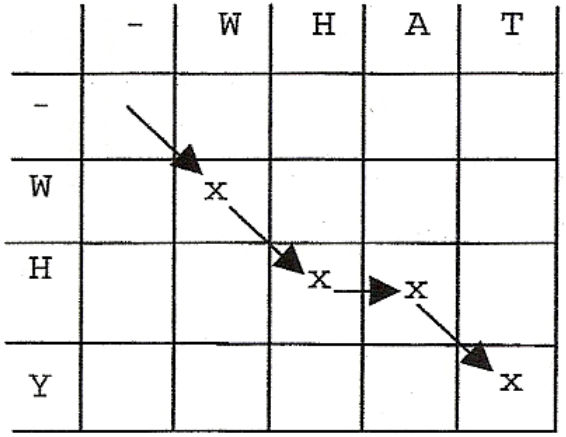
Toy example

- The minus signs for substitutions and indels assure that alignments with many substitutions or indels will have low scores.
- We can then define the score S as the sum of individual scores at each position:

$$S(\text{WHAT/WH} - \text{Y}) = 1 + 1 - \delta - \mu$$

- There is a more general way of describing the scoring process (not necessary for "toy" problems such as the one above).
- In particular: write the target sequence (the sequence we want to compare; WHY) and the search space as rows and columns of a matrix:

Toy example



- We have placed an x in the matrix elements corresponding to a particular alignment
- We have included one additional row and one additional column for initial indels (-) to allow for the possibility (not applicable here) that alignments do not start at the initial letters (W opposite W in this case).

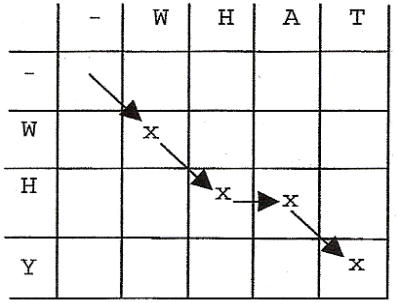
Toy example

- We can indicate the alignment

  WHAT

  WH-Y

  as a path through elements of the matrix (arrows).



- If the sequences being compared were identical, then this path would be along the diagonal.

Toy example

- Other alignments of WHAT with WHY would correspond to paths through the matrix other than the one shown.
- Each step from one matrix element to another corresponds to the incremental shift in position along one or both strings being aligned with each other, and we could write down in each matrix element the running score up to that point instead of inserting x.

	-	W	H	A	T
-	0				
W		1			
H			2	2-δ	
Y					2-δ-μ

Toy example

- What we seek is the path through the matrix that produces the greatest possible score in the element at the lower right-hand corner.
- That is our "destination," and it corresponds to having used up all of the letters in the target string (first column) and search space (first row)-this is the meaning of *global alignment*.
- Using a scoring matrix such as this employs a particular trick of thinking.

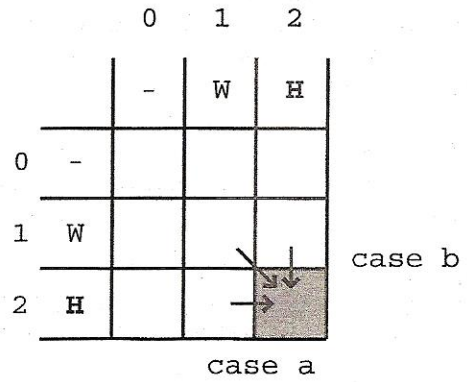
Dynamic programming

- Dynamic programming (DP); a computer algorithmic technique invented in the 1940's,with applications to many types of problems.
- Key properties of problems solvable with DP include that the optimal solution typically contains optimal solutions to subproblems, and only a "small" number of subproblems are needed for the optimal solution.

(T.H. Cormen et al., Introduction to Algorithms, McGraw-Hill 1990).

Toy example (continued)

- The best alignment is revealed by beginning at the destination (lower right-hand corner matrix element) and working backward, identifying the path that maximizes the score at the end.



- To do this, we will have to calculate scores for all possible paths into each matrix element from its neighbouring elements above, to the left, and diagonally above.

Toy example

- There are three possible paths into element (3, 3) (aligning left to right with respect to both strings; letters not yet aligned are written in parentheses):

  Case a. 

  If we had aligned WH in WHY with W in WHAT (corresponding to element (2, 1)), adding H in WHAT without aligning it to H in WHY corresponds to an insertion of H (relative to WHY) and advances the alignment from element (2, 1) to element (2,2) (horizontal arrow):

  (W) H (AT)
  (WH) - (Y)

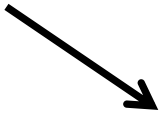
Toy example

Case b. 

If we had aligned W in WHY with WH in WHAT (corresponding to element (1, 2)), adding the H in WHY without aligning it to H in WHAT corresponds to insertion of H (relative to WHAT) and advances the alignment from element (1, 2) to element (2, 2) (vertical arrow):

(WH)-(AT)
(W)H (Y)

Toy example

Case c. 

If we had aligned W in WHY with W in WHAT (corresponding to element (1,1)), then we could advance to the next letter in both strings, advancing the alignment from (1,1) to (2,2) (diagonal arrow above):

(W)H(AT)
(W)H (Y)

- Note that horizontal arrows correspond to adding indels to the string written vertically and that vertical arrows correspond to adding indels to the string written horizontally.

Toy example

- Associated with each matrix element (x, y) from which we could have come into (2,2) is the score $S_{x,y}$ up to that point.
- Suppose that we assigned scores based on the following scoring rules:

identity (match)	+1
substitution (mismatch)	-1
indel	-2

- Then the scores for the three different routes into (2,2) are

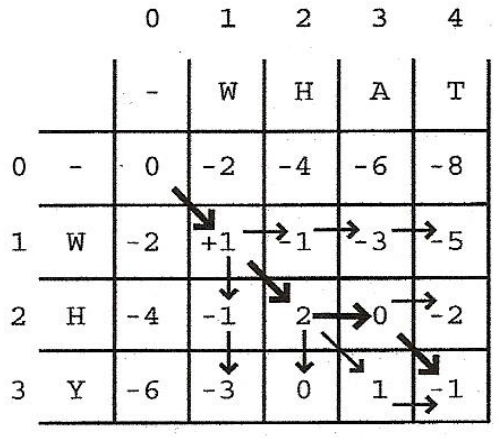
$$\begin{aligned} \text{Case a}: &\quad S_{2,2} = S_{2,1} - 2, \\ \text{Case b}: &\quad S_{2,2} = S_{1,2} - 2, \\ \text{Case c}: &\quad S_{2,2} = S_{1,1} + 1. \end{aligned}$$

Toy example

- The path of the cases a, b, or c that yields the highest score for $S_{2,2}$ is the preferred one, telling us which of the alignment steps is best.
- Using this procedure, we will now go back to our original alignment matrix and fill in all of the scores for all of the elements, keeping track of the path into each element that yielded the maximum score to that element.
- The initial row and column labeled by (-) corresponds to sliding WHAT or WHY incrementally to the left of the other string without aligning against any letter of the other string.
- Since penalties for indels are -2, the successive elements to the right or down from element (0, 0) each are incremented by -2 compared with the previous one.
- Aligning (-) opposite (-) contributes nothing to the alignment of the strings, so element (0,0) is assigned a score of zero.

Toy example

- The remaining elements of the matrix are filled in by the same procedure, with the following result:



- The final score for the alignment is $S_{3,4}$ = -1.
- We just completed a **global alignment** exercise
- Global alignment efforts are not unique

Toy example

- The path through element (2,3) (upper path, bold arrows) corresponds to the alignment

  WHAT
  WH-Y

  which is read by tracing back through all of the elements visited in that path.
- The lower path (through element (3, 3)) corresponds to the alignment

  WHAT
  WHY-
- Each of these alignments is equally good (two matches, one mismatch, one indel).

Local alignment: Rationale

- With only partial sequence similarity and sequences of very different lengths, attempts at global alignment of two sequences can lead to huge cumulative indel penalties.
- What we need is a method to produce the best *local alignment;* that is, an alignment of segments contained *within* two strings (Smith and Waterman, 1981).
- As before, we will need an alignment matrix, and we will seek a high-scoring path through the matrix.
- Unlike before, the path will traverse only *part* of the matrix. Also, we do not apply indel penalties if strings *A* and *B* fail to align at the ends.

Local alignment: Rationale

- Hence, instead of having elements $-i\delta$ and $-j\delta$ in the first row and first column, respectively ($-\delta$ being the penalty for each indel), all the elements in the first row and first column will now be zero.
- Moreover, since we are interested in paths that yield high scores over stretches less than or equal to the length of the smallest string, there is no need to continue paths whose scores become too small.
  - If the best path to an element from its immediate neighbors above and to the left (including the diagonal) leads to a negative score, we will arbitrarily assign a 0 score to that element.
  - We will identify the best local alignment by tracing back from the matrix element having the highest score. This is usually not (but occasionally may be) the element in the lower right-hand corner of the matrix.

Mathematical formulation for local alignments

- We are given two strings $A = a_1a_2a_3 \ldots a_n$ and $B = b_1b_2b_3 \ldots b_m$
- *I* and *J* are *intervals* of *A* and B, respectively. We indicate this by writing $I \subset A$ and $J \subset B$
- The best local alignment score, M(A, *B),* for strings *A* and *B* is

$$M(A, B) = \max\{S(I, J) : I \subset A, J \subset B\}$$

where *S*(*I*, *J*) is the score for subsequences *I* and *J* and *S*(Ø,Ø) = 0.

Mathematical formulation for local alignments

- $M_{i,0} = M_{0,i} = 0$
- The score up to and including the matrix element $M_{i,j}$ is calculated by using scores for the elements immediately above and to the left (including the diagonal), but this time scores that fall below zero will be replaced by zero.
- The scoring for matches, mismatches, and indels is otherwise the same as for global alignment.
- The resulting expression for scoring $M_{i,j}$ is

$$M_{i,j} = \max \begin{Bmatrix} M_{i-1,j-1} + s(a_i, b_i) \\ M_{i-1,j} - \delta \\ M_{i,j-1} - \delta \\ 0 \end{Bmatrix}.$$

Mathematical formulation

- The best local alignment is the one that ends in the matrix element having the highest score:

$$\max\{S(I,J) : I \subset A, J \subset B\} = \max_{i,j} M_{i,j}.$$

- Thus, the best local alignment score for strings *A* and *B* is

$$M(A,B) = \max_{i,j} M_{i,j}.$$

Exercise

- Determine the best local alignment and the maximum alignment score for

A=ACCTAAGG and B=GGCTCAATCA

- What do we need?
  - Definition for $s(a_i, b_j)$
  - Definition for delta; indel penalties $s(a_i, -)$ and $s(-, b_j)$
- For scoring, take
  - $s(a_i, b_j) = +2$ if $a_i = b_j$,
  - $s(a_i, b_j) = -1$ if $a_i \neq b_j$ and
  - $s(a_i, -) = s(-, b_j) = -2$

Scoring rules can be quite diverse

- Studies of mutations in homologous genes have indicated that transition mutations (A $\rightarrow$ G, G $\rightarrow$ A, C $\rightarrow$ T, or T $\rightarrow$ C) occur approximately twice as frequently as do transversions (A $\rightarrow$ T, T $\rightarrow$ A, A $\rightarrow$ C, G $\rightarrow$ T, etc.). (A, G are purines; T, C are larger molecules called pyrimidines)
- Therefore, it may make sense to apply a lower penalty for transitions than for transversions
- The collection of $s(a_i , b_j)$ values in that case might be represented as

a_i: \ b_j:	A	C	G	T
A	1	-1	-0.5	-1
C	-1	1	-1	-0.5
G	-0.5	-1	1	-1
T	-1	-0.5	-1	1

Scoring rules can be quite diverse

- Indels are not necessarily independent.
- Up to now, we have scored a gap of length *k* as

$$\omega(k) = -k\delta$$

- However, insertions and deletions sometimes appear in "chunks" as a result of biochemical processes such as replication slippage at microsatellite repeats.
- Also, deletions of one or two nucleotides in protein-coding regions would produce frameshift mutations (usually non-functional), but natural selection might allow small deletions that are integral multiples of 3, which would preserve the reading frame and some degree of function.

Scoring rules

- The aforementioned examples suggest that it would be better to have gap penalties that are not simply multiples of the number of indels.
- One approach is to use an expression such as

$$\omega(k) = -\alpha-\beta(k-1)$$

- This would allow us to impose a larger penalty for opening a gap *(-α)* and a smaller penalty for gap extension (-β for each additional base in the gap).

Towards automated alignments

- Needleman and Wunsch (1970) were the first to introduce a heuristic alignment algorithm for calculating homology between sequences (global alignment).
- Later, a number of variations have been suggested, among others Sellers (1974) getting closer to fulfill the requests of biology by measuring the metric distance between sequences [Smith and Waterman, 1981].
- Further development of this led to the **Smith-Waterman algorithm** based on calculation of local alignments instead of global alignments of the sequences and allowing a consideration of deletions and insertions of arbitrary length.

The Smith-Waterman algorithm

- The Smith-Waterman algorithm uses individual pair-wise comparisons between characters as:

$$M_{i,j} = \max \begin{Bmatrix} M_{i-1,j-1} + s(a_i, b_i) \\ M_{i-1,j} - \delta \\ M_{i,j-1} - \delta \\ 0 \end{Bmatrix}.$$

Do you recognize this formula?

- The result of a Smith-Waterman algorithm search will only return one result for each pair of compared sequences: the optimal alignment

The Smith-Waterman algorithm

- Through the Japanese Institute of Bioinformatics Research and Development (BIRD) a public available software version of Smith-Waterman, SSEARCH, is accessible: http://www-btls.jst.go.jp/cgi-bin/Tools/SSEARCH/index.cgi.
- There are also commercial software packages available which perform Smith-Waterman searches.

- The Smith-Waterman algorithm is the most accurate algorithm when it comes to search databases for sequence homology but it is also the most time consuming, thus there has been a lot of development and suggestions for optimizations and less time-consuming models. One such models is used by BLAST [Shpaer et al., 1996].
- http://blast.ncbi.nlm.nih.gov/Blast.cgi?CMD=Web&PAGE_TYPE=BlastDocs&DOC_TYPE=Download

BLAST rationale

In particular, three steps are involved in BLAST:

- Step 1: Find local alignments between the query sequence and a data base sequence (“seed hits”)
- Step 2: Extend the seed hits into high-scoring local alignments
- Step 3: Calculate p-values and a rank ordering of the local alignments

Anatomy of BLAST: Finding Local Matches

- At the heart (step 1) lies defining “neighborhood sequences”.
- Consider the following query sequence I and search space J:

J = C C A T C G C C A T C G
I = G C A T C G G C

- We use subsequences of length k = 5. For the neighborhood size, we use all 1-mismatch sequences, which would result from scoring matches 1, mismatches 0, and the test value (threshold) T = 4.
- For sequences of length k = 5 in the neighborhood of GCATC with T = 4 (excluding exact matches), we have:

$$\begin{Bmatrix} A \\ C \\ T \end{Bmatrix} CATC, \quad G \begin{Bmatrix} A \\ G \\ T \end{Bmatrix} ATC, \quad GC \begin{Bmatrix} C \\ G \\ T \end{Bmatrix} TC, \quad GCA \begin{Bmatrix} A \\ C \\ G \end{Bmatrix} C, \quad GCAT \begin{Bmatrix} A \\ G \\ T \end{Bmatrix}$$

Anatomy of BLAST: Finding Local Matches

$$\left\{\begin{matrix}A\\C\\T\end{matrix}\right\}CATC,\quad G\left\{\begin{matrix}A\\G\\T\end{matrix}\right\}ATC,\quad GC\left\{\begin{matrix}C\\G\\T\end{matrix}\right\}TC,\quad GCA\left\{\begin{matrix}A\\C\\G\end{matrix}\right\}C,\quad GCAT\left\{\begin{matrix}A\\G\\T\end{matrix}\right\}$$

- Each of these terms represents three sequences, so that in total there are 1 + (3 x 5) = 16 exact matches to search for in J.
- For the three other 5-word patterns in I (CATCG, ATCGG, and TCGGC), there are also 16 exact 5-words, for a total of 4 x 16 = 64 5-word patterns to locate in J.
- A hit is defined as an instance in the search space (database) of a k-word match, within threshold T, of a k-word in the query sequence.

Anatomy of BLAST: Finding Local Matches

- In our example, there are several hits in I to sequence J. They are

5-words in I	5-words in J	J position(s)	Score
CATCG	CATCG	2, 8	5
GCATC	CCATC	1	4
ATCGG	ATCGC	3	4
TCGGC	TCGCC	4	4

- Observe that the hits correspond to only a tiny fraction of the entire search space.

Anatomy of BLAST: Finding Local Matches

- The next step (step 2) is to extend the alignment starting from these "seed" hits.
- In particular, early versions of BLAST involved:
  - Starting from any seed hit, extend by including successive positions, with corresponding increments to the alignment score.
  - Continue until the alignment score falls below the maximum score attained up to that point by a specified amount.
- With the original version of BLAST, over 90% of the computation time was employed in producing the un-gapped extensions from the hits.
- Later versions of BLAST have reduced the time required for the un-gapped extensions considerably.
- Even with the additional capabilities for allowing gaps in the alignment, the newer versions of BLAST run about three times faster than the original version (Altschul et aL, 1997).

Anatomy of BLAST: Scores

- Another aspect of a BLAST analysis is to rank-order by p-values the sequences found (step 3).
- If the database is D and a sequence X scores S(D,X) = s against the database, the p-value is $P(S(D,Y) \geq s)$, where Y is a random sequence.
- The smaller the p-value, the greater the "surprise" and hence the greater the belief that something real has been discovered.
- A p-value of 0.1 means that with a collection of query sequences picked at random, in 1/10 of the instances a score that is as large or larger would be discovered.
- A p-value of 10^{-6} means that only once in a million instances would a score of that size appear by chance alone.
- There is a nice way of computing BLAST p-values that has a solid mathematical basis.

Anatomy of BLAST: p-value derivation – intuitive approach

- In our n x m alignment matrix, there are (approximately) n x m places to begin an alignment.
- With

$$p = \mathbb{P}(\text{two random letters are equal}).$$

  the mean or expected number of local alignments of at least length t is $nm(1 - p)p^t$.
- Obviously, we want this to be a rare event that is well-modelled by the Poisson distribution with mean:

$$\lambda = nm(1-p)p^t$$

- Recall: For Poisson distribution, we need to use the function of j: $\frac{\lambda^j}{j!}e^{-\lambda}$ (the expected nr of occurrences is lambda)

Anatomy of BLAST: p-value derivation – intuitive approach

$$\begin{aligned}\mathbb{P}(\text{there is local alignment } t \text{ or longer}) &\approx 1 - \mathbb{P}(\text{no such event}) \\ &= 1 - \mathrm{e}^{-\lambda} \\ &= 1 - \exp(-nm(1-p)p^t).\end{aligned}$$

- *P-value; probability value*:
  this is the probability that a hit would attain at least the given score, by random chance for the search database
- *E value; expectation value*:
  this is the expected number of hits of at least the given score that you would expect by random chance for the search database

E-value interpretation in BLAST

- E-value < 10e-100: Identical sequences.
  - You will get long alignments across the entire query and hit sequence.
- 10e-50 < E-value < 10e-100: Almost identical sequences.
  - A long stretch of the query protein is matched to the database.
- 10e-10 < E-value < 10e-50: Closely related sequences.
  - Could be a domain match or similar.
- 1 < E-value < 10e-6: Could be a true homologue but it is a gray area.

Which sequence similarity algorithm to choose?

- A combination of the Smith-Waterman algorithm with a reduction of the search space (such as k-word identification in FASTA) may speed up the process.
- BLAST and FASTA are heuristic approximations of dynamic programming algorithms. These approximations are less sensitive (then f.i. Smith-Waterman) and do not guarantee to find the best alignment between two sequences. However, these methods are not as time-consuming as they reduce computation time and CPU usage [Shpaer et al., 1996].
- Hence, the researcher needs to make a choice between having a fast and effective data analysis or reducing the risk of missing important information by using the most sensitive algorithms for data base searching

9 Tutorial (at home)

- R scripts used throughout this Chapter can be replayed via the code included in the file

  RCode to Chapter5 and BackgroundInfo.7z

- R scripts illustrating relevant R packages for sequence pattern recognition and sequence comparison (see also practical session):
  - DNA sequence statistics: http://a-little-book-of-r-for-bioinformatics.readthedocs.org/en/latest/src/chapter1.html
  - Quering sequence data bases: http://a-little-book-of-r-for-bioinformatics.readthedocs.org/en/latest/src/chapter3.html
  - Pairwise sequence alignment: http://a-little-book-of-r-for-bioinformatics.readthedocs.org/en/latest/src/chapter4.html
  - Multiple alignments and phylogenetic analysis: http://a-little-book-of-r-for-bioinformatics.readthedocs.org/en/latest/src/chapter5.html
  - Computational gene finding: http://a-little-book-of-r-for-bioinformatics.readthedocs.org/en/latest/src/chapter7.html
  - Comparative genomics: http://a-little-book-of-r-for-bioinformatics.readthedocs.org/en/latest/src/chapter9.html

Main reference

- Deonier et al. *Computational Genome Analysis*, 2005, Springer.
  (Chapters 6,7)

Background reading

- Pabinger et al. 2013. A survey of tools for variant analysis of next-generation genome sequencing data. Briefings in Bioinformatics.
- Pavlopoulos et al. 2013. Unraveling genomic variation from next generation sequencing data. BioData Mining 6:13.

Guiding questions are provided on the Theory Course Website.