

BC205: Algorithms for Bioinformatics. II. Sequence Analysis

Christoforos Nikolaou

March 15th, 2018

The biological problems:

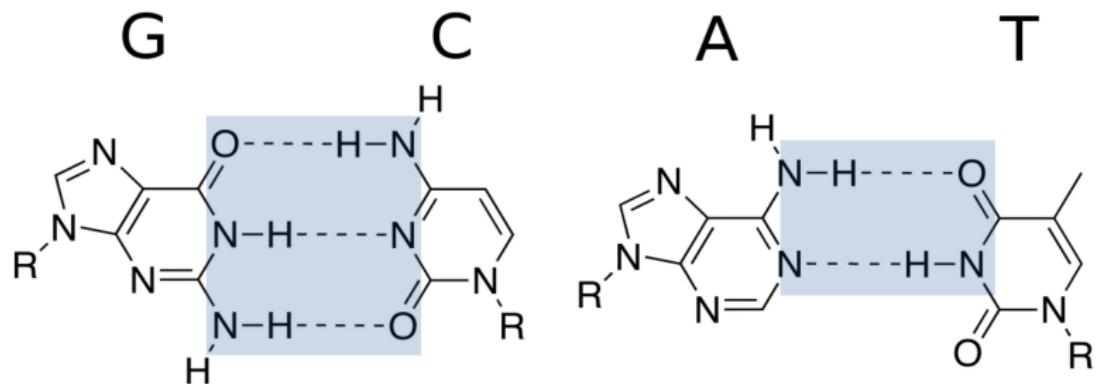
- ▶ Compare different species on the basis of DNA composition
- ▶ Find evidence of horizontal gene transfer in a bacterial genome
- ▶ Locate the Origin of Replication of a Bacterial Genome

Aspects of DNA Composition

- ▶ GC content
- ▶ genomic signatures
- ▶ parity distributions
- ▶ k-mer frequencies

GC content

We call GC content (or GC%) the ratio of (G+C) nucleotides of a given DNA sequence * Why is it important:

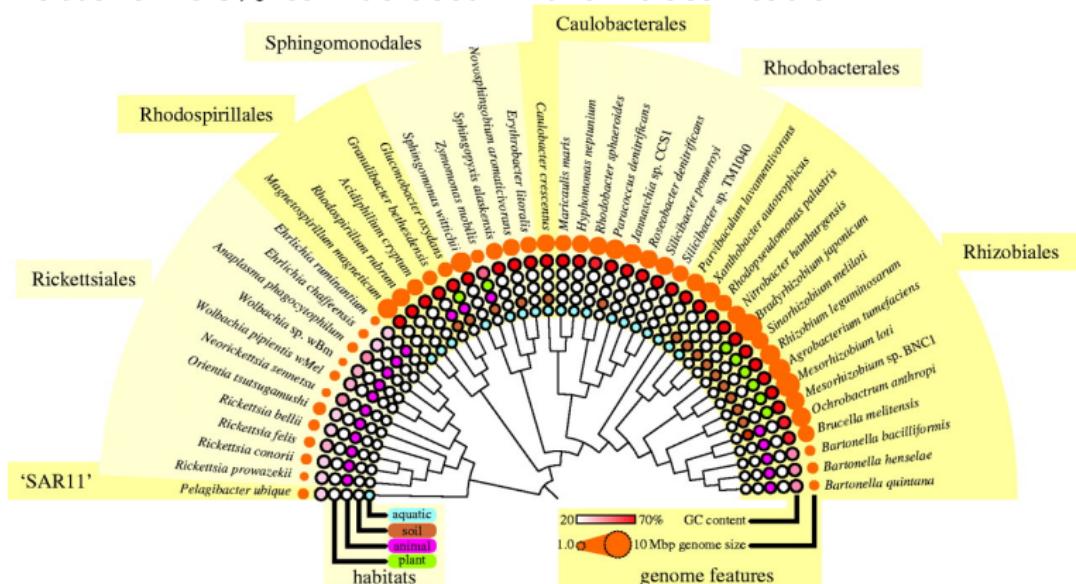


GC is related to:

- ▶ Biochemical level: Thermal stability
- ▶ Evolutionary level: Organism Phylogeny, Mutational pressures
- ▶ Genomic level: Genome size
- ▶ Functional level: Functional role of underlying sequences
- ▶ and many more

GC content in Genomic Sequences

- ▶ Bacteria: GC% is highly variable **between** species
- ▶ Bacteria: GC% is rather homogeneous **within** each genome
- ▶ Bacteria: GC% can be used in their classification



GC content in Genomic Sequences

- ▶ Eukaryotes: Very homogeneous overall GC% (~40-45% in all animals)
- ▶ Eukaryotes: Fluctuation of GC content along the chromosomes and organization in areas of (rather) stable GC%
- ▶ Eukaryotes: Regions of stable high/low GC content that segregate mammalian genomes in isochores

Isochores in a mammalian genome

Ισόχωρες Περιοχές στο Γονιδίωμα του Ποντικιού
(*Mus musculus*)

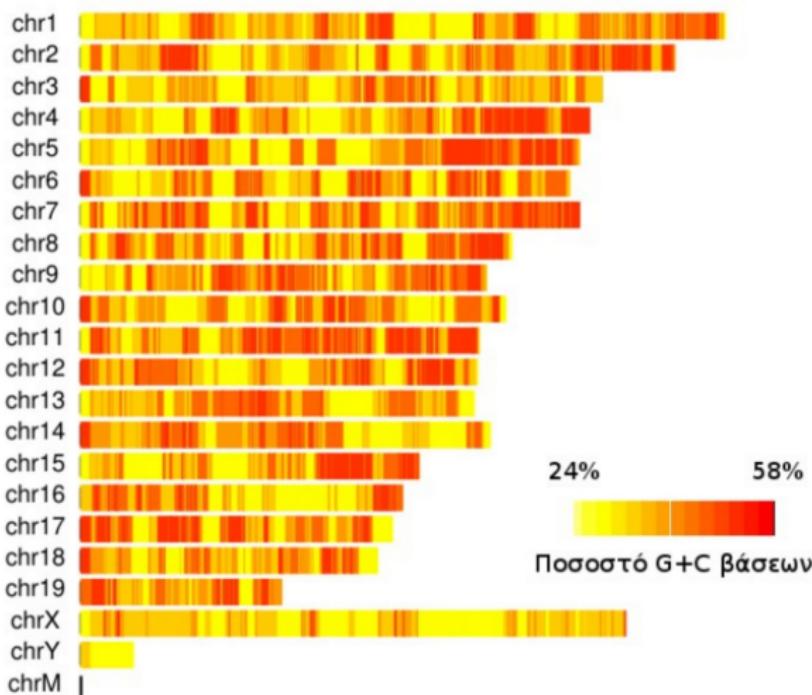


Figure 1: GC content along a mammalian genome

Problem 1: GC content in Bacterial Genomes

- ▶ Given the DNA sequence of a Bacterial Genome, calculate its GC content:
 - ▶ Read the Sequence
 - ▶ Enumerate G
 - ▶ Enumerate C
 - ▶ Divide $(G+C)$ over length of the sequence

Problem 1: Implementation

```
f = open('ecoli.fa', 'r')
seq = ""
window=1000
total = 0
A=T=G=C=[]
times=0;
for line in f:
    x=re.match(">", line)
    if x == None:
        length=len(line)
        total=total+length
        seq=seq+line[0:length-1]
f.close()
C=seq.count("C")
G=seq.count("G")
print (float(G)+float(C))/len(seq);
```

Hands on #1:

- ▶ Download a couple of bacterial genome sequences from ENSEMBL Bacteria
(<http://bacteria.ensembl.org/index.html>)
- ▶ Implement GC content
- ▶ Report the results

Problem 2: Variability of GC content *between* Bacterial Genomes

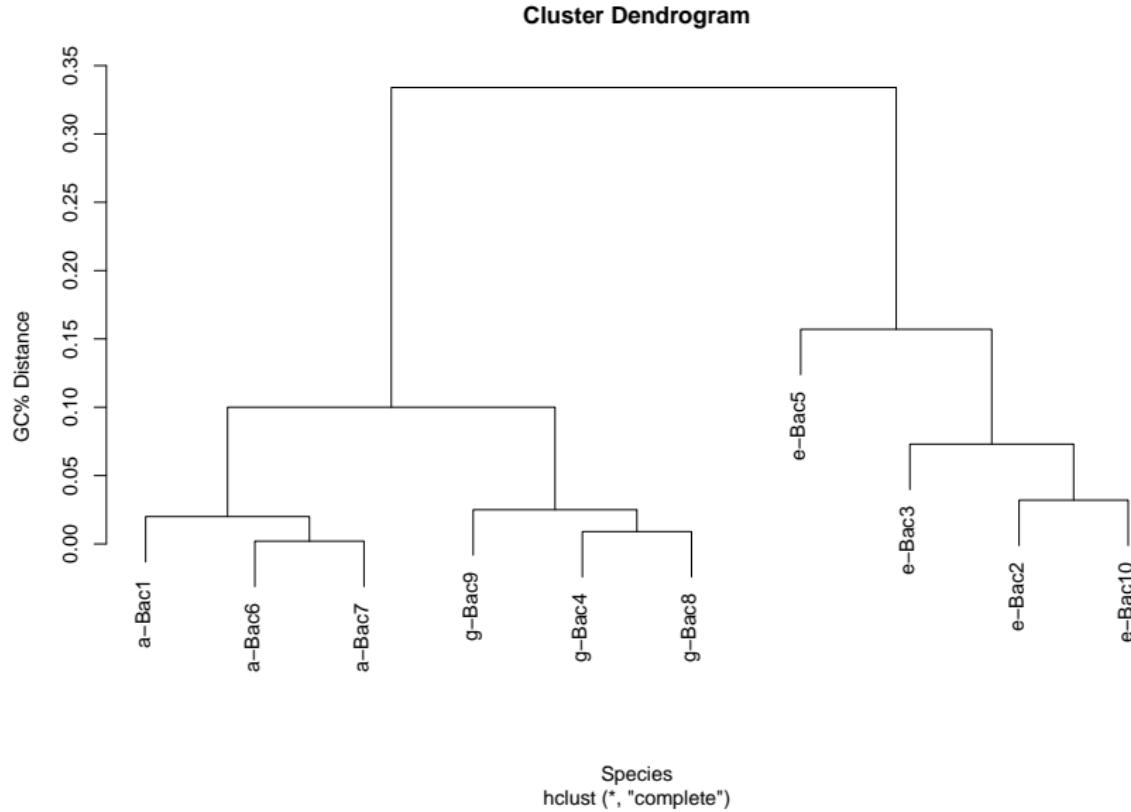
- ▶ Given a number of bacterial genomes:
 - ▶ Get their genome sequences
 - ▶ Calculate the GC contents
 - ▶ Calculate differences between the GC contents
 - ▶ Rank genomes based on their differences
- ▶ Pseudocode:
 - ▶ Perform GC_content on each of the genomes you downloaded
 - ▶ Calculate $D_{(i,j)} = |GC_i - GC_j|$ over all i,j
 - ▶ Sort $D_{(i,j)}$

Problem 2: Approach

- ▶ Instead of Sorting Distances, we can do something better
- ▶ Use clustering (of any type) on the distance matrix

```
gc_values=c(0.334, 0.595, 0.668,
           +0.409, 0.511, 0.352,
           +0.354, 0.418, 0.434, 0.627)
species=c("a-Bac1", "e-Bac2", "e-Bac3",
          +"g-Bac4", "e-Bac5", "a-Bac6", +
          "a-Bac7", "g-Bac8", "g-Bac9", +
          "e-Bac10")
# Create distance matrix of values with dist()
gc_dist<-dist(gc_values)
plot(hclust(gc_dist), labels = species, +
      xlab="Species", ylab="GC% Distance")
```

Problem 2: Approach



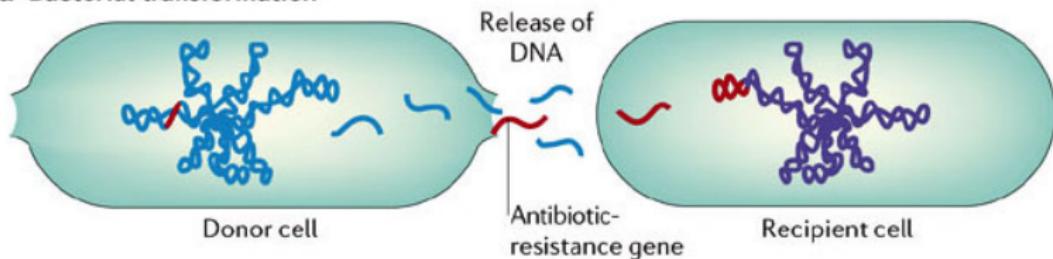
Problem 3: What about different regions of the genome?

- ▶ We just saw how genomic GC% values may be used to draw conclusions for bacterial phylogeny
- ▶ But: How representative is the GC% value you calculated above?
- ▶ And: How efficiently can it be used to describe a genome?

Problem 3: Why should we care?

- ▶ We mentioned that GC% is stable within bacterial genomes
- ▶ **But** Some areas of bacterial genomes are special

a Bacterial transformation



- ▶ Parts of the bacterial genome have been “horizontally” (as opposed to vertically, i.e. from their “mom”) transferred from other species.

Problem 3: Stability of GC content *along* Bacterial Genomes

- ▶ Regions of “strange”, or “divergent” GC% values in a given genome are red flags of HGT. The problem now is:
 - ▶ Given a bacterial genome sequence:
 - ▶ Locate regions of the genome where horizontal gene transfer may have occurred.

Problem 3: Approach

- ▶ Choose a window to scan your sequence. This will be your resolution
- ▶ Calculate GC per window
- ▶ Try to locate GC values that deviate from the genome average

Problem 3: The core

```
window=1000
step=100
times=len(seq)/step;

for i in range(times):
    DNA=seq[i*step:i*step+window]
    A=DNA.count("A")
    T=DNA.count("T")
    C=DNA.count("C")
    G=DNA.count("G")
    print i*step, "\t", i*step+window, "\t",
+(float(G)+float(C))/window;
```

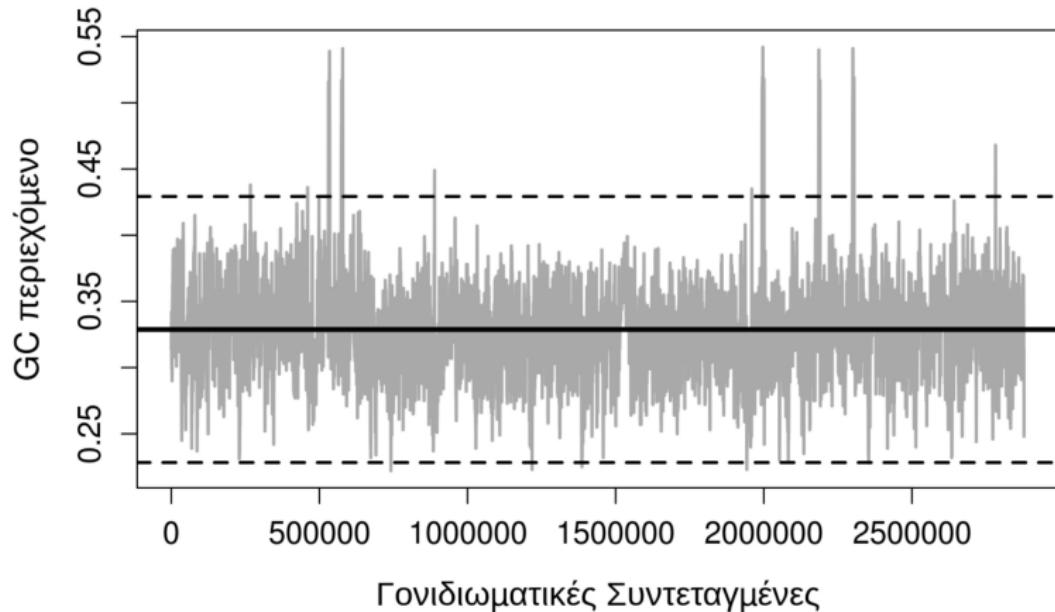
Hands on #2:

- ▶ Get the genome sequence of St. aureus
- ▶ Implement Sliding GC
- ▶ Plot the results in R

GC content along the Genome of St. aureus

- ▶ It should look something like this
- ▶ Now how do we locate HGT candidates?

Staphylococcus aureus



Problem 3: Statistics Interlude

- ▶ Given a set/sample of values, how can we decide on whether a value could be part of that sample or not?
- ▶ In our problem: We know that the GC% of bacteria tends to be characteristic of the genome. Can we “spot” regions of the genome that bear GC% values that are *different* from that characteristic value?
- ▶ Q1: How will we define that characteristic value?
- ▶ Q2: How will we quantify the *difference* as big enough or not?

Problem 3: Theoretical basis (simplified)

- ▶ Central Limit Theorem (simplified):
 - ▶ Regardless of the underlying distribution, the mean of a large number of samples follow the normal distribution.
 - ▶ We can thus model GC values per window based on the normal distribution

Modeling with the Normal Distribution

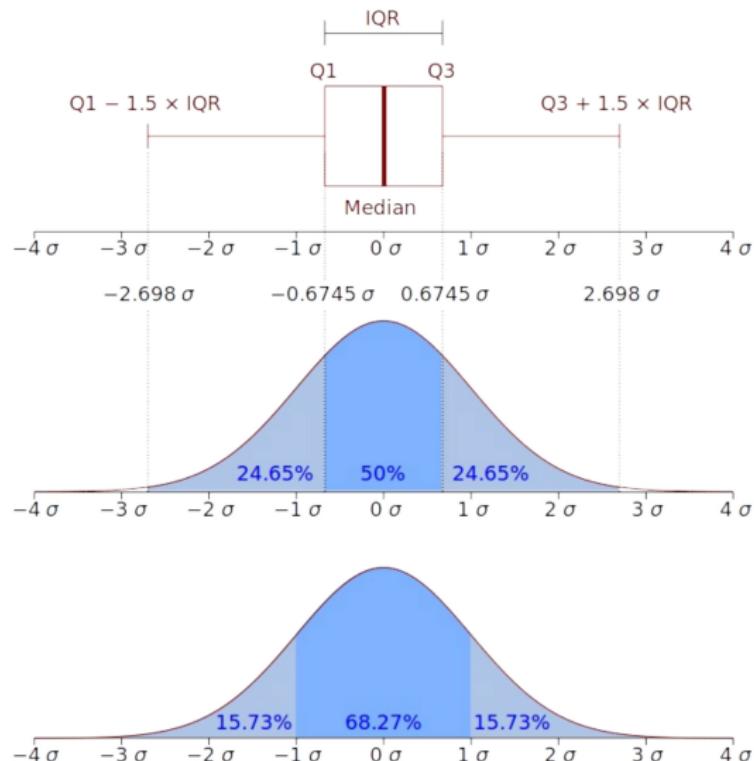


Figure 2: The Bell Curve

Problem 3: The statistics

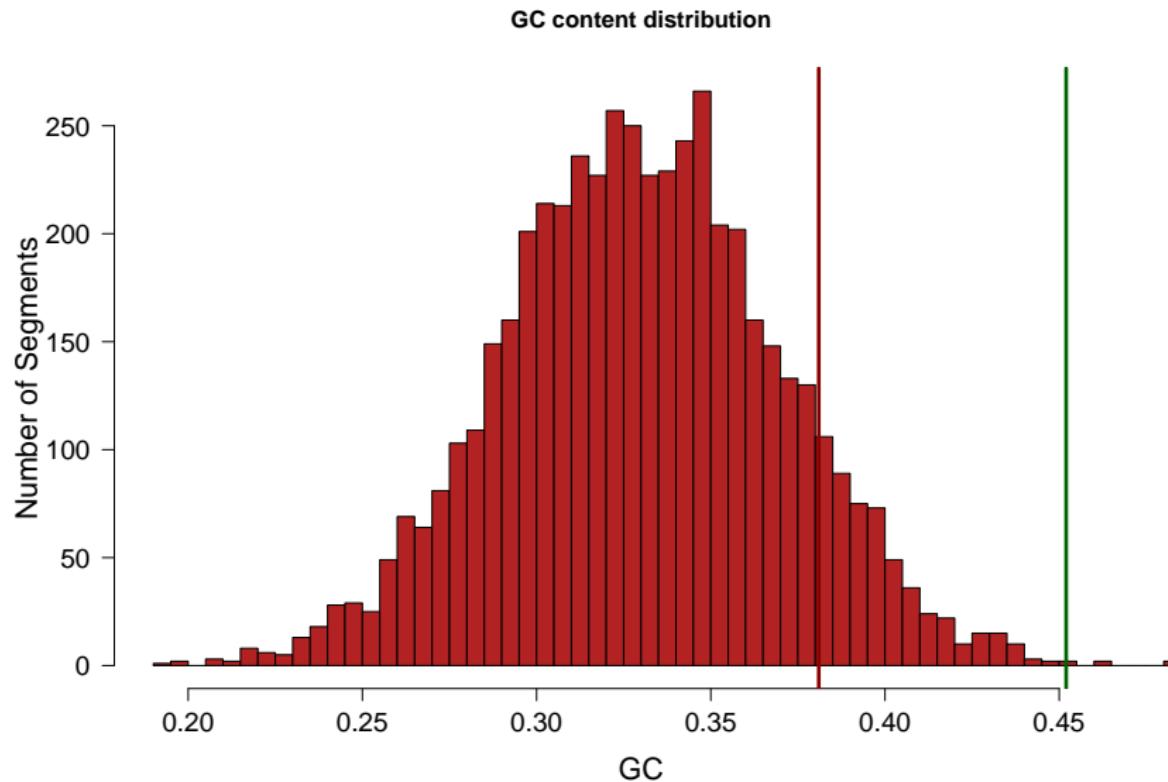
- ▶ We will model the “characteristic value” as the mean of GC values for all windows
- ▶ We will also calculate the standard deviation of these values to model variance

```
gc_mean=0.33  
gc_sd=0.04  
x<-rnorm(5000, mean=gc_mean, sd=gc_sd)  
my_gc1=0.381  
my_gc2=0.452
```

Problem 3: The statistics

```
hist(x, breaks=100, col="firebrick",
      +main="GC content distribution",
      +las=1, xlab="GC", ylab="Number of Segments",
      +xlim=c(gc_mean-3.5*gc_sd, gc_mean+3.5*gc_sd))
abline(v=my_gc1, col="darkred", lwd=3)
abline(v=my_gc2, col="darkgreen", lwd=3)
```

Problem 3: The statistics



Z-transformation

- ▶ Notice the difference between the position of the two vertical bars in the previous plot. One is much more “inside” the distribution than the other
- ▶ Can we have a quantitative measure of this?
- ▶ Given a value x , we can compare x to a normal distribution with $\text{mean}=m$ and $\text{standard deviation}=std$ with the z-score:
$$Z(x) = (x - m) / std$$

$Z(x)$ is thus the difference of x from m in units of standard deviation.

Knowing that in a normal distribution $\sim 99\%$ of the values fall within $+\/-2*std$ a value of $Z(x)>2$ or $Z(x)<-2$ makes it highly unlikely that x is part of our distribution.

Problem 3: The Statistics

```
gc_mean=0.33
gc_sd=0.04
x<-rnorm(5000, mean=gc_mean, sd=gc_sd)
my_gc1=0.381
my_gc2=0.452
z1=(my_gc1-gc_mean)/gc_sd
z1
```

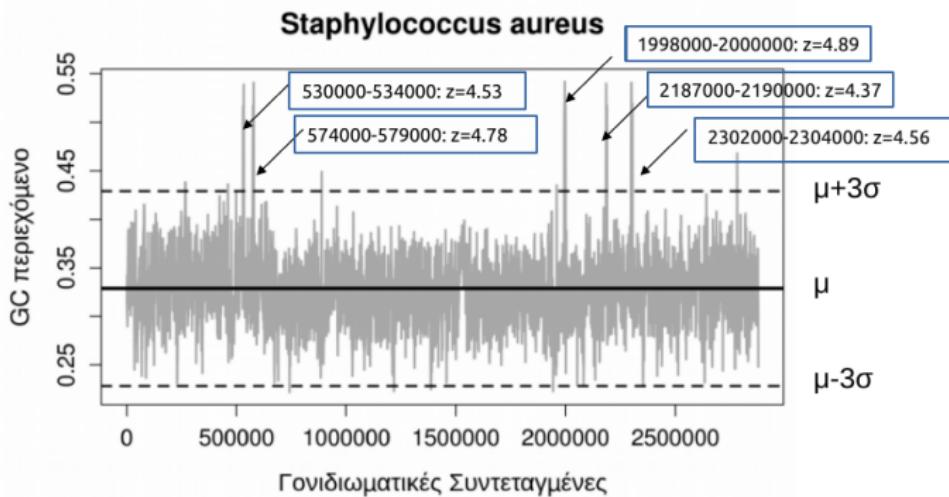
```
## [1] 1.275
```

```
z2=(my_gc2-gc_mean)/gc_sd
z2
```

```
## [1] 3.05
```

Problem 2: Revisited

Let's take another look at the GC analysis of the St. aureus genome



Θέσεις με πολύ υψηλό z φαίνονται γραφικά στο σχήμα.

Οι θέσεις αυτές είναι οι θέσεις των γονιδίων του ριβοσωμικού RNA (rRNA)!

The problem:

- ▶ Background DNA composition has some **functional** role besides simply reflecting mutational pressures
- ▶ This means that in some cases we need to know why the local composition is guided by *other* aspects of molecular evolution.
E.g. why would rRNA genes be G+C-rich even in AT-rich genomes?
- ▶ We need to find a way to control for *background nucleotide composition*

Problem 2 Revisited: Distinguishing between genomes through their sequence composition

1. Going beyond the GC content
 2. Going beyond simple bases (mononucleotides, k=1)
 3. Analyzing all dinucleotide frequencies of k=2
- Pseudocode:
- For each kmer in 4^k k-mers
 - Calculate N(kmer)
 - Create a table

Problem 2 Revisited: K-mer frequencies

```
import re
import math
import itertools

bases=['A', 'T', 'G', 'C']
k=2
kmer=[''.join(p) for p in
+itertools.product(bases, repeat=k)]
counts={}

for i in kmer:
    counts[i]=seq.count(str(i))
    print '%s %.3f' % (str(i),float(counts[i])/len(seq))
```

Problem 2 Revisited: A table of 4^k frequencies of occurrence

Base	A	T	G	C
A	0.090	0.112	0.048	0.053
T	0.095	0.090	0.064	0.053
G	0.052	0.052	0.023	0.034
C	0.066	0.048	0.026	0.023

- ▶ Values may be seen as “probabilities” of finding each k-mer in the sequence
- ▶ Can we use the notion of the probability to modify the table so that we get rid of the background nucleotide composition?

Problem 2 Revisited: Removing Background Composition

- ▶ The problem stated above persists at the level of k-mers: The background DNA composition may affect our results
- ▶ At the k-mer level we can remove the background using ratios of observed/expected frequencies
- ▶ Which is the expected frequency of a given k-mer?

Problem 2 Revisited: Observed/Expected(o/e) k-mer frequencies

- ▶ Mathematics Interlude:
 - ▶ Assume two events A, B that are linked with each other
 - ▶ We then say that A and B are dependent (or conditioned) and we have a “conditional probability” of A happening given B is also happening
 - ▶ We can think of k-mers the same way: a k-mer is more probable to occur if its constituent mono-mers are occurring
 - ▶ Bottomline: Any given k-mer's frequency of occurrence is dependent on the frequencies of occurrence of its mononucleotides. Thus:

Given a k-mer of length k the o/e-ratio frequency is defined as:

$$R[N_1 N_2 \dots N_k] = F[N_1 N_2 \dots N_k] / (F[N_1] F[N_2] \dots F[N_k])$$

In this way we can define a new table of modified frequencies that is independent of mono-nucleotide composition

Problem 2 Revisited: Observed/Expected K-mer frequencies

```
bases=['A', 'T', 'G', 'C']
k=1
kmer1=[''.join(p) for p in
+itertools.product(bases, repeat=k)]
k=2
kmer2=[''.join(p) for p in
+itertools.product(bases, repeat=k)]
oecounts={}

for i in kmer2:
    bg=list(i)
    oecounts[i]=float(seq.count(str(i)))/len(seq)
    for j in bg:
        oecounts[i]/=(float(seq.count(j))/len(seq))
print '%s %.3f' % (str(i),oecounts[i])
```

Problem 2 Revisited: A table of o/e 4^k frequencies of occurrence

Base	A	G	C	T
A	0.800	0.997	0.878	0.949
G	0.848	0.799	1.174	0.957
C	0.946	0.955	0.848	1.252
T	1.183	0.872	0.946	0.841

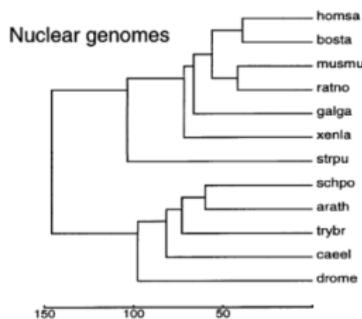
- ▶ Notice how values now go >1 . What does this mean?
- ▶ How is this table better (or not) than the previous one?

Genomic Signatures: Comparing o/e k-mer composition

- ▶ Genomic Signatures are defined as the table of o/e k-mers for a given genome
- ▶ We can use these tables to analyze distances between genomes.
(Hint: even eukaryote genomes!)

$$\rho_{XY}^* = f_{XY}^*/f_X^*f_Y^*$$

$$\delta^*(p, q) = \frac{1}{16} \sum_{XY} |\rho_{XY}^*(p) - \rho_{XY}^*(q)|,$$



Hands on #3:

- ▶ Get chromosome 1 from (human, mouse, fly, worm, yeast)
- ▶ Use a genomic signature approach to cluster genomic signatures from different genomes
- ▶ Calculate the distance between $\rho_{xy}(p)$ and $\rho_{xy}(p)$ to create a table of distances.

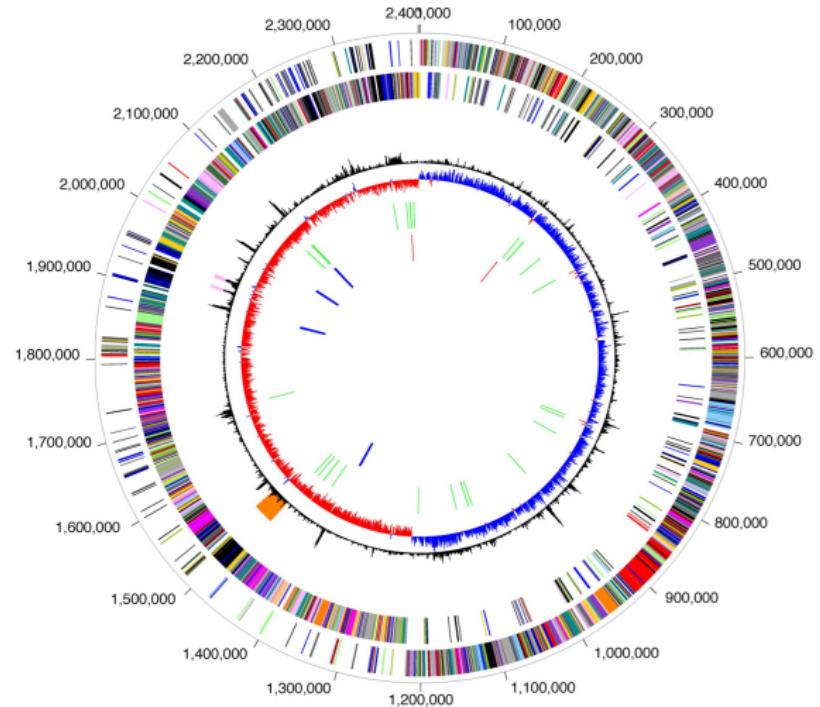
Hands on #4:

- ▶ The approach used above for $k=2$ works rather well because the k -mers are 16
- ▶ What about larger numbers of k ($k=7$ or more)? Would you use the same approach?
- ▶ What is the main problem with increasing k and how would you solve it?
- ▶ Try this as an exercise.

Accumulating arrays/hashes

1. Starting with a window of length= k
2. Read the first k nucleotides of the sequence S in N_k
3. Increment a hash table with key= N_k
4. Continue to the end of the sequence

Problem 4: Finding the DNA Replication in a bacterial genome



What we know

- ▶ Due to the pioneering work of E. Chargaff we know that A~T and G~C in **single-stranded DNA**
- ▶ We know that this holds for all complete genomes except very few exceptions
- ▶ The exceptions are the few genomes that **do not** replicate symmetrically
- ▶ DNA-strand parity:
 - ▶ Strand X is replicated in-continuously
 - ▶ Accumulates more substitutions
 - ▶ If substitutions are biased the strand will guide the change in both strands through base-pairing

Why should you care?

Γονιδίωμα *S. aureus*

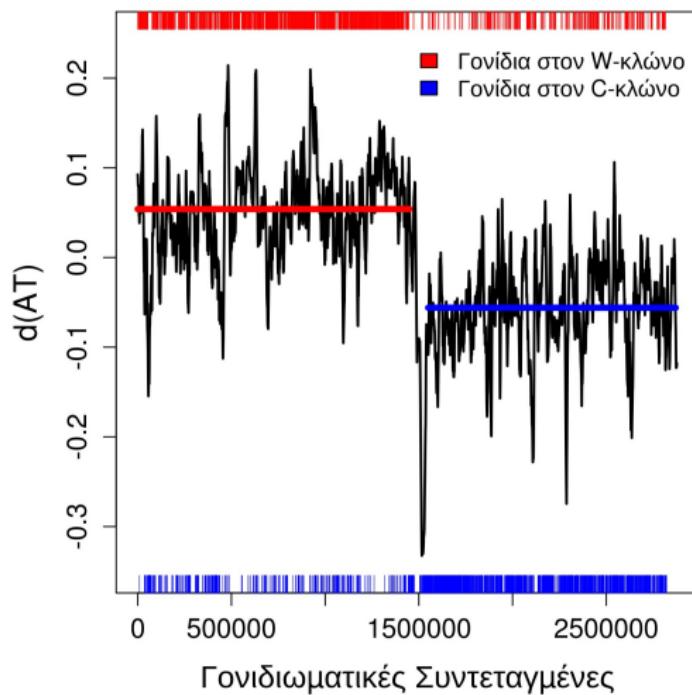


Figure 4: Nucleotide Parity

Approaching the problem

- ▶ We thus expect (and observe) the parity to be violated and that this violation occurs symmetrically on either side of the OriC
- ▶ We are looking for a way to locate this *phase transition* in the parity violation
- ▶ We thus need:
 - ▶ A measure of the parity
 - ▶ A way to monitor this measure along the genome
 - ▶ A way to locate abrupt changes in its values

Breaking the problem into pieces

1. Analyze the DNA composition *along* the genome
 2. Calculate a quantity that will be informative
 3. Create a condition that will test the location of the Ori
-
- ▶ Pseudocode: Given a bacterial genome:
 - ▶ Count nucleotides in windows of N base pairs
 - ▶ Calculate the scaled AT-skew as $(A-T)/(A+T)$
 - ▶ Create an array of the skew values along the genome
 - ▶ Locate the transition point

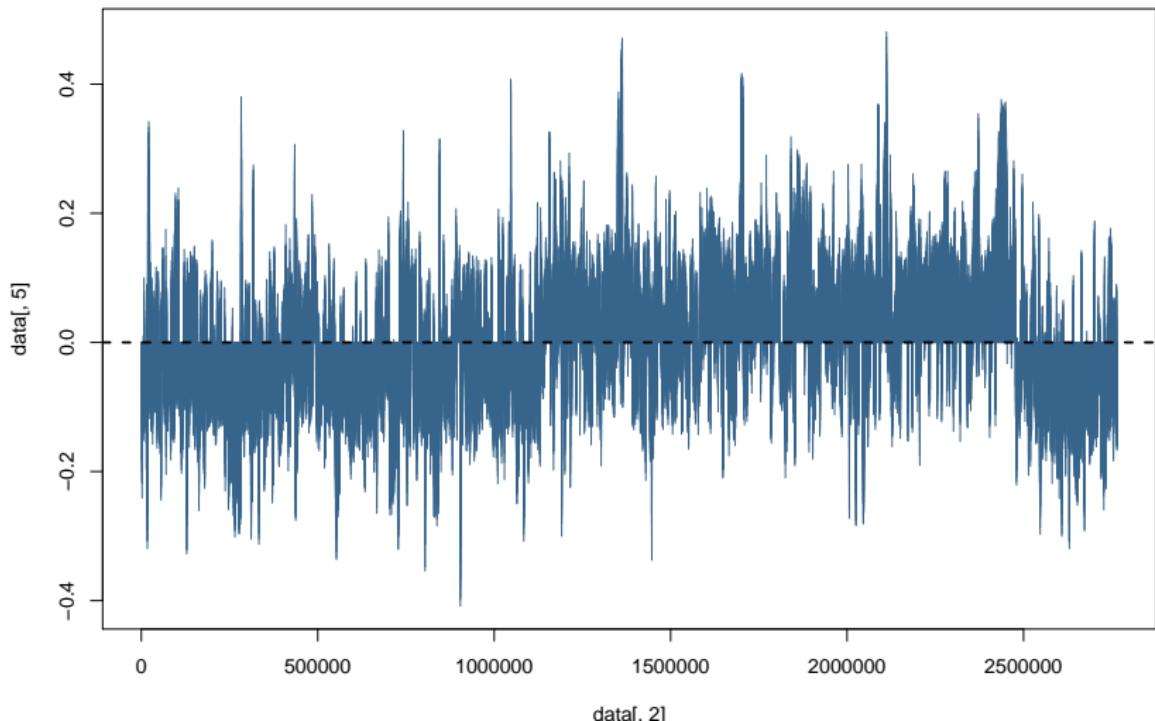
Problem 4: Parity Measure Implementation

```
window=1000
step=100
times=len(seq)/step;

for i in range(times):
    DNA=seq[i*step:i*step+window]
    A=DNA.count("A")
    T=DNA.count("T")
    C=DNA.count("C")
    G=DNA.count("G")
    print i*step,"\t",i*step+window,"\t",float(A-T)/(A+T)
```

Problem 4: Plotting the Values

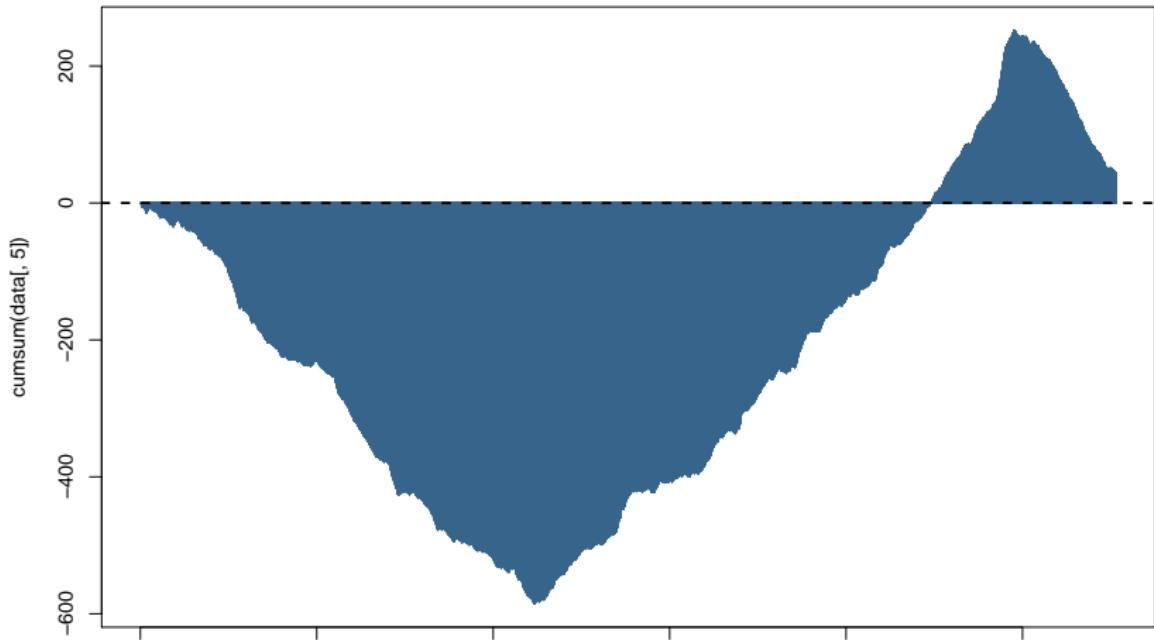
```
data<-read.delim("out", header=F)
plot(data[,2], data[,5], type="h", col="steelblue4")
abline(h=0, lty=2, lwd=2)
```



Problem 4: Plotting the Values

- ▶ Using a cumulative approach often helps

```
data<-read.delim("out", header=F)
plot(data[,2], cumsum(data[,5]), type="h", col="steelblue4"
      abline(h=0, lty=2, lwd=2)
```



Problem 4: Locating the breakpoint(s)

- ▶ Not a simple problem. In fact one (breakpoint detection) for which research is ongoing in many fields
- ▶ Things you could try:
 - ▶ Using derivation (checking the difference between each value and the previous one)
 - ▶ Density-based approaches: Trying to locate the region around which changes in the sign occur more robustly (i.e. given many different points around it)

Exercises: To think about

- ▶ Use a genomic signature approach to locate possible HGT genes in the genome of St. aureus. Do your results of “outliers” differ from those obtained with the GC content approach?
- ▶ Write a complete program to locate the origin of replication for the two genomes attached at the end of the page, using the parity rules described in the lecture.