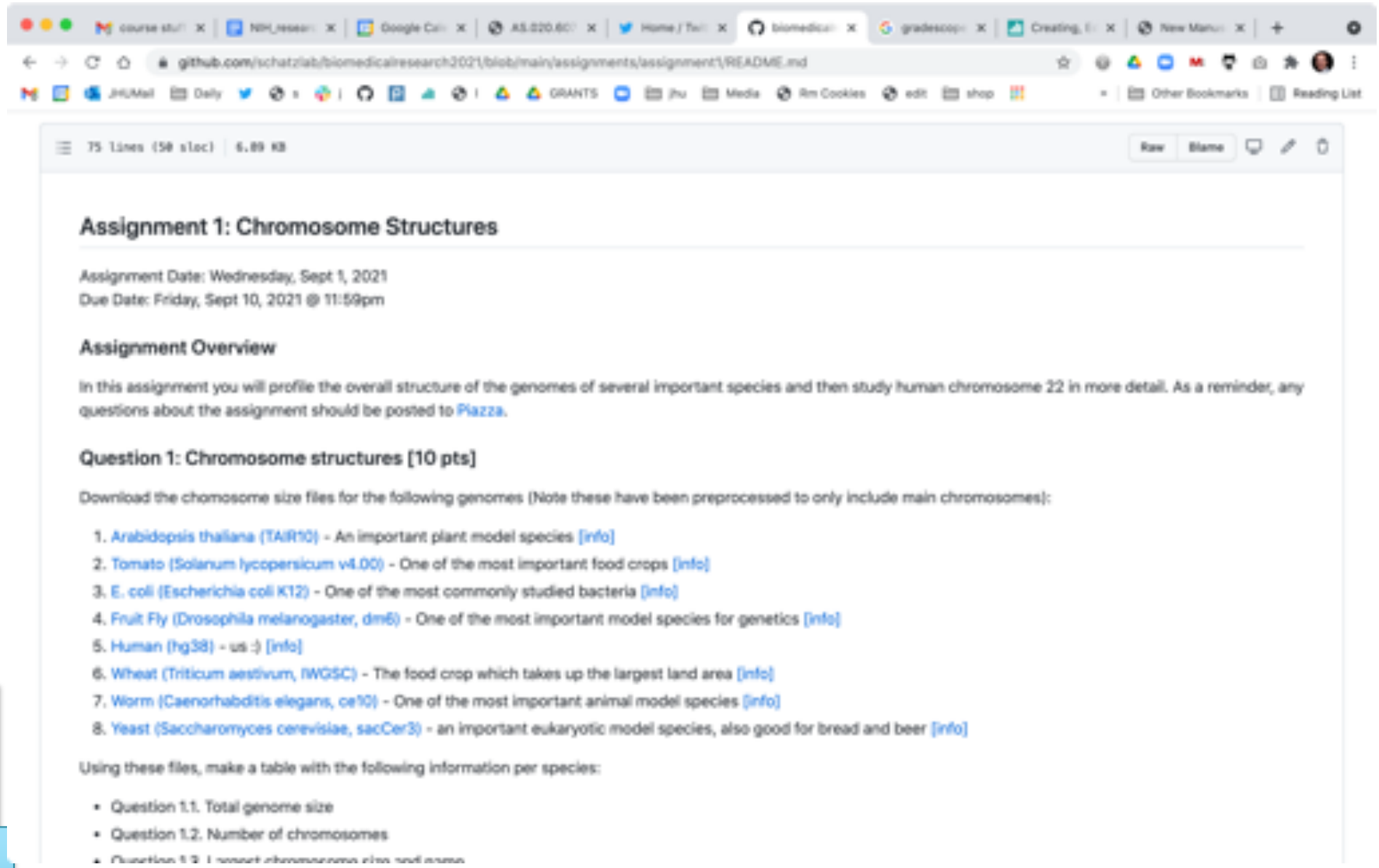
Whole Genome Alignment

Michael Schatz

Sept 13, 2021

Lecture 4: Biomedical Research

Assignment 1: Chromosome Structures
Due Friday Sept 10 @ 11:59pm



github.com/schatzlab/biomedicalresearch2021/blob/main/assignments/assignment1/README.md

75 lines (50 sloc) | 6.89 KB

Raw | Blame

Assignment 1: Chromosome Structures

Assignment Date: Wednesday, Sept 1, 2021
Due Date: Friday, Sept 10, 2021 @ 11:59pm

Assignment Overview

In this assignment you will profile the overall structure of the genomes of several important species and then study human chromosome 22 in more detail. As a reminder, any questions about the assignment should be posted to Piazza.

Question 1: Chromosome structures [10 pts]

Download the chomosome size files for the following genomes (Note these have been preprocessed to only include main chromosomes):

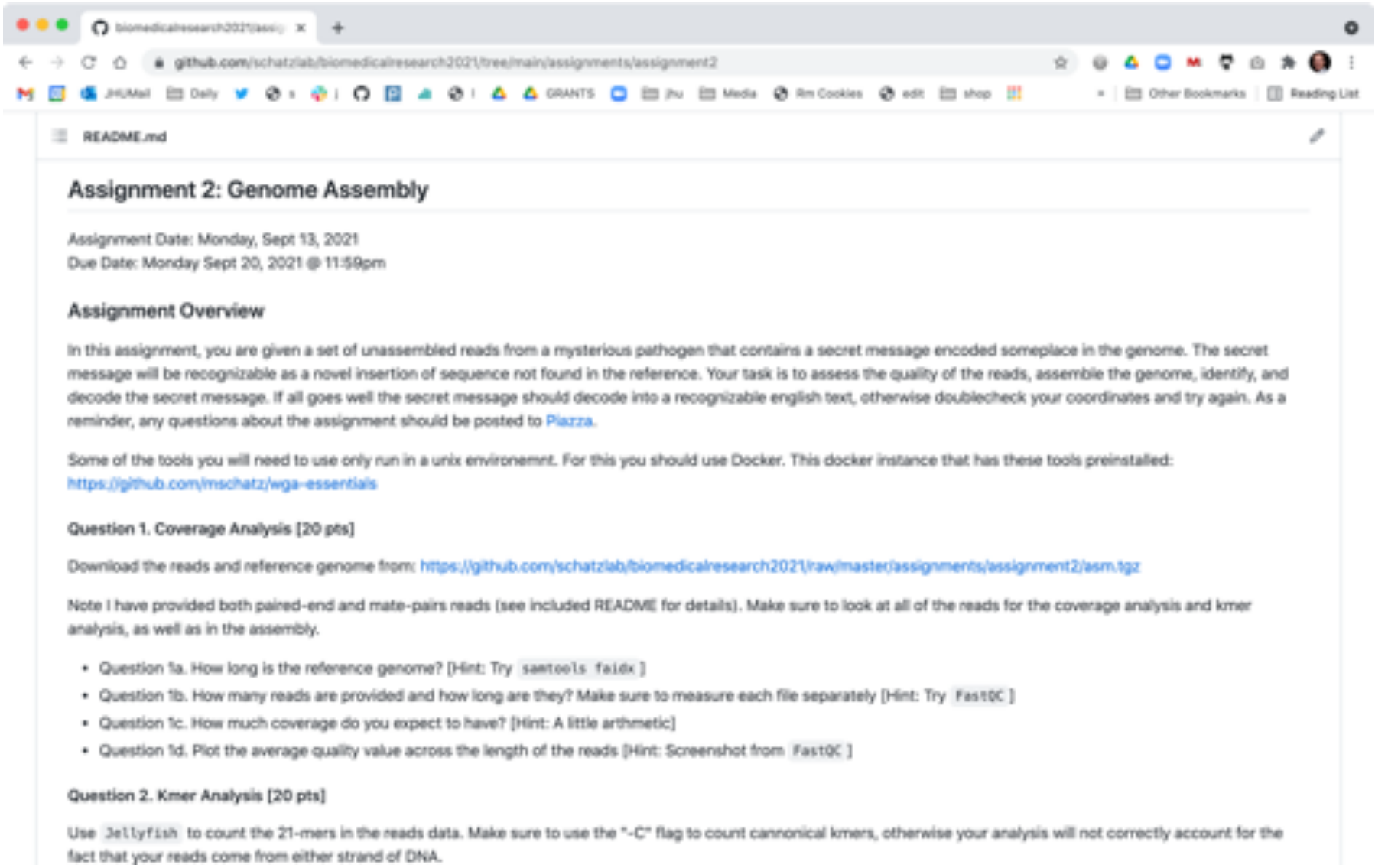
1. Arabidopsis thaliana (TAIR10) - An important plant model species [info]
2. Tomato (Solanum lycopersicum v4.00) - One of the most important food crops [info]
3. E. coli (Escherichia coli K12) - One of the most commonly studied bacteria [info]
4. Fruit Fly (Drosophila melanogaster, dm6) - One of the most important model species for genetics [info]
5. Human (hg38) - us :) [info]
6. Wheat (Triticum aestivum, IWGSC) - The food crop which takes up the largest land area [info]
7. Worm (Caenorhabditis elegans, ce10) - One of the most important animal model species [info]
8. Yeast (Saccharomyces cerevisiae, sacCer3) - an important eukaryotic model species, also good for bread and beer [info]

Using these files, make a table with the following information per species:

- Question 1.1. Total genome size
- Question 1.2. Number of chromosomes
- Question 1.3. Largest chromosome size and name

https://github.com/schatzlab/biomedicalresearch2021

Assignment 2: Genome Assembly
Due Monday Sept 20 @ 11:59pm



README.md

Assignment 2: Genome Assembly

Assignment Date: Monday, Sept 13, 2021
Due Date: Monday Sept 20, 2021 @ 11:59pm

Assignment Overview

In this assignment, you are given a set of unassembled reads from a mysterious pathogen that contains a secret message encoded someplace in the genome. The secret message will be recognizable as a novel insertion of sequence not found in the reference. Your task is to assess the quality of the reads, assemble the genome, identify, and decode the secret message. If all goes well the secret message should decode into a recognizable english text, otherwise doublecheck your coordinates and try again. As a reminder, any questions about the assignment should be posted to Piazza.

Some of the tools you will need to use only run in a unix environemnt. For this you should use Docker. This docker instance that has these tools preinstalled: https://github.com/mschatz/wga-essentials

Question 1. Coverage Analysis [20 pts]

Download the reads and reference genome from: https://github.com/schatzlab/biomedicalresearch2021/raw/master/assignments/assignment2/asm.tgz

Note I have provided both paired-end and mate-pairs reads (see included README for details). Make sure to look at all of the reads for the coverage analysis and kmer analysis, as well as in the assembly.

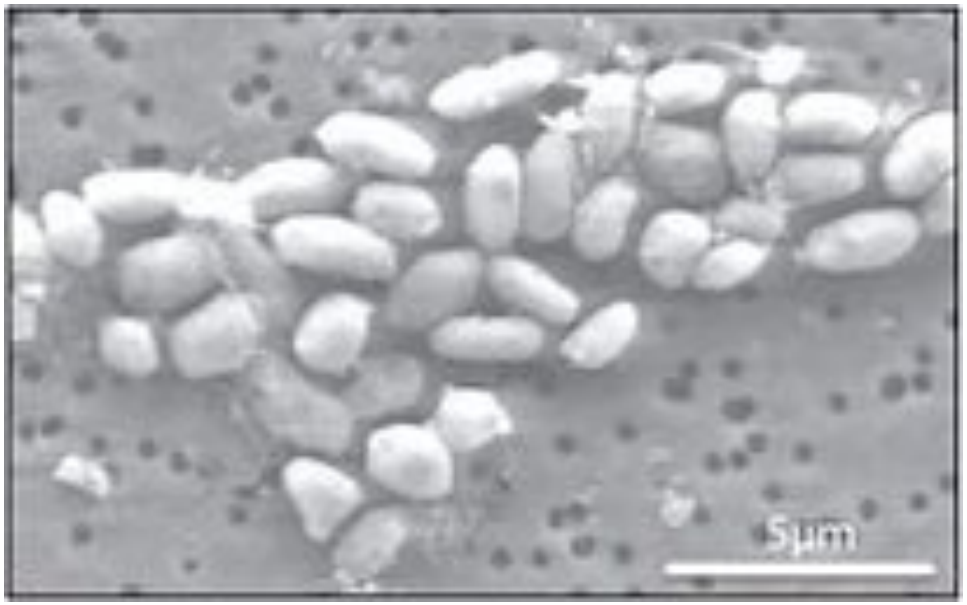
- Question 1a. How long is the reference genome? [Hint: Try `samtools faidx`]
- Question 1b. How many reads are provided and how long are they? Make sure to measure each file separately [Hint: Try `FastQC`]
- Question 1c. How much coverage do you expect to have? [Hint: A little arthmetic]
- Question 1d. Plot the average quality value across the length of the reads [Hint: Screenshot from `FastQC`]

Question 2. Kmer Analysis [20 pts]

Use `Jellyfish` to count the 21-mers in the reads data. Make sure to use the "-C" flag to count cannonical kmers, otherwise your analysis will not correctly account for the fact that your reads come from either strand of DNA.

https://github.com/schatzlab/biomedicalresearch2021

Halomonas sp. GFAJ-1



Library 1: Fragment

Avg Read length: 100bp

Insert length: 180bp

Library 2: Short jump

Avg Read length: 50bp

Insert length: 2000bp

A Bacterium That Can Grow by Using Arsenic Instead of Phosphorus
Wolfe-Simon et al (2010) *Science. 332(6034)1163-1166.*

Digital Information Storage

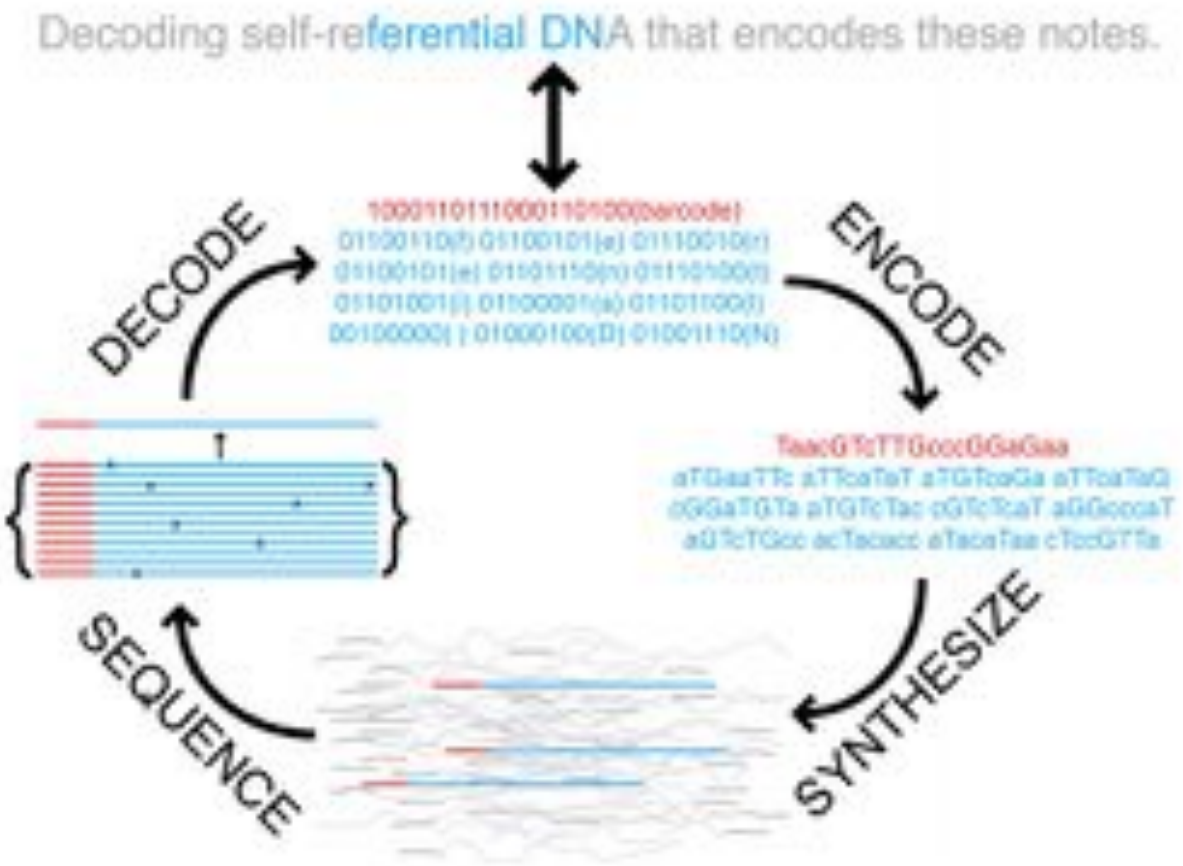


Fig. S1. Schematic of DNA information storage.

Encoding/decoding algorithm implemented in dna-encode.pl from David Dooling.

Next-generation Digital Information Storage in DNA
Church et al (2010) *Science*. 337(6102)1628

Genome Assembly Lab
Due September 20 @ 11:59pm

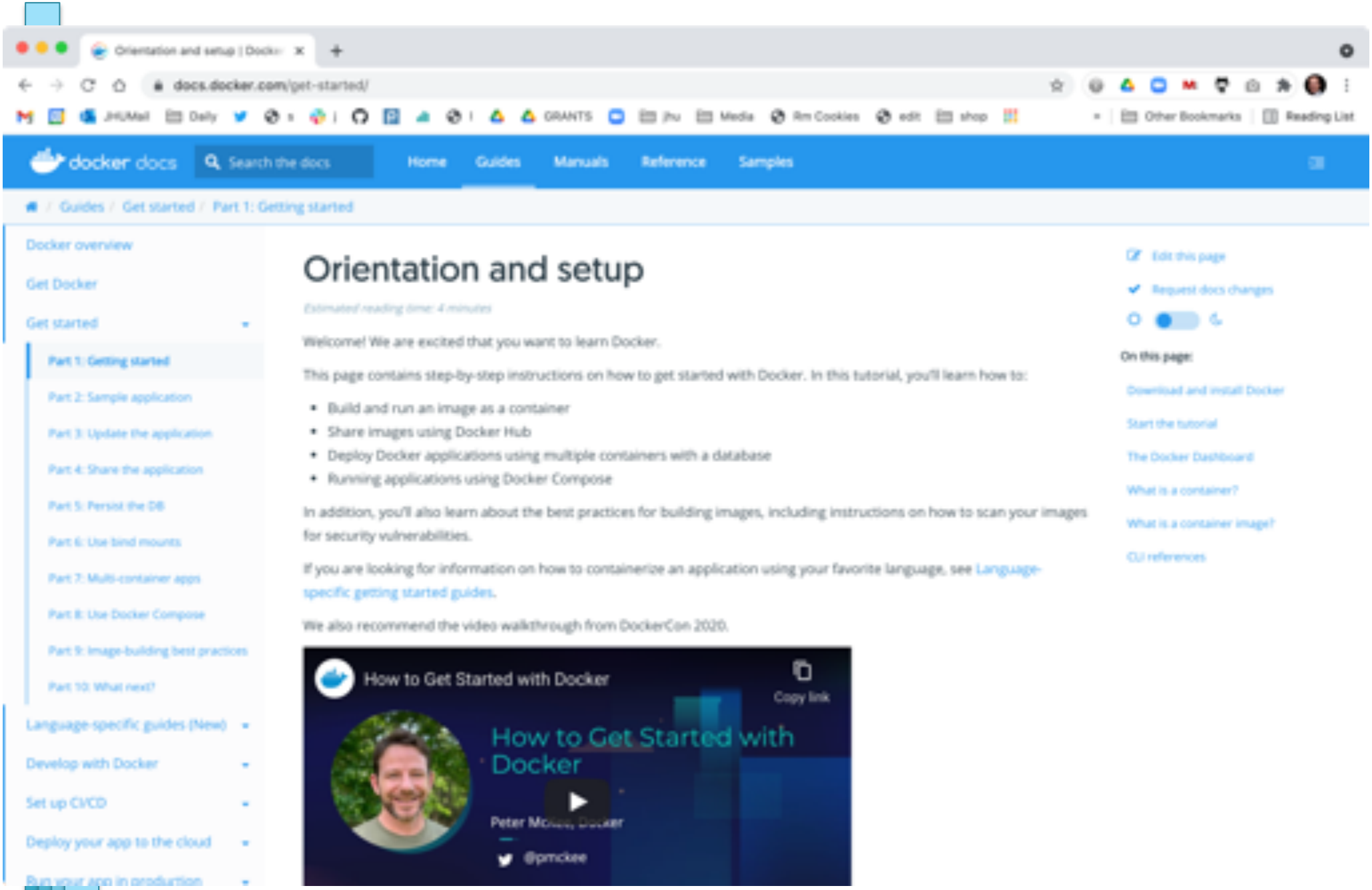
1. ***Initialize Tools***
2. ***Download Reference Genome & Reads***
3. ***Decode the secret message***
    1. *Estimate coverage, check read quality*
    2. *Check kmer distribution*
    3. *Assemble the reads with spades*
    4. *Align to reference with MUMmer*
    5. *Extract foreign sequence*
    6. *dna-decode.py -d*

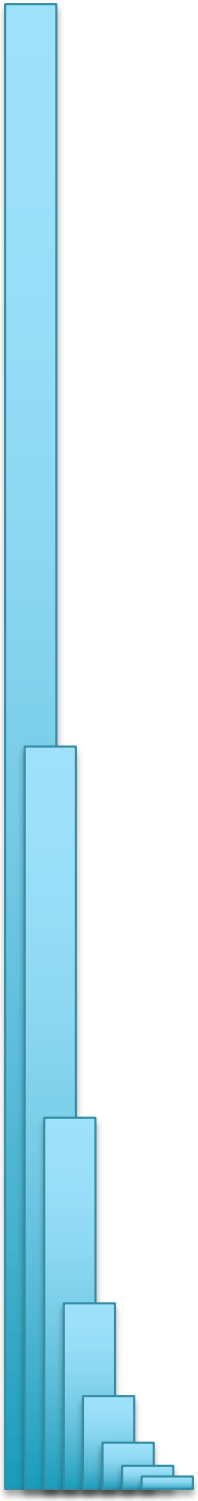
Find and decode

```
nucmer -maxmatch ref.fasta \
   default/ASSEMBLIES/test/final.contigs.fasta
   -maxmatch     Find maximal exact matches (MEMs) without repeat filtering
   -p refctg     Set the output prefix for delta file

show-coords -rclo out.delta
   -r            Sort alignments by reference position
   -c            Show percent coverage
   -l            Show sequence lengths
   -o            Annotate each alignment with BEGIN/END/CONTAINS

samtools faidx default/ASSEMBLIES/test/final.contigs.fasta
   Index the fasta file

samtools faidx default/ASSEMBLIES/test/final.contigs.fasta \
    contig_XXX:YYY-ZZZ > msg.fa

./dna-decode.py -d -input msg.fa
```

***** Note you may need to reverse complement the extracted sequence depending on the orientation of the alignments *****

See manual at http://mummer.sourceforge.net/manual


docs.docker.com/get-started/
docker docs
Search the docs
Home
Guides
Manuals
Reference
Samples
Guides / Get started / Part 1: Getting started
Docker overview
Get Docker
Get started
Part 1: Getting started
Part 2: Sample application
Part 3: Update the application
Part 4: Share the application
Part 5: Persist the DB
Part 6: Use bind mounts
Part 7: Multi-container apps
Part 8: Use Docker Compose
Part 9: Image-building best practices
Part 10: What next?
Language-specific guides (New)
Develop with Docker
Set up CI/CD
Deploy your app to the cloud
Run your app in production
Orientation and setup
Welcome! We are excited that you want to learn Docker.
This page contains step-by-step instructions on how to get started with Docker. In this tutorial, you'll learn how to:
Build and run an image as a container
Share images using Docker Hub
Deploy Docker applications using multiple containers with a database
Running applications using Docker Compose
In addition, you'll also learn about the best practices for building images, including instructions on how to scan your images for security vulnerabilities.
If you are looking for information on how to containerize an application using your favorite language, see Language-specific getting started guides.
We also recommend the video walkthrough from DockerCon 2020.
How to Get Started with Docker
Copy link
How to Get Started with Docker
@pmckee
On this page:
Download and install Docker
Start the tutorial
The Docker Dashboard
What is a container?
What is a container image?
CLI references

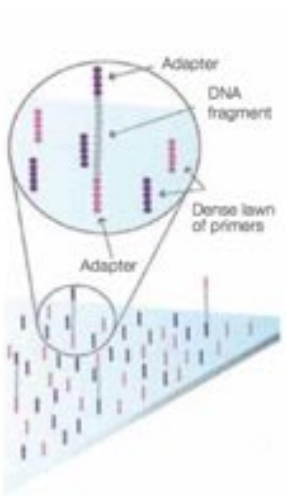


Genome Assembly

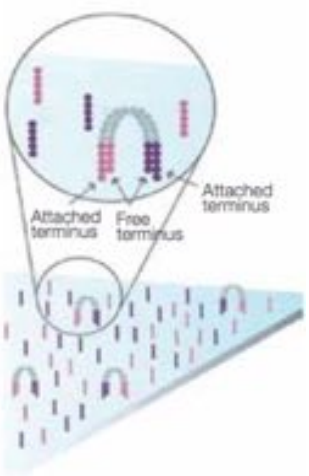
Second Generation Sequencing

Illumina NovaSeq 6000
Sequencing by Synthesis

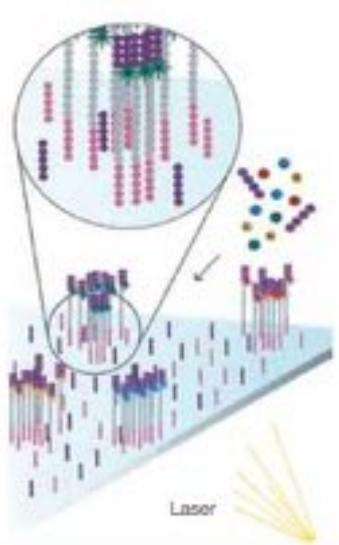
>3Tbp / day



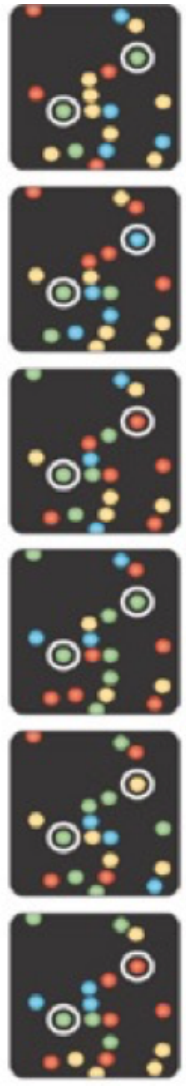
1. Attach



2. Amplify



3. Image

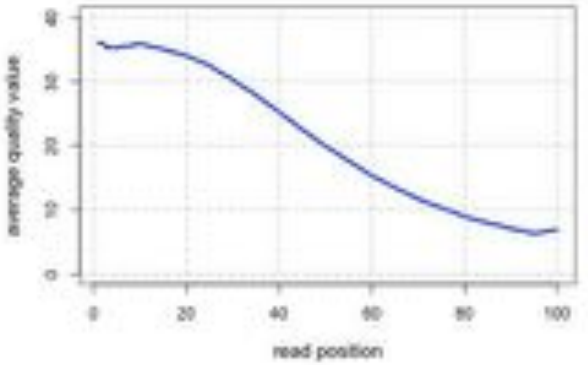


Metzker (2010) Nature Reviews Genetics 11:31-46
https://www.youtube.com/watch?v=fCd6B5HRaZ8

Illumina Quality

QV	p_{error}
40	1/10000
30	1/1000
20	1/100
10	1/10

$$Q_{\text{sanger}} = -10 \log_{10} p$$




```
  SSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSS.....................................................
  ..........................XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX.....................
  ......................................IIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIII.....................
  ......................................JJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJ.....................
  LLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLL.....................................................
  !"#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefghijklmnopqrstuvwxyz{|}~
  |                         |    |        |                              |                     |
 33                        59   64       73                            104                   126

 S - Sanger        Phred+33,  raw reads typically (0, 40)
 X - Solexa        Solexa+64, raw reads typically (-5, 40)
 I - Illumina 1.3+ Phred+64,  raw reads typically (0, 40)
 J - Illumina 1.5+ Phred+64,  raw reads typically (3, 40)
     with 0=unused, 1=unused, 2=Read Segment Quality Control Indicator (bold)
     (Note: See discussion above).
 L - Illumina 1.8+ Phred+33,  raw reads typically (0, 41)
```

http://en.wikipedia.org/wiki/FASTQ_format

Coverage Statistics

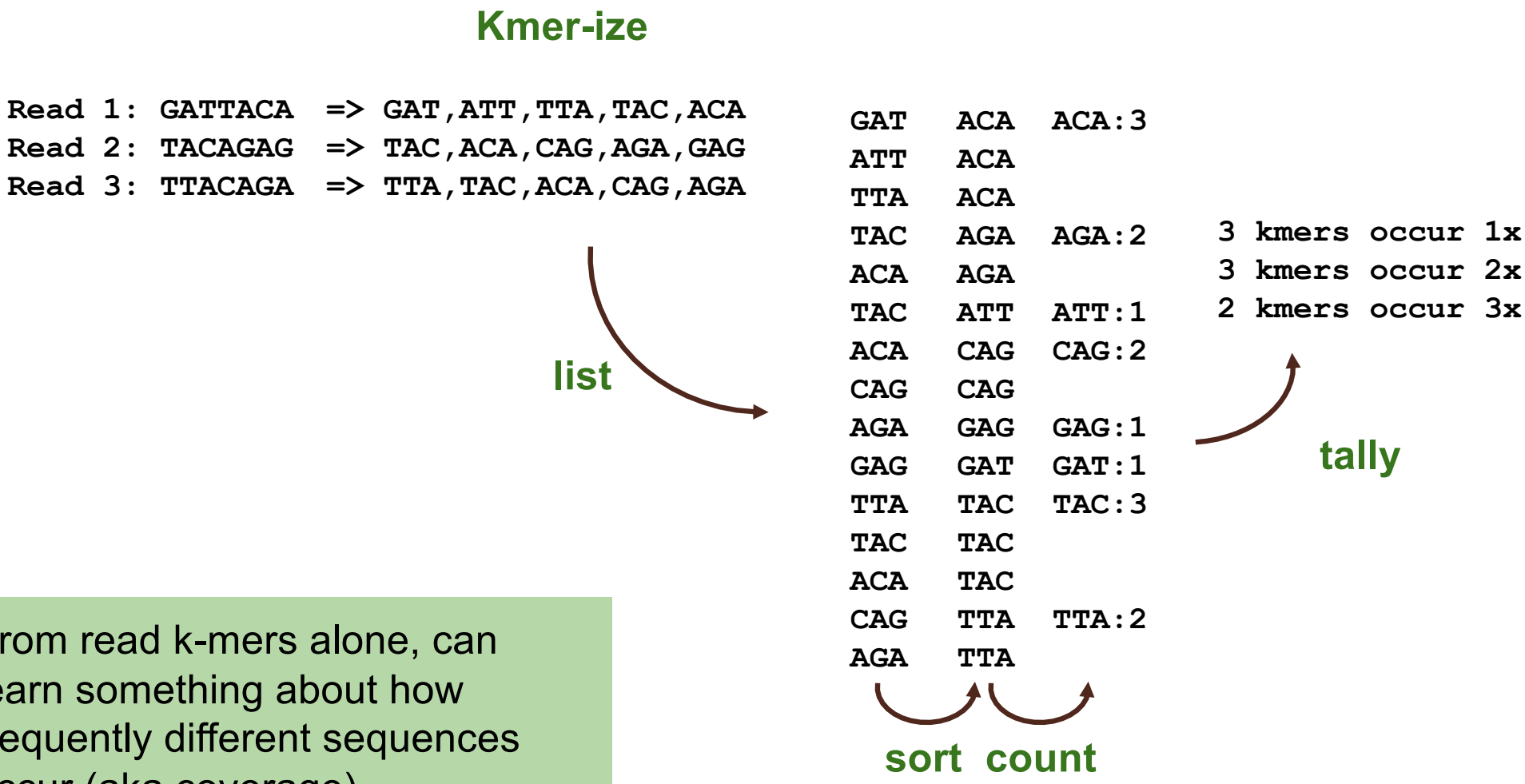
$$\text{sequencing\_coverage} = \frac{\text{total\_bases\_sequenced}}{\text{genome\_size}}$$

$$\text{genome\_size} = \frac{\text{total\_bases\_sequenced}}{\text{sequencing\_coverage}}$$

$$\text{genome\_size} = \frac{100\text{Gb}}{50\text{x}} = 2\text{Gb}$$

But how can you figure out
the coverage without a genome?

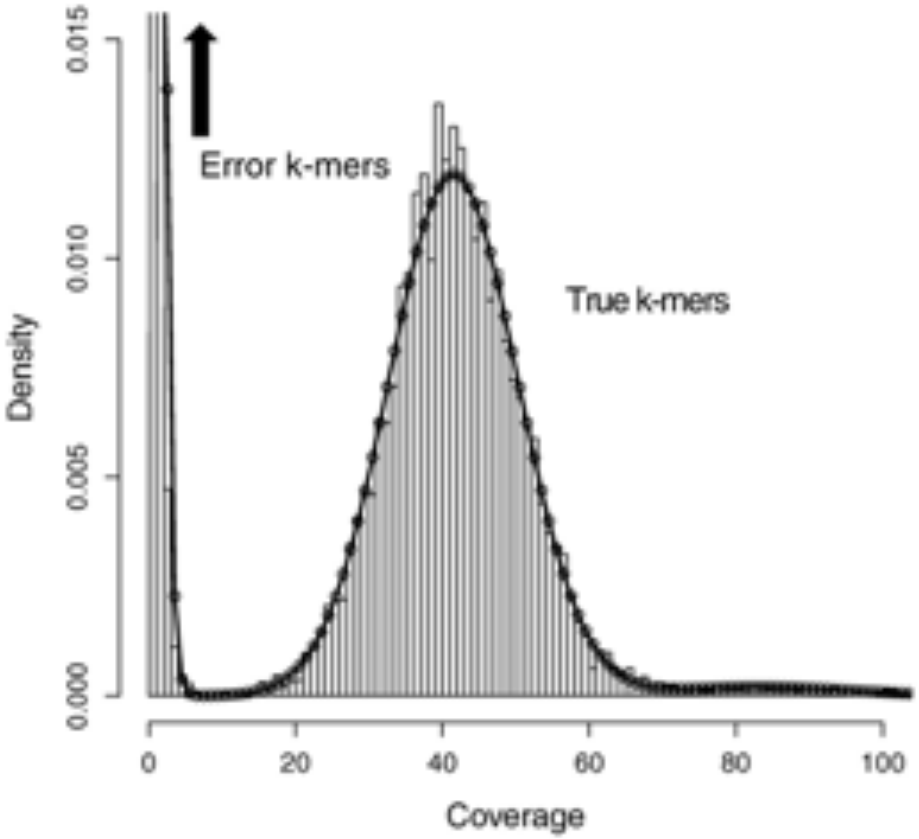
K-mer counting



From read k-mers alone, can learn something about how frequently different sequences occur (aka coverage)

Fast to compute even over huge datasets

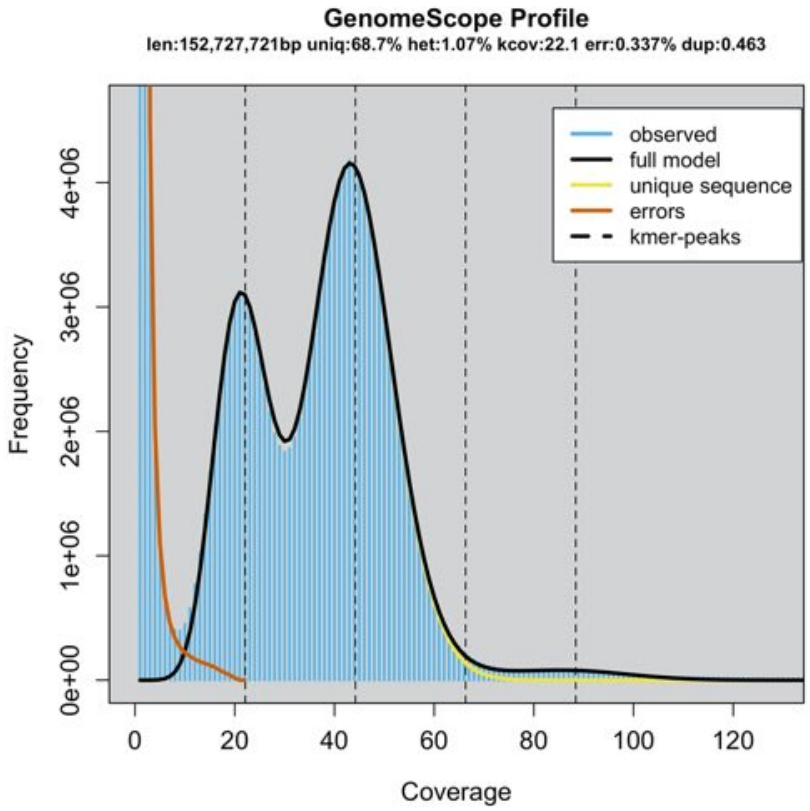
K-mer counting in real genomes

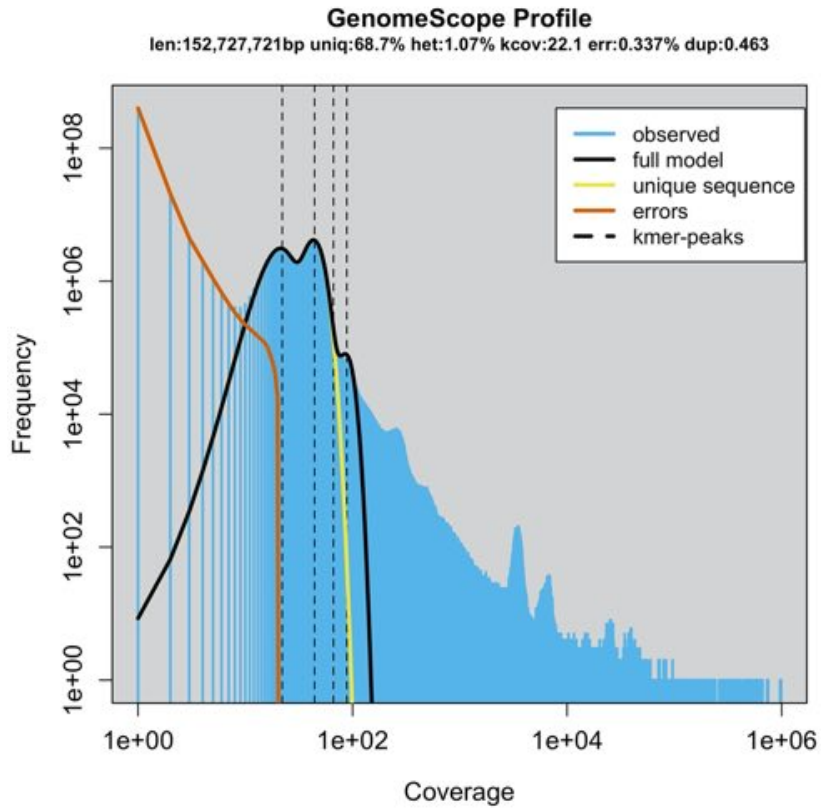


- The tally of k-mer counts in real genomes reveals the coverage distribution.
- Here we sequenced 120Gb of reads from a female human (haploid human genome size is 3Gb), and indeed we see a clear peak centered at 40x coverage
- There are also many kmers that only occur <5 times. These are from errors in the reads
- There are also kmers that occur many times (>>70 times). These are repeats in the genome

GenomeScope: Fast genome analysis from short reads

http://genomescope.org



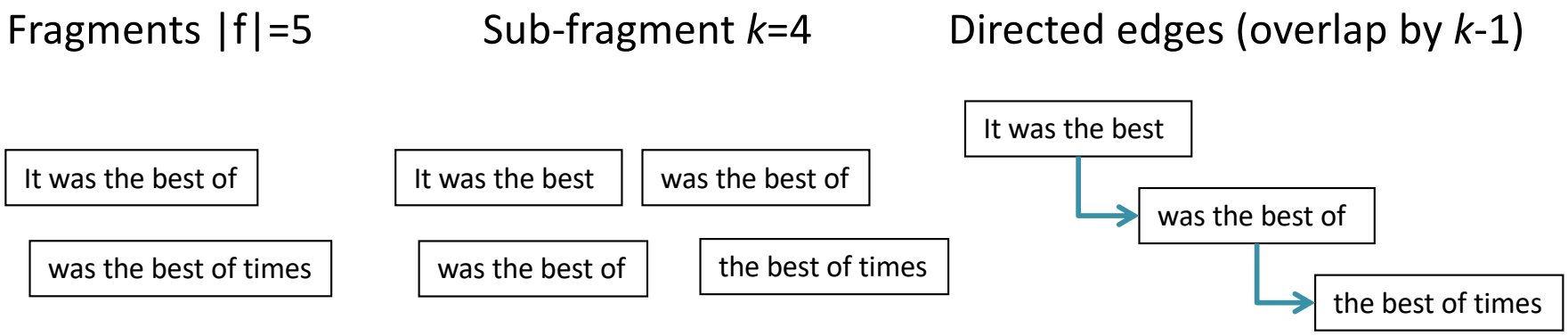


- Theoretical model agrees well with published results:
  - Rate of heterozygosity is higher than reported by other approaches but likely correct.
  - Genome size of plants inflated by organelle sequences (exclude very high freq. kmers)

Vurture, GW*, Sedlazeck FJ*, et al. (2017) *Bioinformatics*
Ranallo-Benavidez, TR. *et al.* (2020) *Nature Communication*

de Bruijn Graph Construction

- $G_k = (V,E)$
  - V = Length-*k* sub-fragments
  - E = Directed edges between consecutive sub-fragments
    - Sub-fragments overlap by *k*-1 words

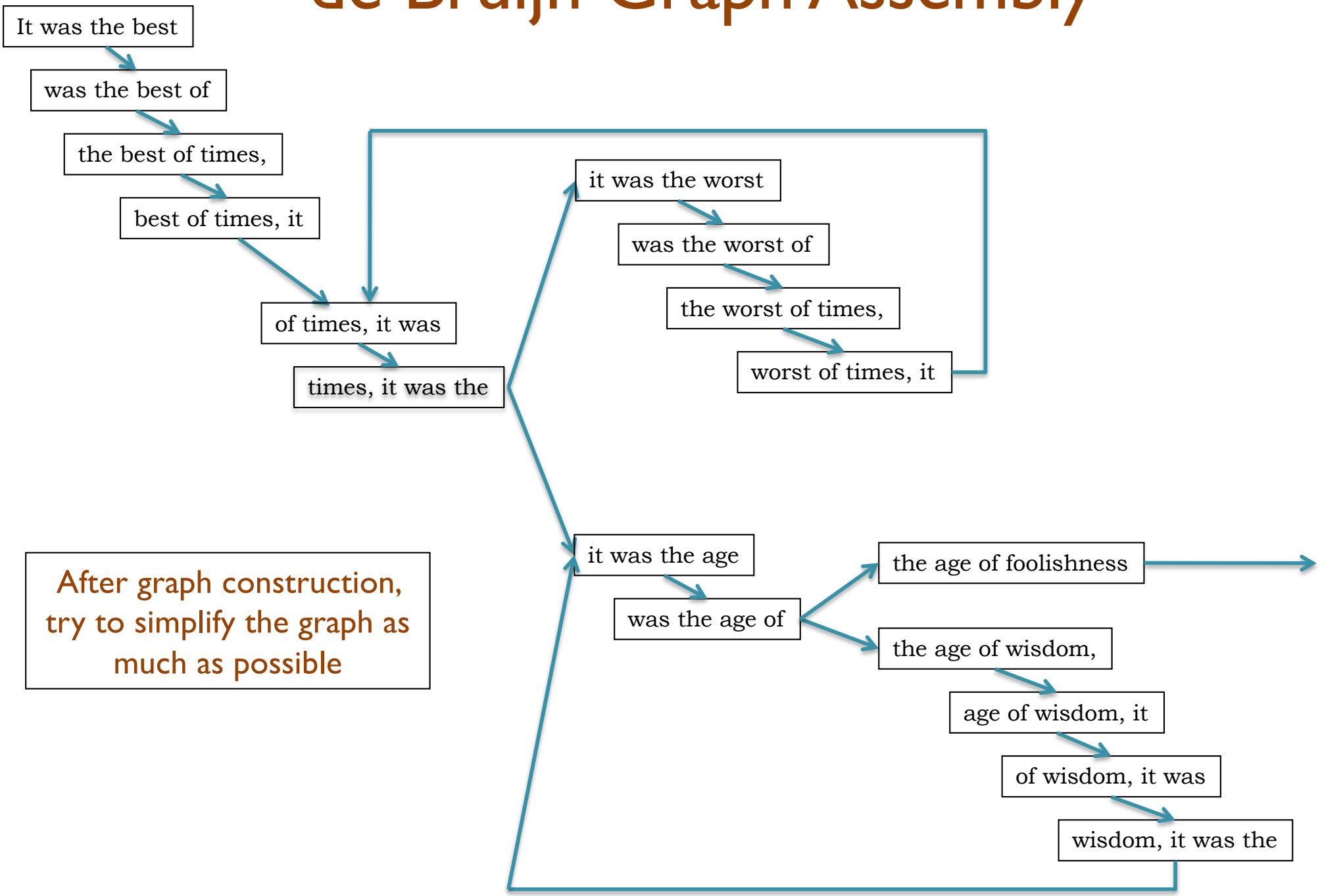


– Overlaps between fragments are implicitly computed

How to pronounce:
https://forvo.com/word/de_bruijn/

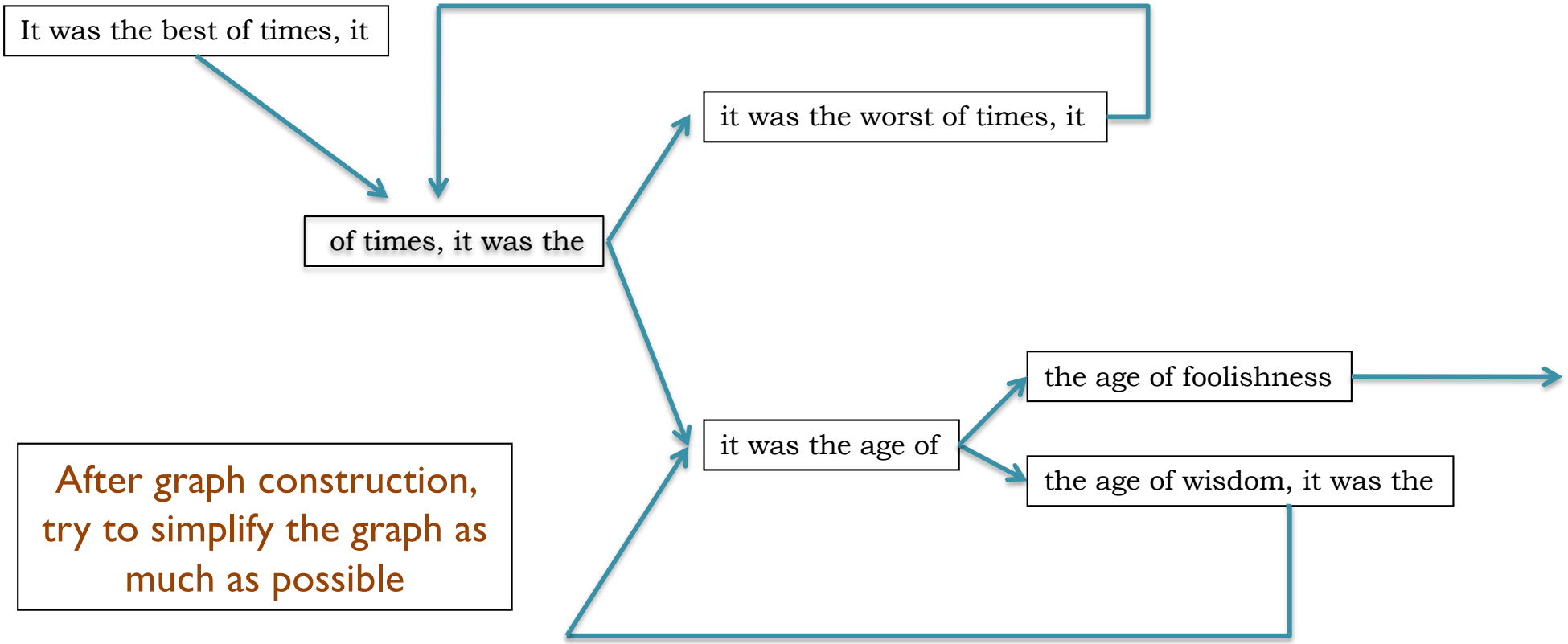
de Bruijn, 1946
Idury et al., 1995
Pevzner et al., 2001

de Bruijn Graph Assembly



After graph construction, try to simplify the graph as much as possible

de Bruijn Graph Assembly



After graph construction, try to simplify the graph as much as possible

The full tale

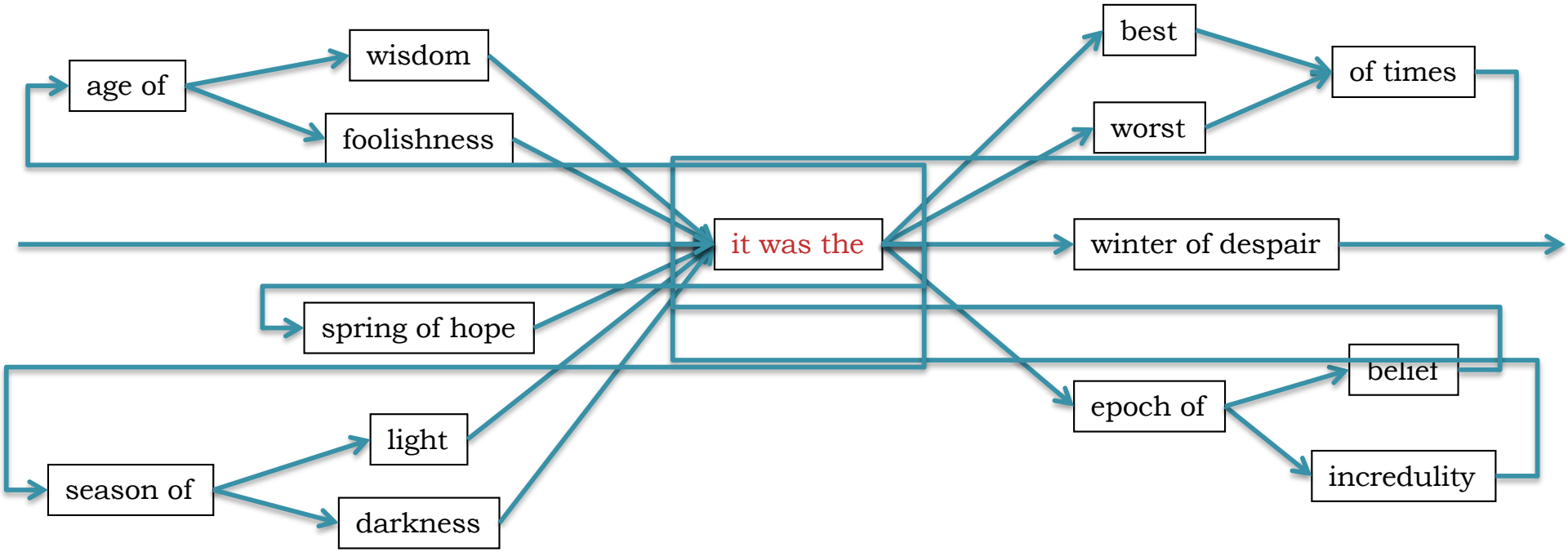
… it was the best of times it was the worst of times …

… it was the age of wisdom it was the age of foolishness …

… it was the epoch of belief it was the epoch of incredulity …

… it was the season of light it was the season of darkness …

… it was the spring of hope it was the winder of despair …

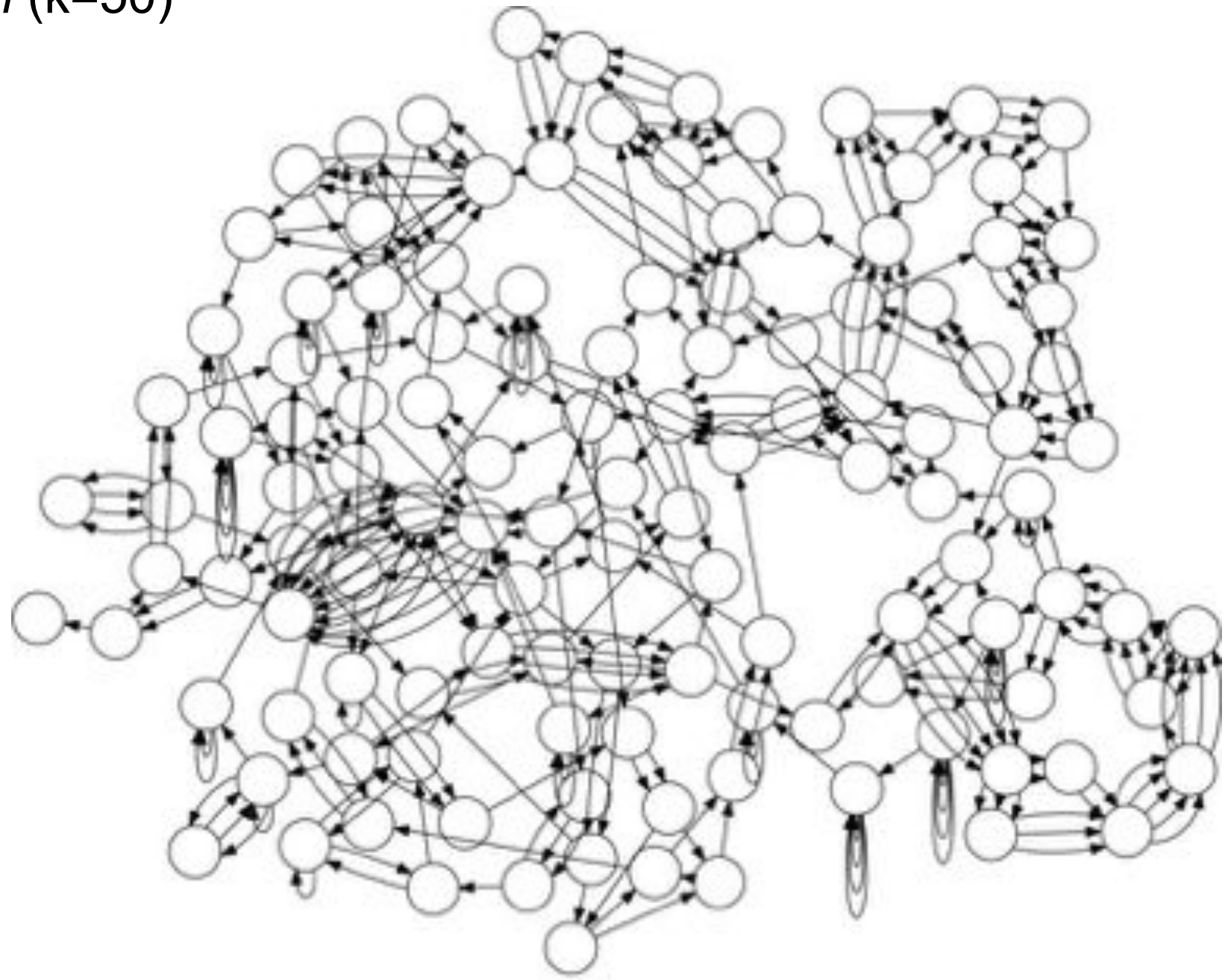


Repetitive regions

Repeat Type	Definition / Example	Prevalence
Low-complexity DNA / Microsatellites	$(b_1b_2\ldots b_k)^N$ where $1 \leq k \leq 6$ CACACACACACACACACACA	2%
SINEs (Short Interspersed Nuclear Elements)	*Alu* sequence (~280 bp) Mariner elements (~80 bp)	13%
LINEs (Long Interspersed Nuclear Elements)	~500 – 5,000 bp	21%
LTR (long terminal repeat) retrotransposons	Ty1-copia, Ty3-gypsy, Pao-BEL (~100 – 5,000 bp)	8%
Other DNA transposons		3%
Gene families & segmental duplications		4%

- Over 50% of mammalian genomes are repetitive
  - Large plant genomes tend to be even worse
  - Wheat: 16 Gbp; Pine: 24 Gbp

E. coli (k=50)



Reducing assembly complexity of microbial genomes with single-molecule sequencing
Koren et al (2013) Genome Biology. ***14***:R101 https://doi.org/10.1186/gb-2013-14-9-r101

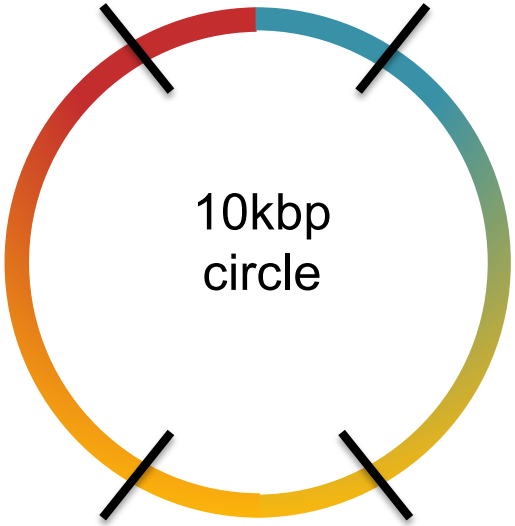
Paired-end and Mate-pairs

Paired-end sequencing

- Read one end of the molecule, flip, and read the other end
- Generate pair of reads separated by up to 500bp with inward orientation

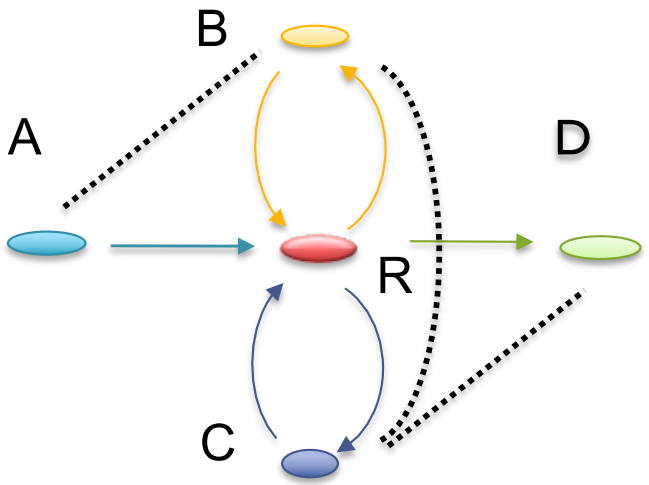
Mate-pair sequencing

- Circularize long molecules (1-10kbp), shear into fragments, & sequence
- Mate failures create short paired-end reads

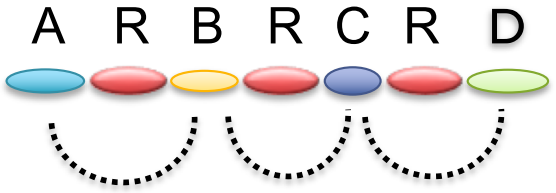


Scaffolding

- Initial contigs (*aka* unipaths, unitigs) terminate at
  - *Coverage gaps*: especially extreme GC
  - *Conflicts*: errors, repeat boundaries



- Use mate-pairs to resolve correct order through assembly graph
  - Place sequence to satisfy the mate constraints
  - Mates through repeat nodes are tangled



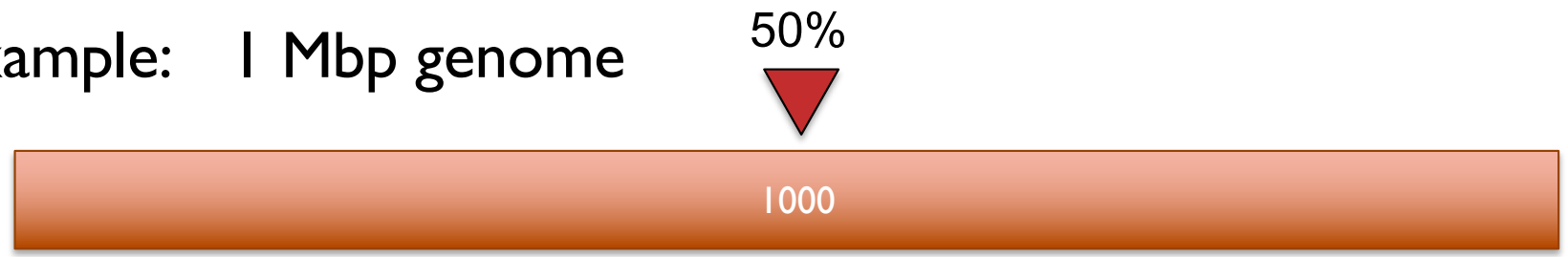
- Final scaffold may have internal gaps called sequencing gaps
  - We know the order, orientation, and spacing, but just not the bases. Fill with Ns instead

Why do scaffolds end?

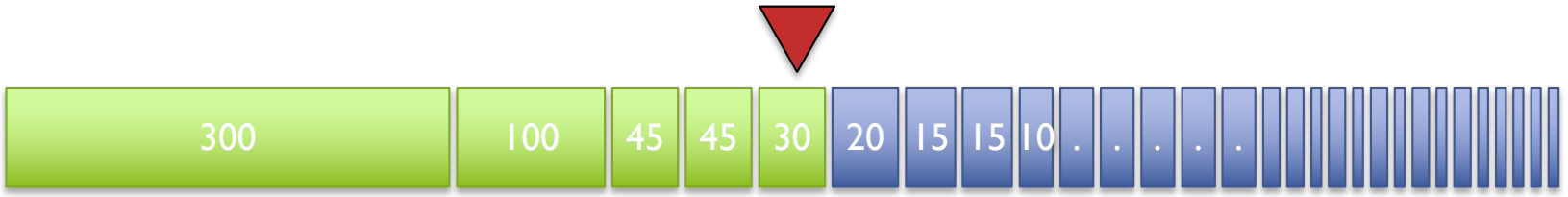
N50 Length

Def: 50% of the genome is in contigs as large as the N50 value

Example: 1 Mbp genome

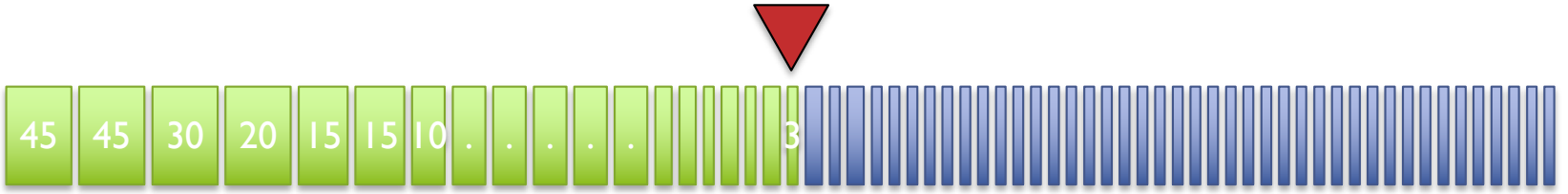


A



N50 size = 30 kbp

B



N50 size = 3 kbp

N50 Length

Def: 50% of the genome is in contigs as large as the N50 value

Better N50s improves the analysis in every dimension

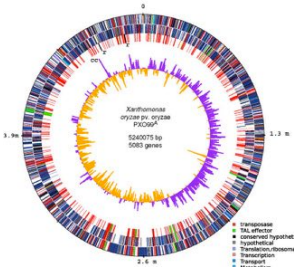
- Better resolution of genes and flanking regulatory regions
- Better resolution of transposons and other complex sequences
- Better resolution of chromosome organization
- Better sequence for all downstream analysis

Just be careful of N50 inflation!

- *A very very very bad assembler in 1 line of bash:*
- *cat *.reads.fa > genome.fa*

N50 size = 3 kbp

Assembly Summary



Assembly quality depends on

1. ***Coverage***: low coverage is mathematically hopeless
2. ***Repeat composition***: high repeat content is challenging
3. ***Read length***: longer reads help resolve repeats
4. ***Error rate***: errors reduce coverage, obscure true overlaps

- Assembly is a hierarchical, starting from individual reads, build high confidence contigs/unitigs, incorporate the mates to build scaffolds
  - Extensive error correction is the key to getting the best assembly possible from a given data set

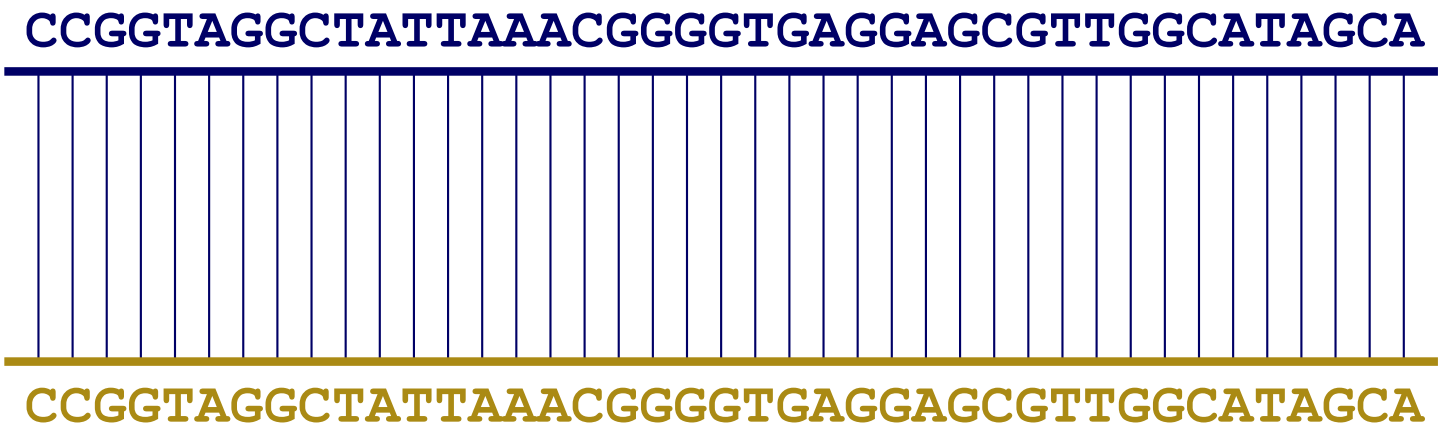
- Watch out for collapsed repeats & other misassemblies
  - Globally/Locally reassemble data from scratch with better parameters & stitch the 2 assemblies together

Whole Genome Alignment with MUMmer

Slides Courtesy of Adam M. Phillippy
NHGRI

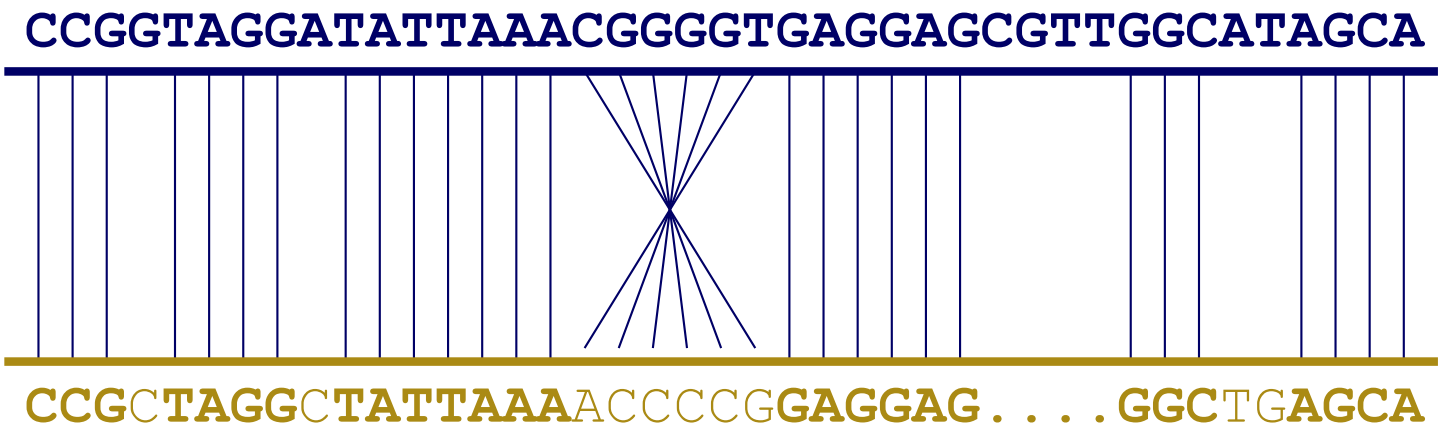
Goal of WGA

- For two genomes, *A* and *B*, find a mapping from each position in *A* to its corresponding position in *B*



Not so fast...

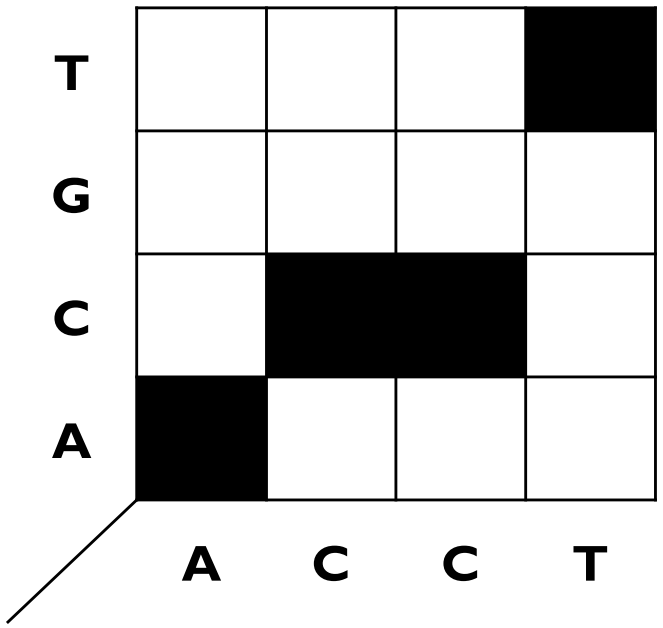
- Genome *A* may have insertions, deletions, translocations, inversions, duplications or SNPs with respect to *B* (sometimes all of the above)



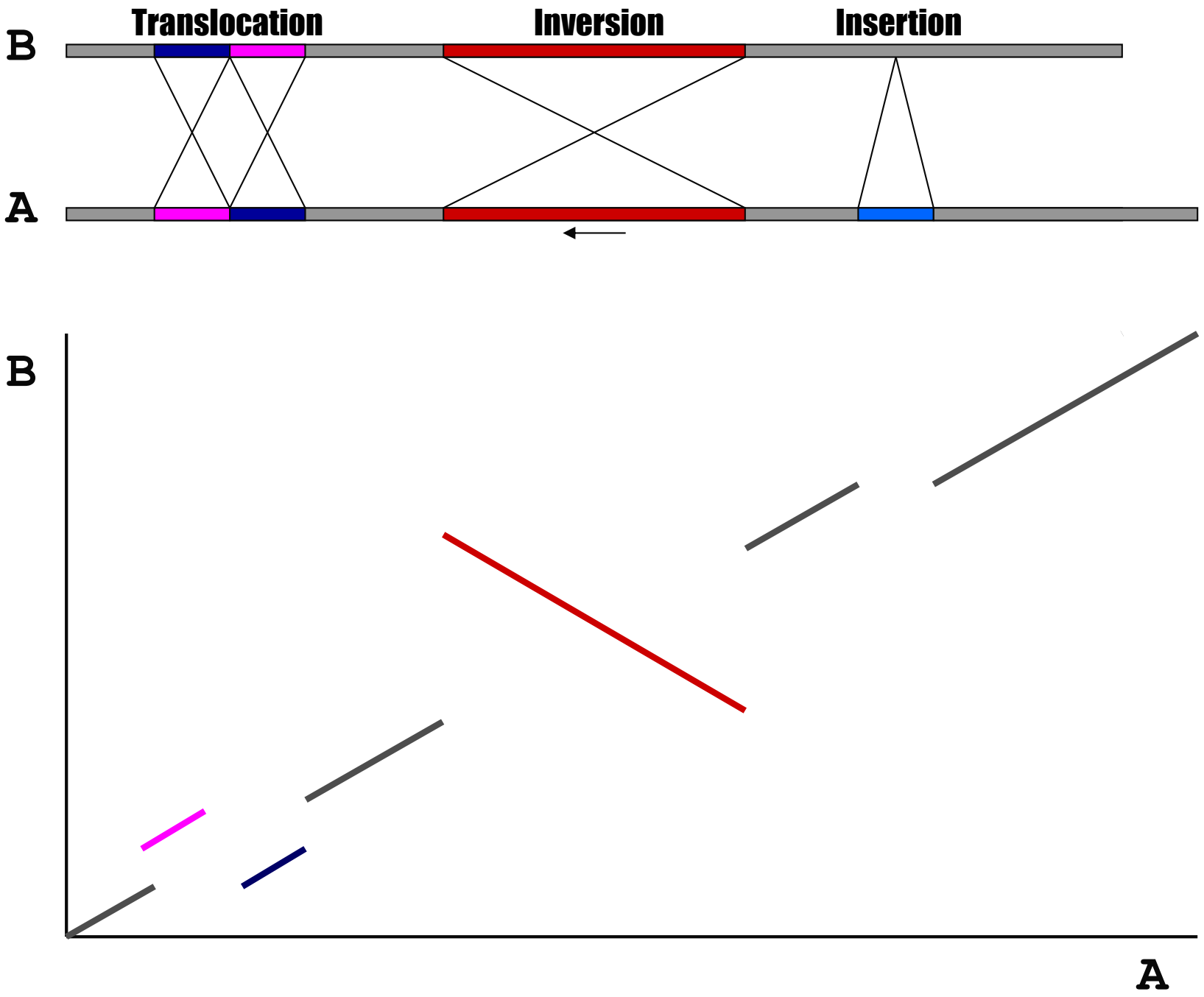
WGA visualization

- How can we visualize *whole* genome alignments?

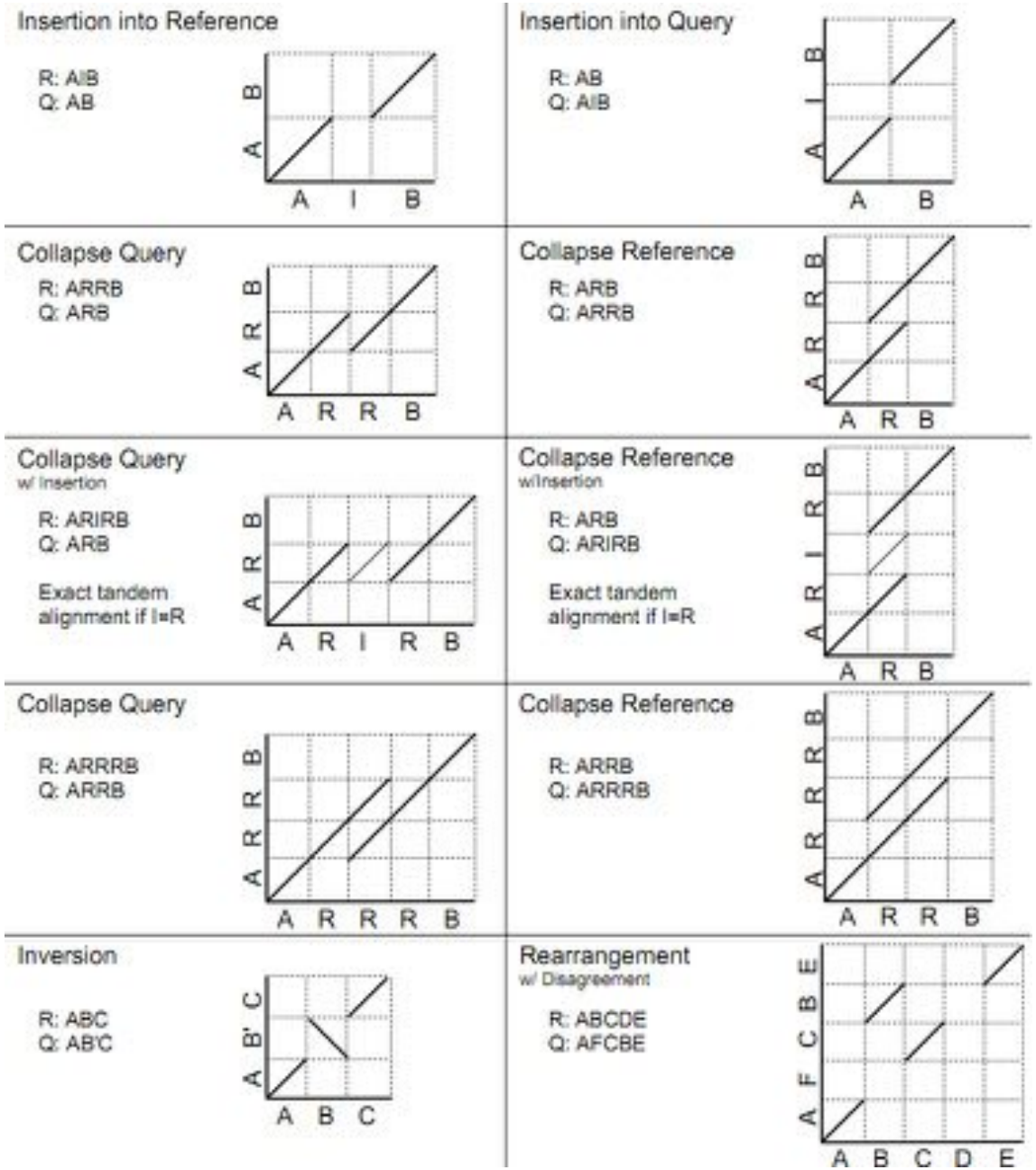
- With an alignment dot plot
  - *N* x *M* matrix
    - Let *i* = position in genome *A*
    - Let *j* = position in genome *B*
    - Fill cell *(i,j)* if A_i shows similarity to B_j



  - A perfect alignment between *A* and *B* would completely fill the positive diagonal

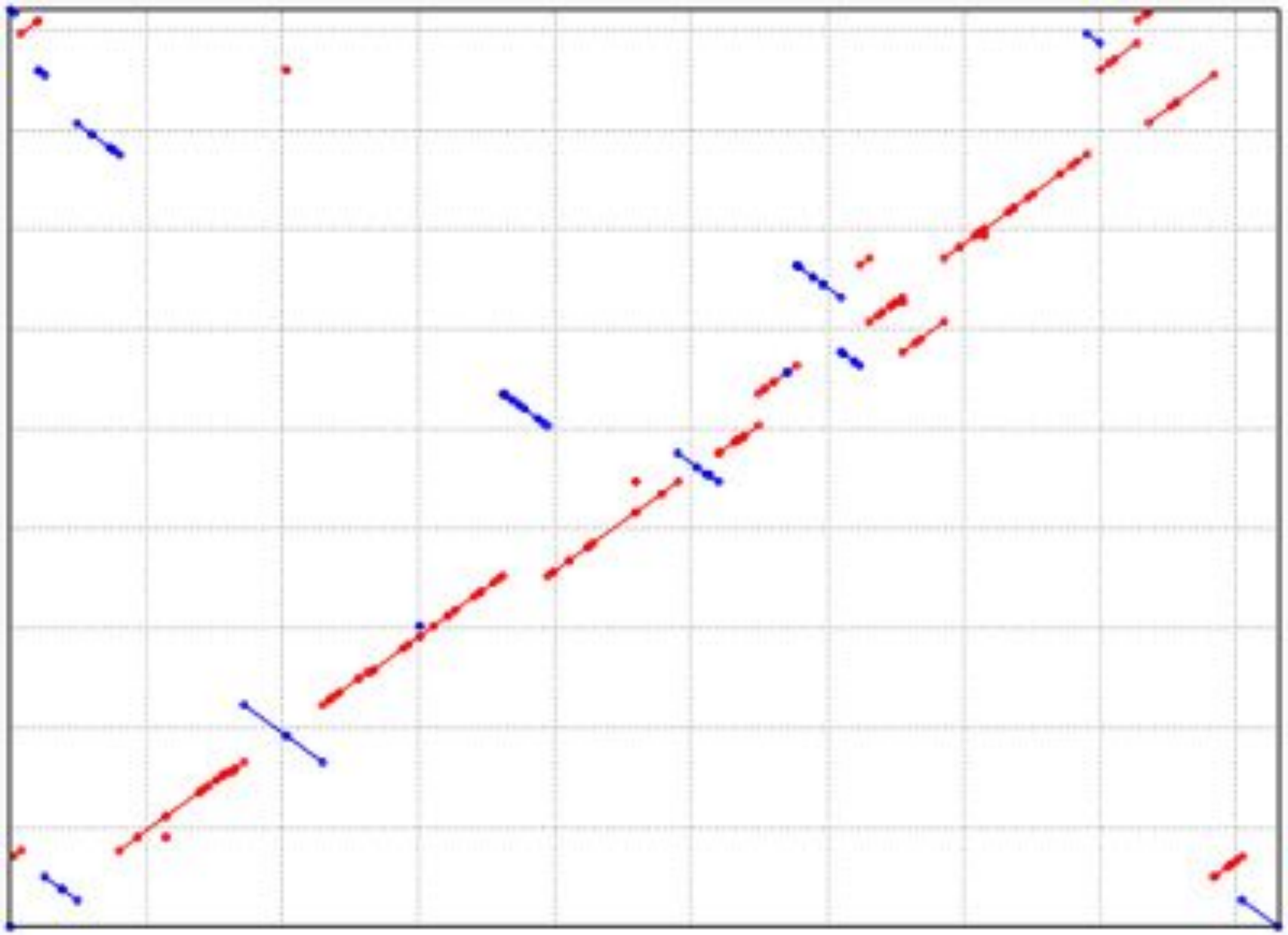

Translocation
Inversion
Insertion
B
A
B
A

SV Types



- Different structural variation types / misassemblies will be apparent by their pattern of breakpoints
- Most breakpoints will be at or near repeats
- Things quickly get complicated in real genomes

http://mummer.sf.net/manual/AlignmentTypes.pdf



Alignment of 2 strains of Y. pestis
http://mummer.sourceforge.net/manual/

Pop Quiz 1

Assemble these reads using a de Bruijn graph approach (k=3):

```
ATTA
GATT
TACA
TTAC
```

Pop Quiz 1

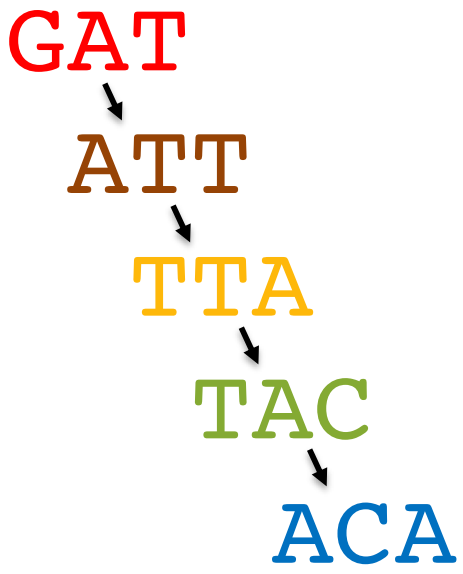
Assemble these reads using a de Bruijn graph approach (k=3):

```
ATTA: ATT -> TTA
GATT: GAT -> ATT
TACA: TAC -> ACA
TTAC: TTA -> TAC
```

Pop Quiz 1

Assemble these reads using a de Bruijn graph approach (k=3):

```
ATTA:  ATT -> TTA
GATT:  GAT -> ATT
TACA:  TAC -> ACA
TTAC:  TTA -> TAC
```



GATTACA

Pop Quiz 2

Assemble these reads using a de Bruijn graph approach (k=3):

```
ACGA
ACGT
ATAC
CGAC
CGTA
GACG
GTAT
TACG
```

Pop Quiz 2

Assemble these reads using a de Bruijn graph approach (k=3):

~~ACGA~~

ACGT

ATAC

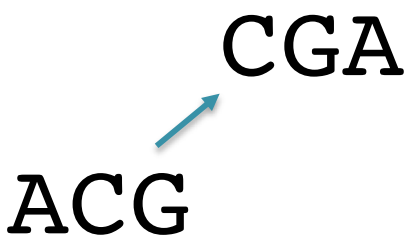
CGAC

CGTA

GACG

GTAT

TACG



Pop Quiz 2

Assemble these reads using a de Bruijn graph approach (k=3):

~~ACGA~~

~~ACGT~~

ATAC

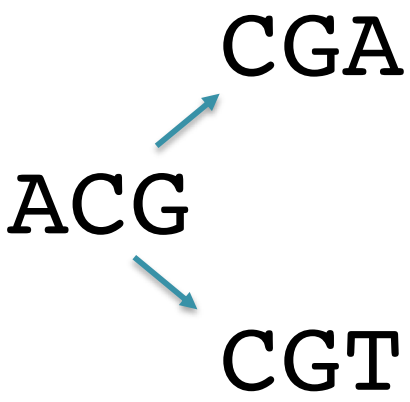
CGAC

CGTA

GACG

GTAT

TACG



Pop Quiz 2

Assemble these reads using a de Bruijn graph approach (k=3):

~~ACGA~~

~~ACGT~~

ATAC

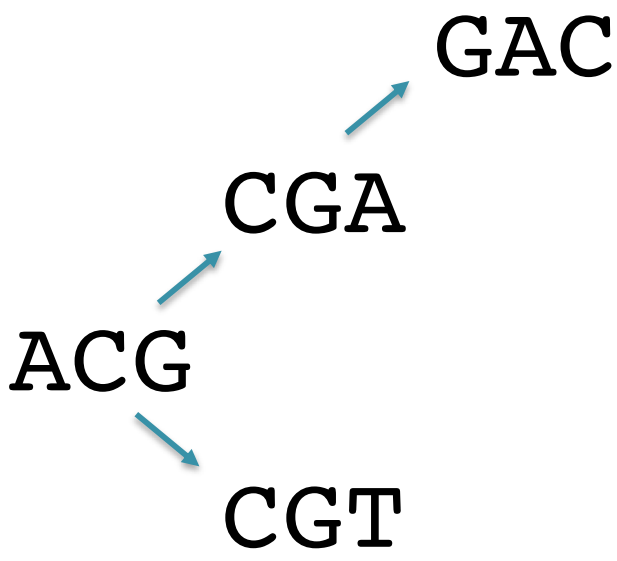
~~CGAC~~

CGTA

GACG

GTAT

TACG



Pop Quiz 2

Assemble these reads using a de Bruijn graph approach (k=3):

~~ACGA~~

~~ACGT~~

ATAC

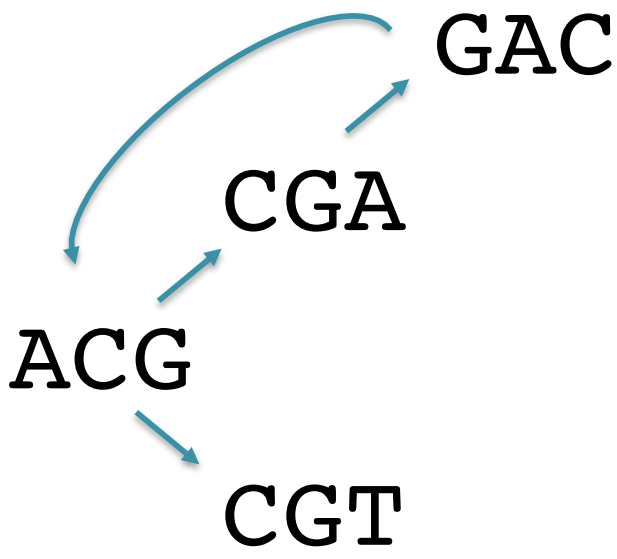
~~CGAC~~

CGTA

~~GACG~~

GTAT

TACG



Pop Quiz 2

Assemble these reads using a de Bruijn graph approach (k=3):

~~ACGA~~

~~ACGT~~

ATAC

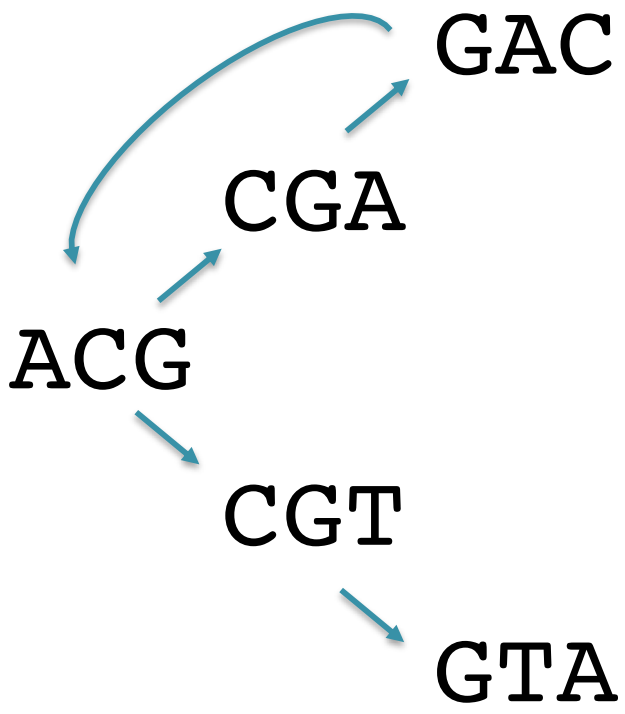
~~CGAC~~

~~CGTA~~

~~GACG~~

GTAT

TACG



Pop Quiz 2

Assemble these reads using a de Bruijn graph approach (k=3):

~~ACGA~~

~~ACGT~~

ATAC

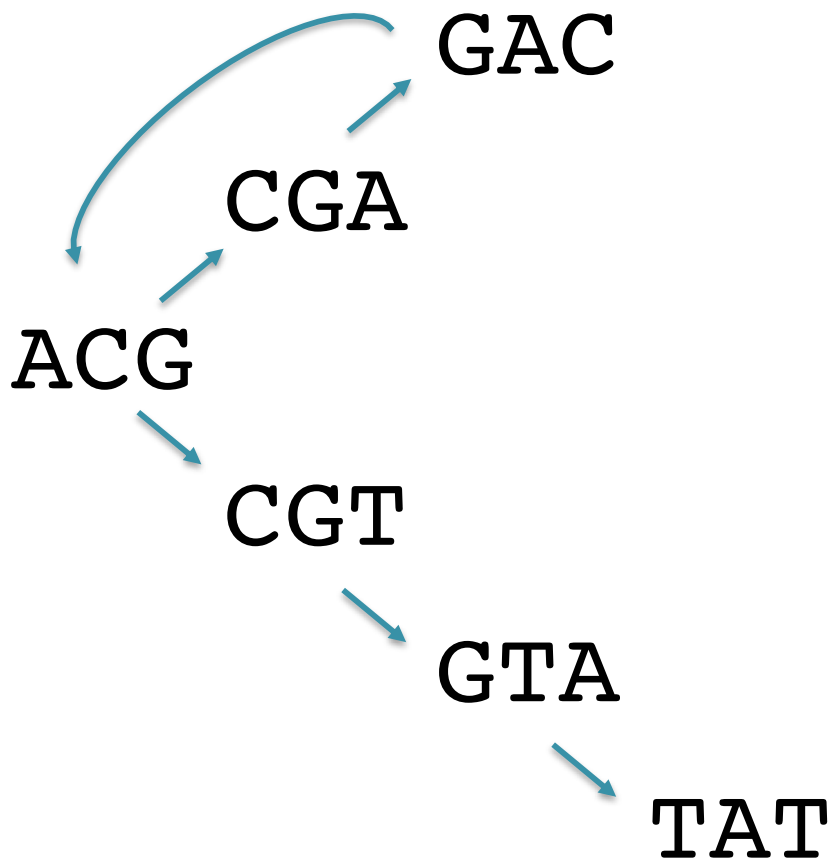
~~CGAC~~

~~CGTA~~

~~GACG~~

~~GTAT~~

TACG



Pop Quiz 2

Assemble these reads using a de Bruijn graph approach (k=3):

~~ACGA~~

~~ACGT~~

ATAC

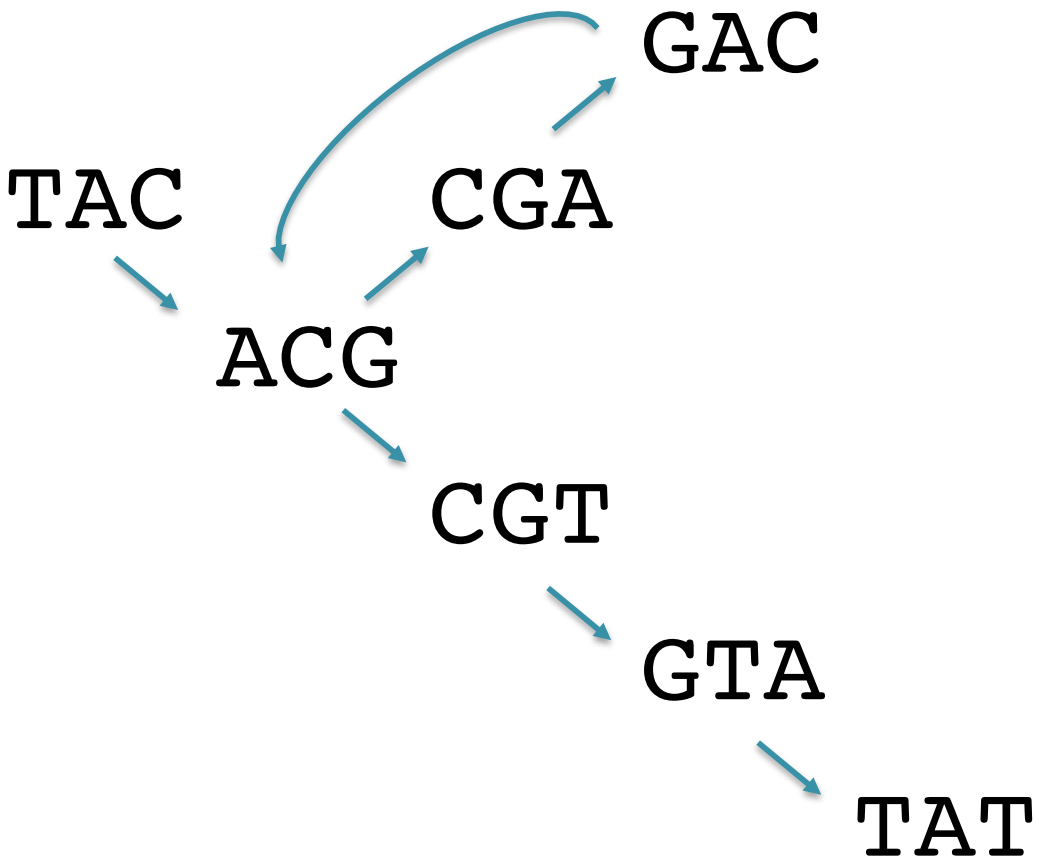
~~CGAC~~

~~CGTA~~

~~GACG~~

~~GTAT~~

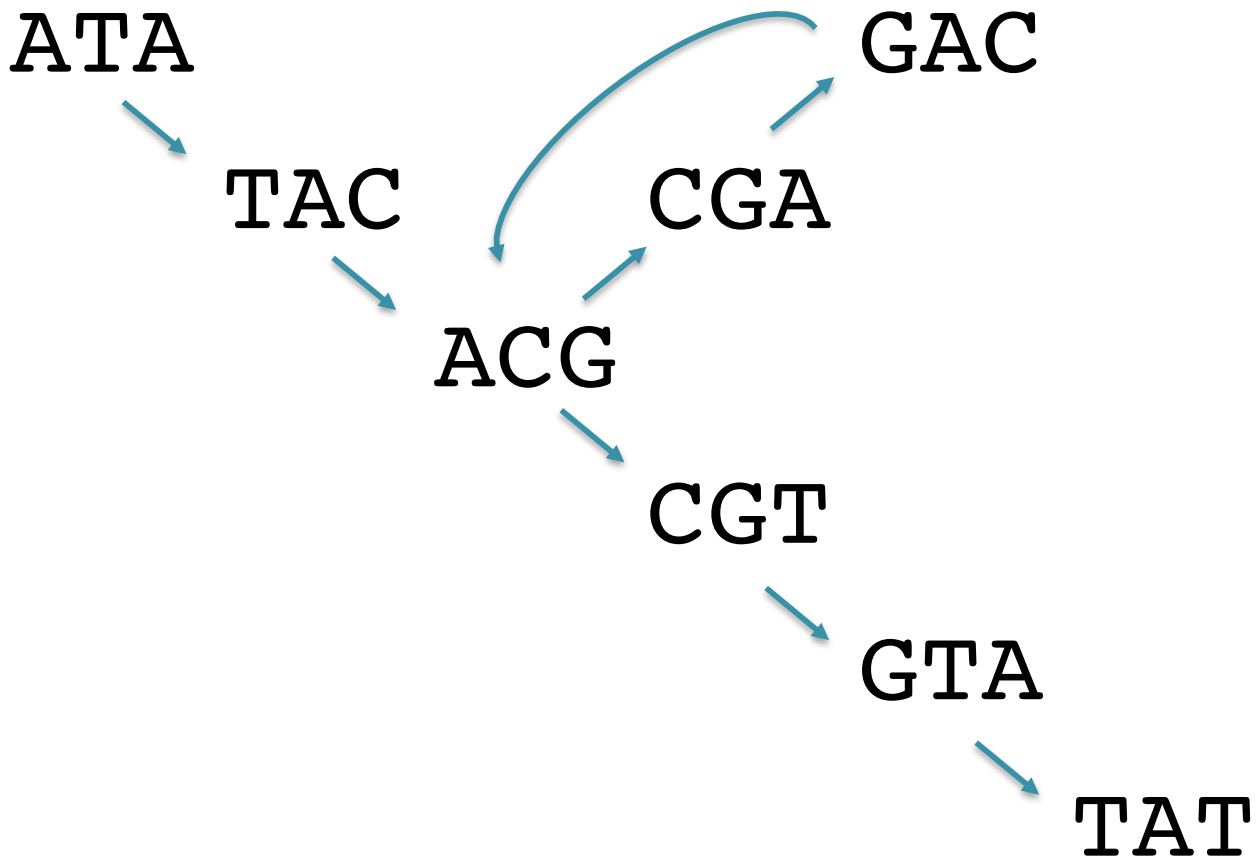
~~TACG~~



Pop Quiz 2

Assemble these reads using a de Bruijn graph approach (k=3):

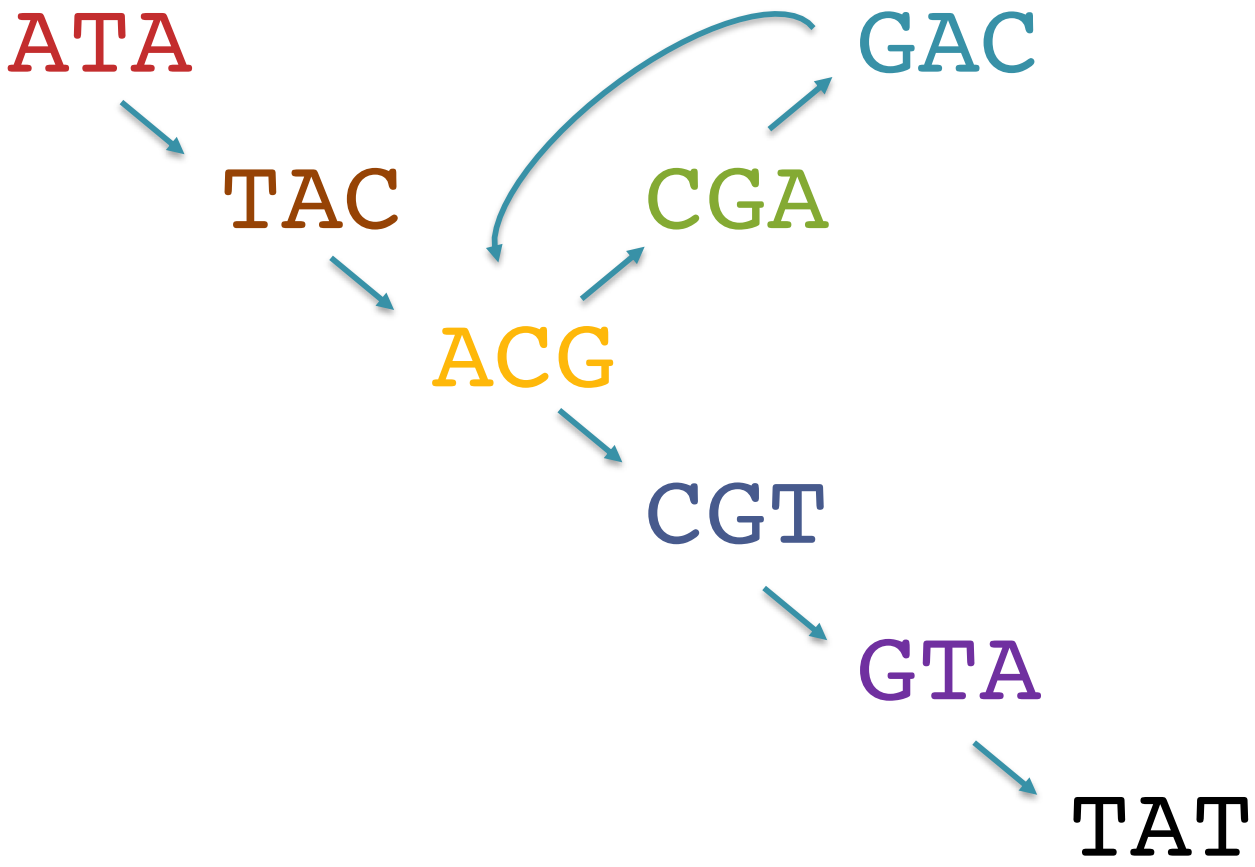
~~ACGA~~
~~ACGT~~
~~ATAC~~
~~CGAC~~
~~CGTA~~
~~GACG~~
~~GTAT~~
~~TACG~~



Pop Quiz 2

Assemble these reads using a de Bruijn graph approach (k=3):

~~ACGA~~
~~ACGT~~
~~ATAC~~
~~CGAC~~
~~CGTA~~
~~GACG~~
~~GTAT~~
~~TACG~~

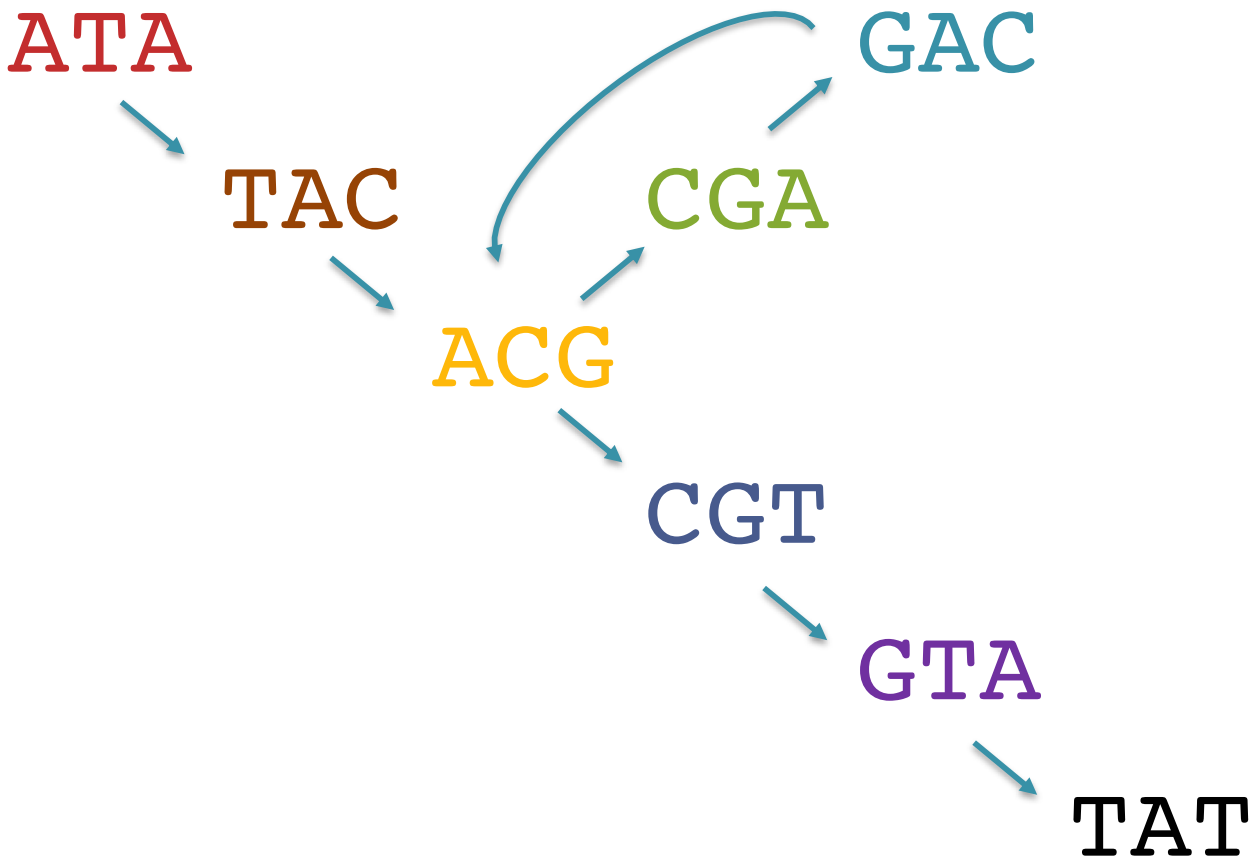


ATACGACGTAT

Pop Quiz 2

Assemble these reads using a de Bruijn graph approach (k=3):

~~ACGA~~
~~ACGT~~
~~ATAC~~
~~CGAC~~
~~CGTA~~
~~GACG~~
~~GTAT~~
~~TACG~~



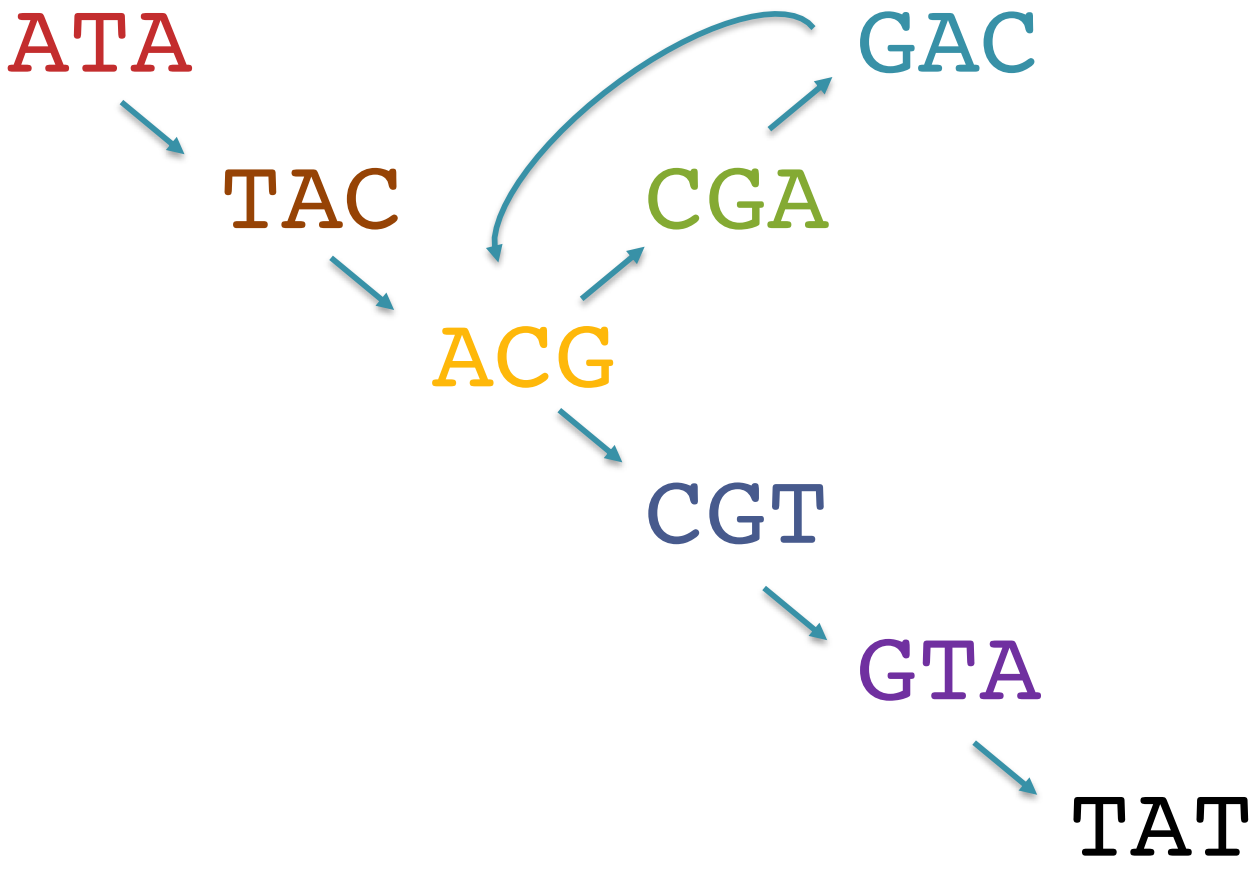
Whats another possible genome?

ATACGACGTAT

Pop Quiz 2

Assemble these reads using a de Bruijn graph approach (k=3):

~~ACGA~~
~~ACGT~~
~~ATAC~~
~~CGAC~~
~~CGTA~~
~~GACG~~
~~GTAT~~
~~TACG~~



Should we add the edge TAT -> ATA?

ATACGACGTAT