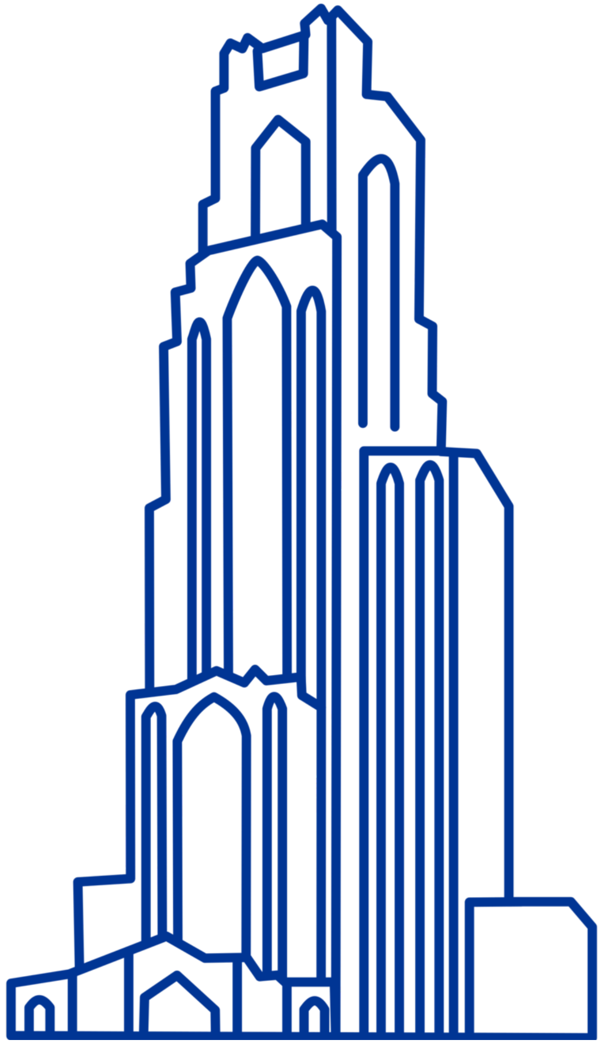


Computational Biology

(BIOSC 1540)

Lecture 05:

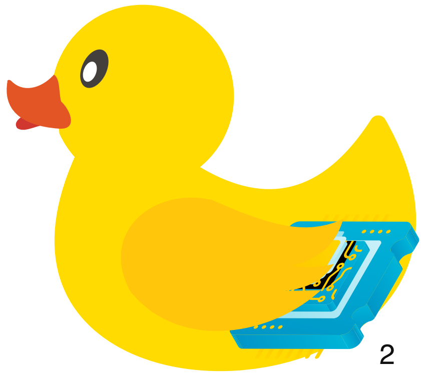
Gene annotation

Sep 10, 2024

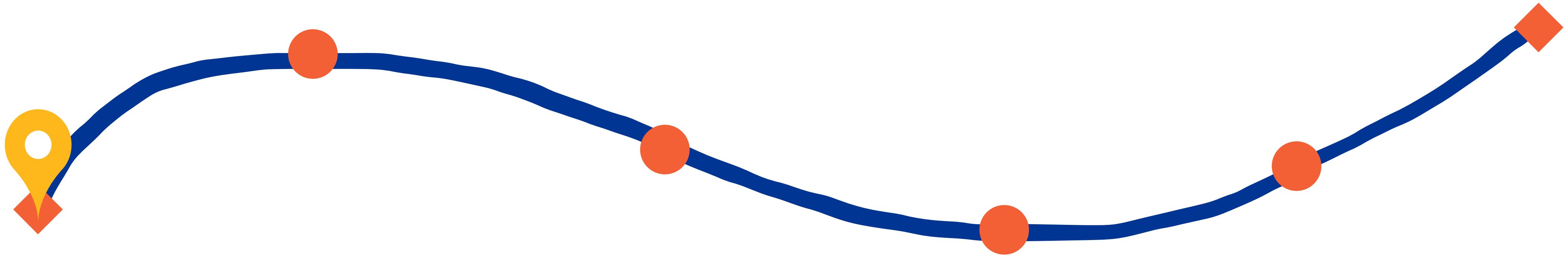
University of Pittsburgh

Announcements

- A01 solutions are posted
    - Grading will take me a hot minute
- A02 is due Thursday at 11:59 pm
- **Programming+** problems will be posted each homework and are completely optional



After today, you should be able to



1. **Explain the graph traversal and contig extraction process in genome assemblers.**
2. Understand key output files and quality metrics of genome assembly results.
3. Define gene annotation and describe its key components.
4. Outline the main computational methods used in gene prediction and annotation.
5. Analyze and interpret basic gene annotation data and outputs.

Walking along the graph produces strings

$$\begin{tikzcd}
CG \arrow[r] & GT \arrow[r] & TA \arrow[r] & AA \arrow[r] \arrow[loop below] & AT
\end{tikzcd}$$

Results in **CGTAAAT**

$$\begin{tikzcd}
 & & & & & CGA \arrow[r, "1"] & GAA & & & \\
AAT \arrow[r, "1"] & ATG \arrow[r, "1"] & TGG \arrow[r, "1"] & GGC \arrow[r, "2"] & GCG \arrow[r] \arrow[ur, "1"] & CGT \arrow[r, "2"] & GTA \arrow[r, "1"] & TAA \arrow[r, "2"] & AAA \arrow[r, "2"] & AAG \arrow[d, "1"] \\
 & & & & & & & & & AGG \arrow[llllllu, bend left, "1"]
\end{tikzcd}$$

Results in: **AATGGCGTAAAGGCGAA**

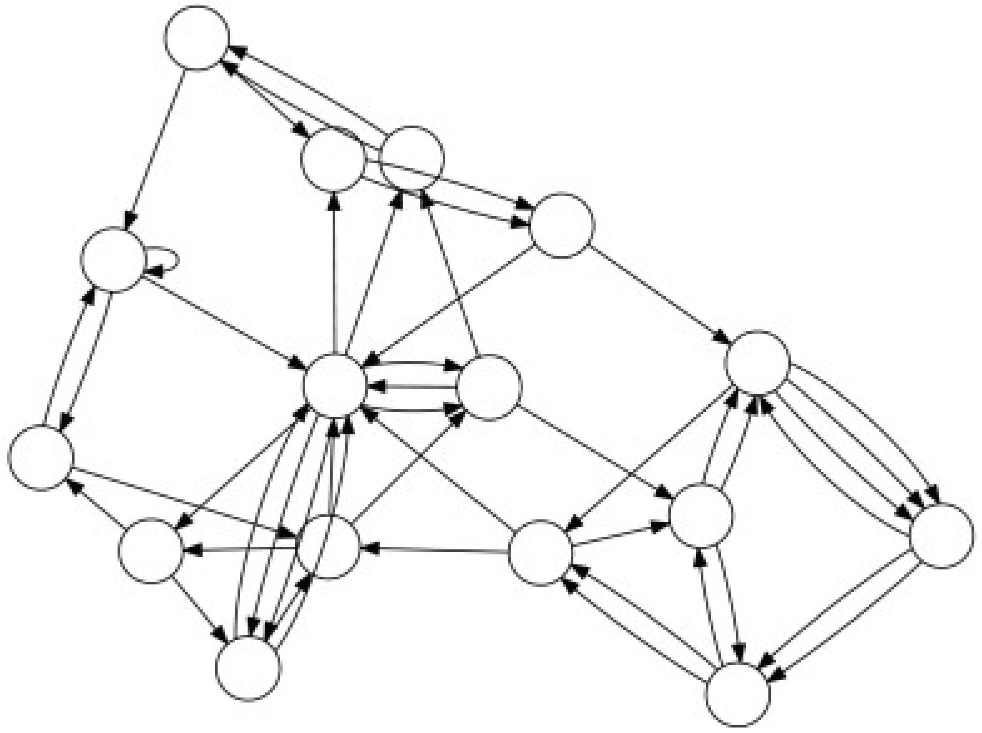
Graphs in practice are not this easy

Graph traversal algorithms are used to extract contigs

General overview

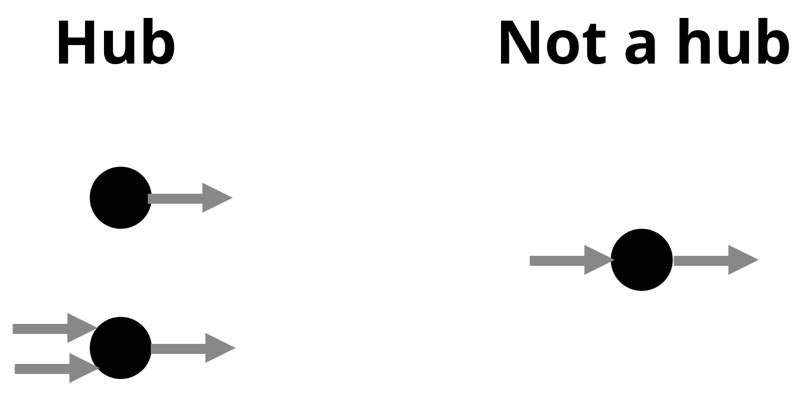
- Select a start node
- Walk along the graph until a dead end or previously visited node is reached
- Backtrack and explore alternative paths
- Repeat for remaining unvisited nodes

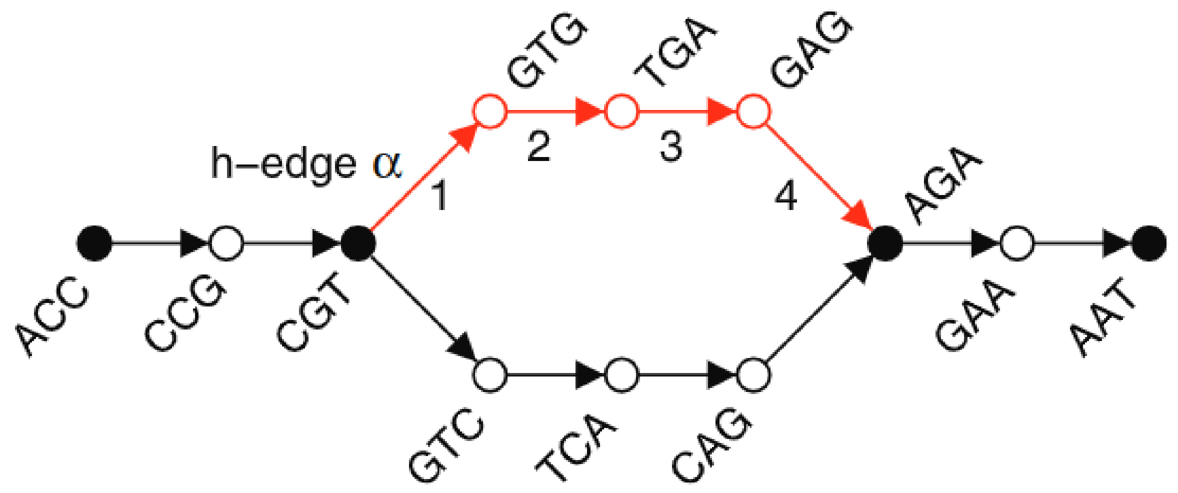
Multiple approaches are used and comes down to personal preference



How do we select a starting node?

Hubs: Indegree and outdegree != 1





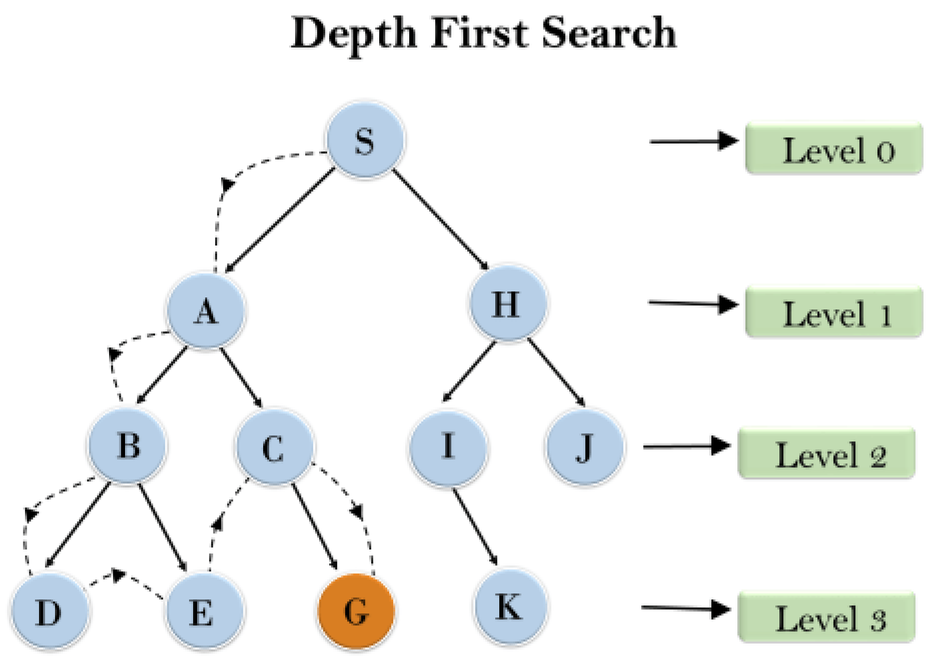
High coverage: Suggests that the node is likely a true sequence rather than an error

Hubs are shown as filled-in nodes

Depth-first search explores graph for potential paths (i.e., contigs)

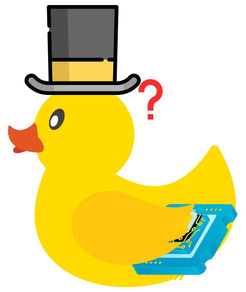
How do you choose a walk?

1. Start at a chosen vertex (node).
2. Mark the **current vertex as visited**
3. **Explore an adjacent unvisited vertex**
4. If no unvisited adjacent vertices exist, **backtrack** to the last vertex with unvisited adjacent vertices.
5. **Repeat** steps 2-4 until all reachable vertices have been visited.



How do we choose the "best" path for our contig?

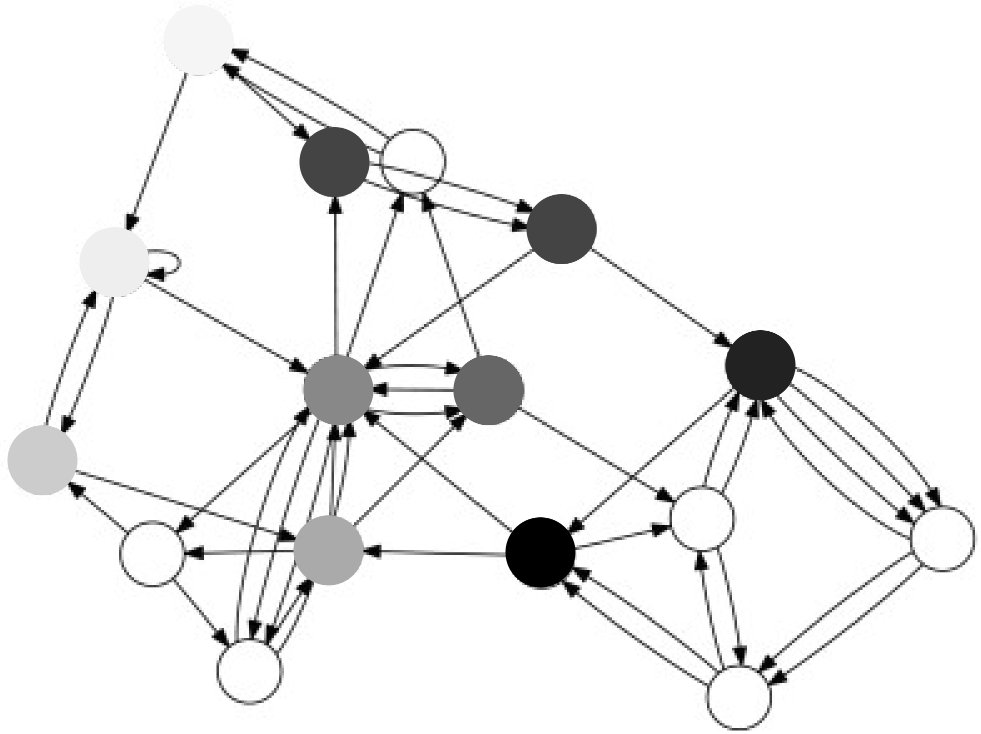
What factors would you look for? **Talk to your neighbors**



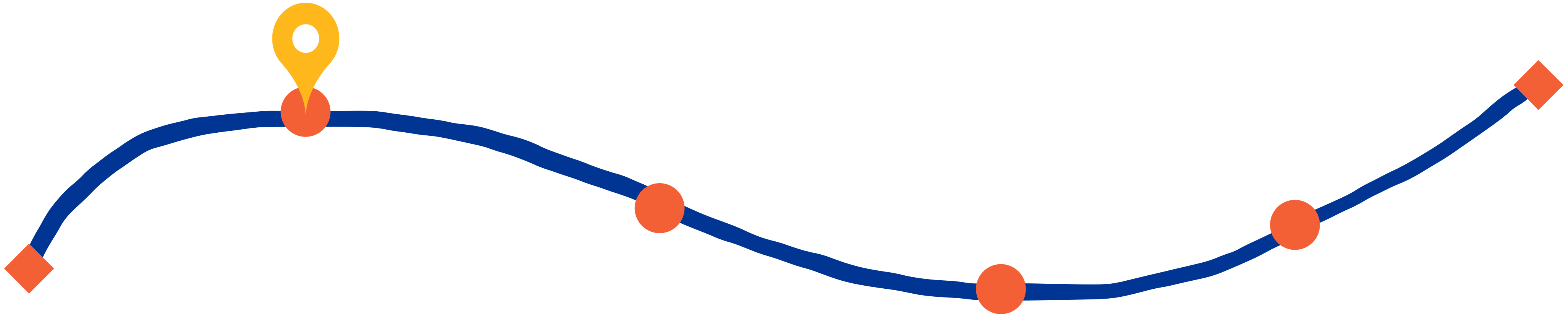
Long paths are desired but not always reliable due to potential repeats

High, consistent read coverage

Unique, non-branching paths



After today, you should be able to

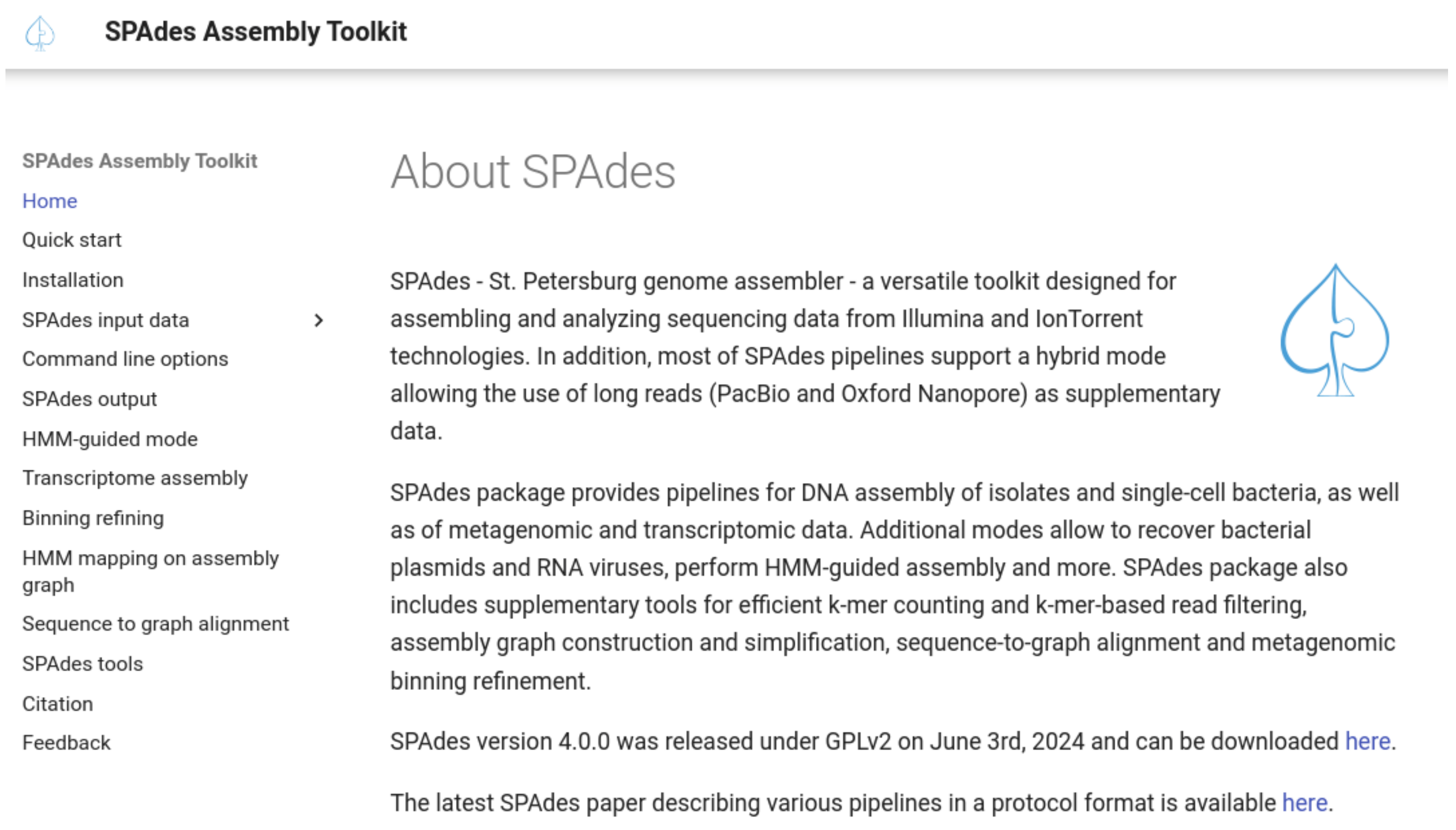


1. Explain the graph traversal and contig extraction process in genome assemblers.
2. **Understand key output files and quality metrics of genome assembly results.**
3. Define gene annotation and describe its key components.
4. Outline the main computational methods used in gene prediction and annotation.
5. Analyze and interpret basic gene annotation data and outputs.

Let's get practical with SPAdes

SPAdes is a popular prokaryote genome assembler

Based on De Bruijn graphs with numerous improvements



SPAdes Assembly Toolkit

About SPAdes

SPAdes - St. Petersburg genome assembler - a versatile toolkit designed for assembling and analyzing sequencing data from Illumina and IonTorrent technologies. In addition, most of SPAdes pipelines support a hybrid mode allowing the use of long reads (PacBio and Oxford Nanopore) as supplementary data.

SPAdes package provides pipelines for DNA assembly of isolates and single-cell bacteria, as well as of metagenomic and transcriptomic data. Additional modes allow to recover bacterial plasmids and RNA viruses, perform HMM-guided assembly and more. SPAdes package also includes supplementary tools for efficient k-mer counting and k-mer-based read filtering, assembly graph construction and simplification, sequence-to-graph alignment and metagenomic binning refinement.

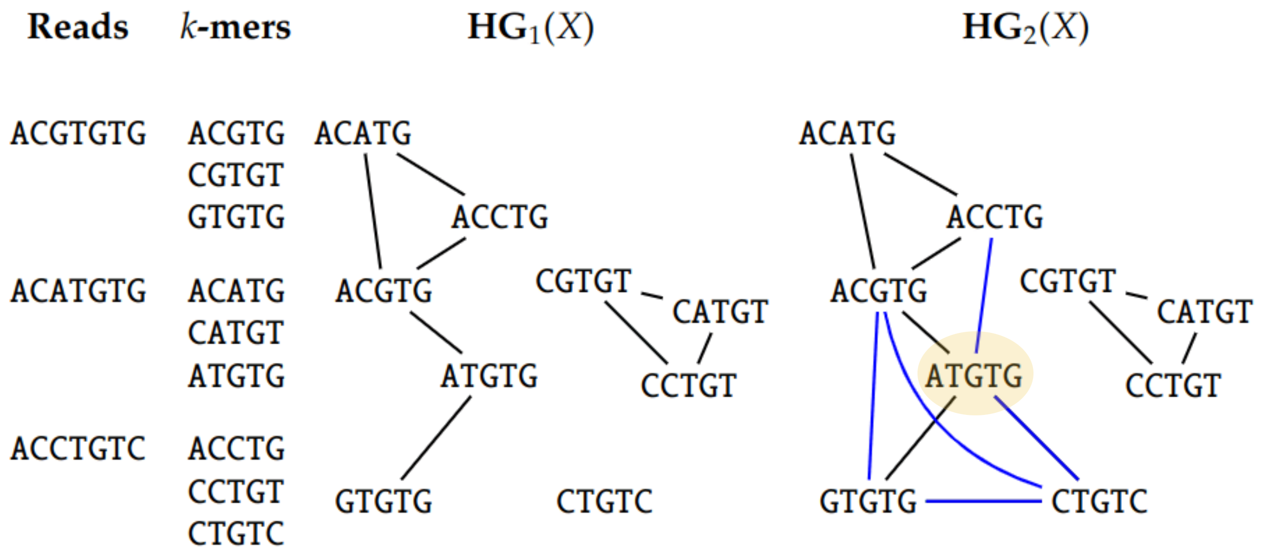
SPAdes version 4.0.0 was released under GPLv2 on June 3rd, 2024 and can be downloaded here.

The latest SPAdes paper describing various pipelines in a protocol format is available here.

GitHub

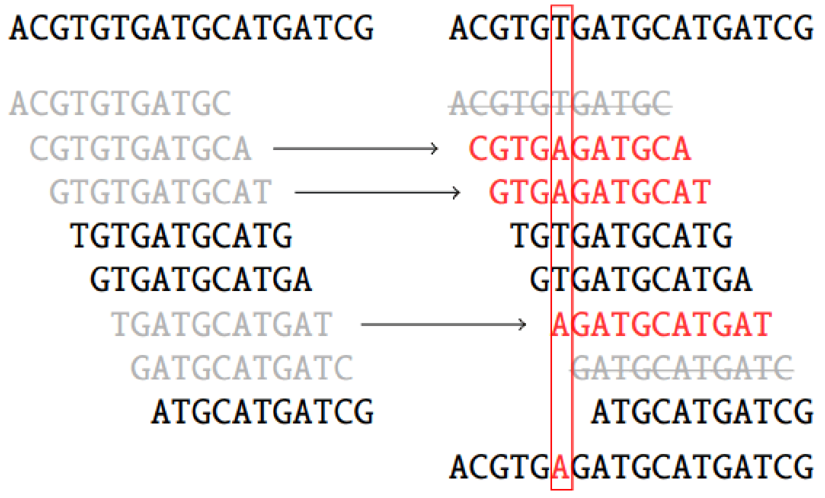
Error correction with BayesHamming

Build Hamming graphs for k-mers



Undirected edges for Hamming distance of n nucleotide differences

Identify strong k-mers based on clustering (i.e., high similarity)

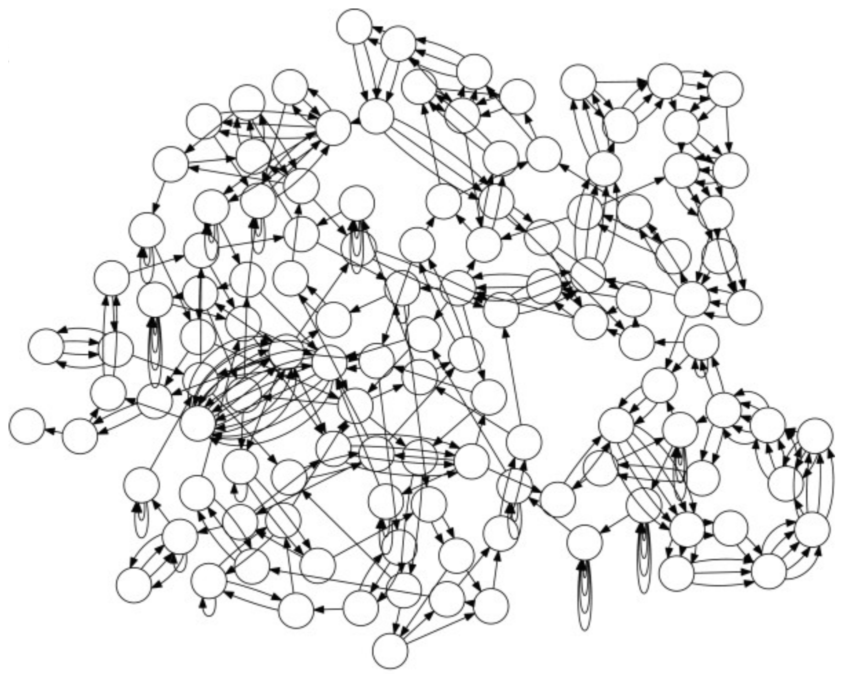


Estimate read error based on base qualitites

Builds multisized graphs with different k's

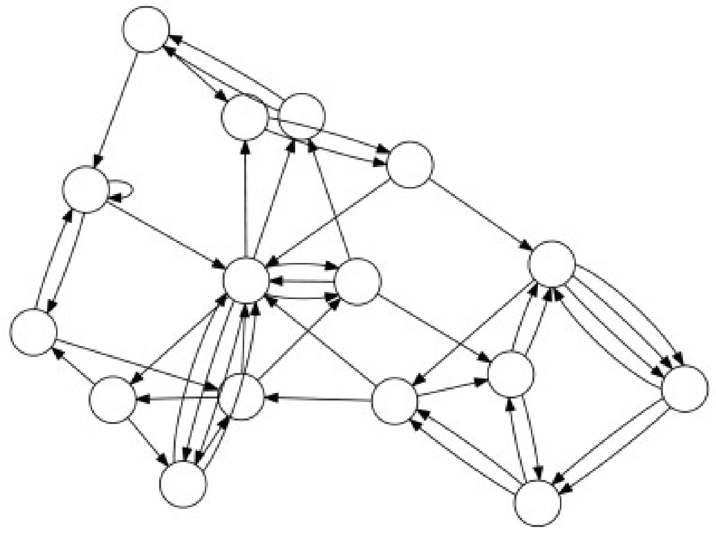
By using multiple graphs, SPAdes can
better handle variable coverage

Large k



Leads to **fragmented graphs** and
helps reduce repeat collapsing

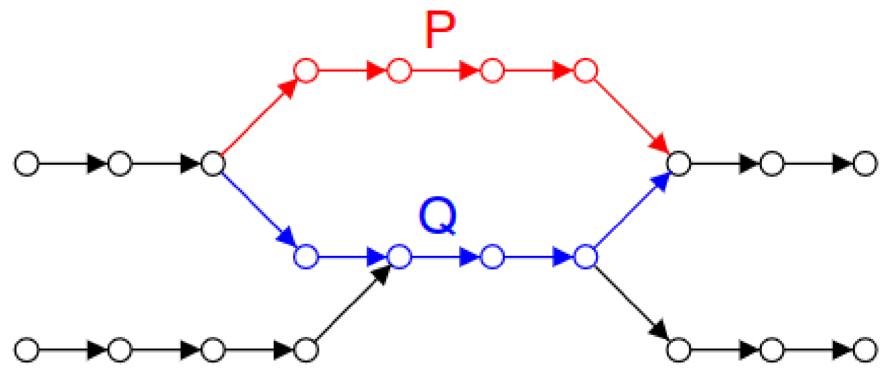
Small k



Collapsed, tangled graphs great for
low-coverage regions

Graph simplification and correction

Potential bulge

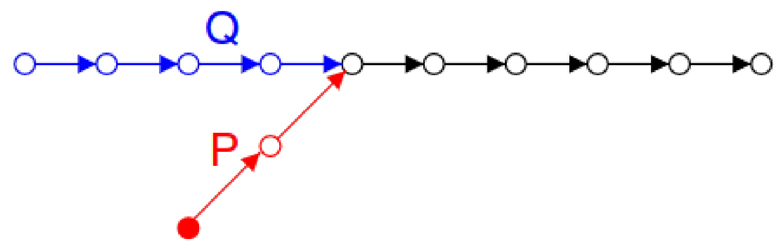


Removal of a bulge will quickly deteriorate the graph and lose read information

If P needs to be removed, we "project" the information (e.g., coverage) onto Q

P's edges are then removed in the process

Potential tips

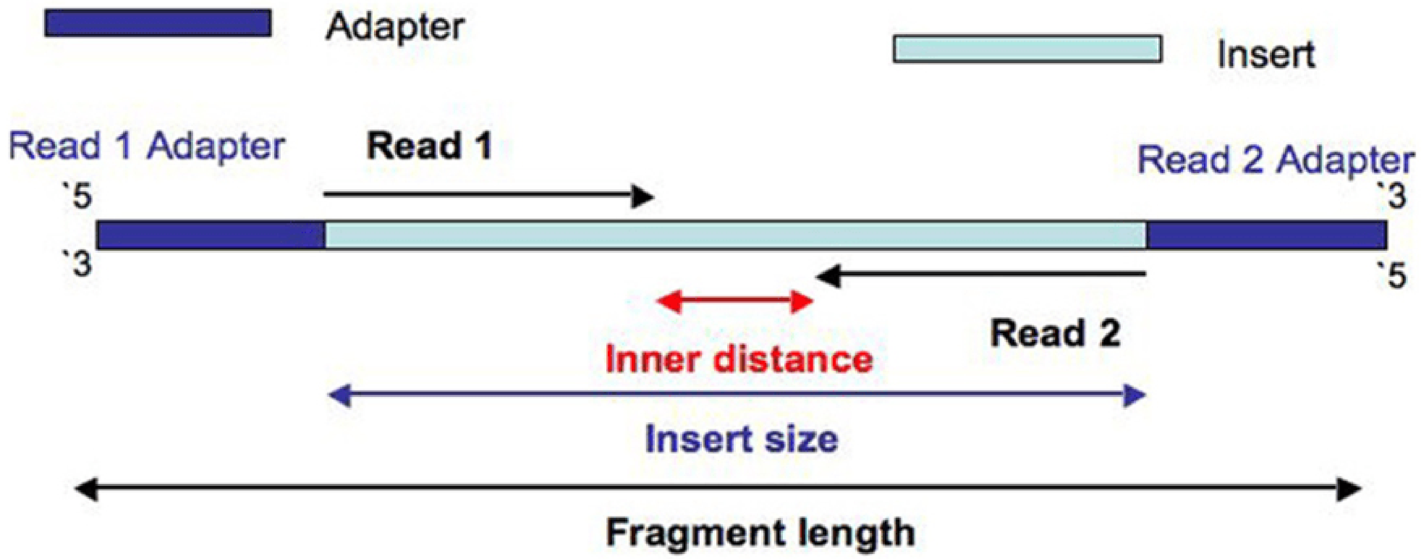


Removes P (shortest) and projects information onto Q

Clarification: Paired-ended reads do not always cover our whole insert

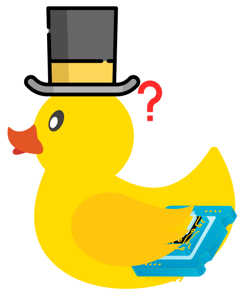
Read 1 (forward) and Read 2 (reverse) are stored in FASTQ

If our insert (i.e., DNA sample) is longer than reads, then we don't sequence the inner distance



Should we minimize this inner distance?

False



A gap between paired reads gives us insight into repeated regions

Suppose I have an "AT" repeat for both Read 1 and 2

The assembler will have to figure out if these are overlapped or separated, but by how far?

ATATATATATATATATATATATATATATATATATATATATATATATATATATATATAT**ATATATATATATATATAT**

Read 1 **Gap** **Read 2**

Having a gap tells me they don't overlap, but for how long?

Knowing length of Read 1, Read 2, and total insert length allows me to calculate gap length

Assembly algorithms (e.g., SPAdes) can estimate this and refine their results

Assemblers provide contigs and scaffolds

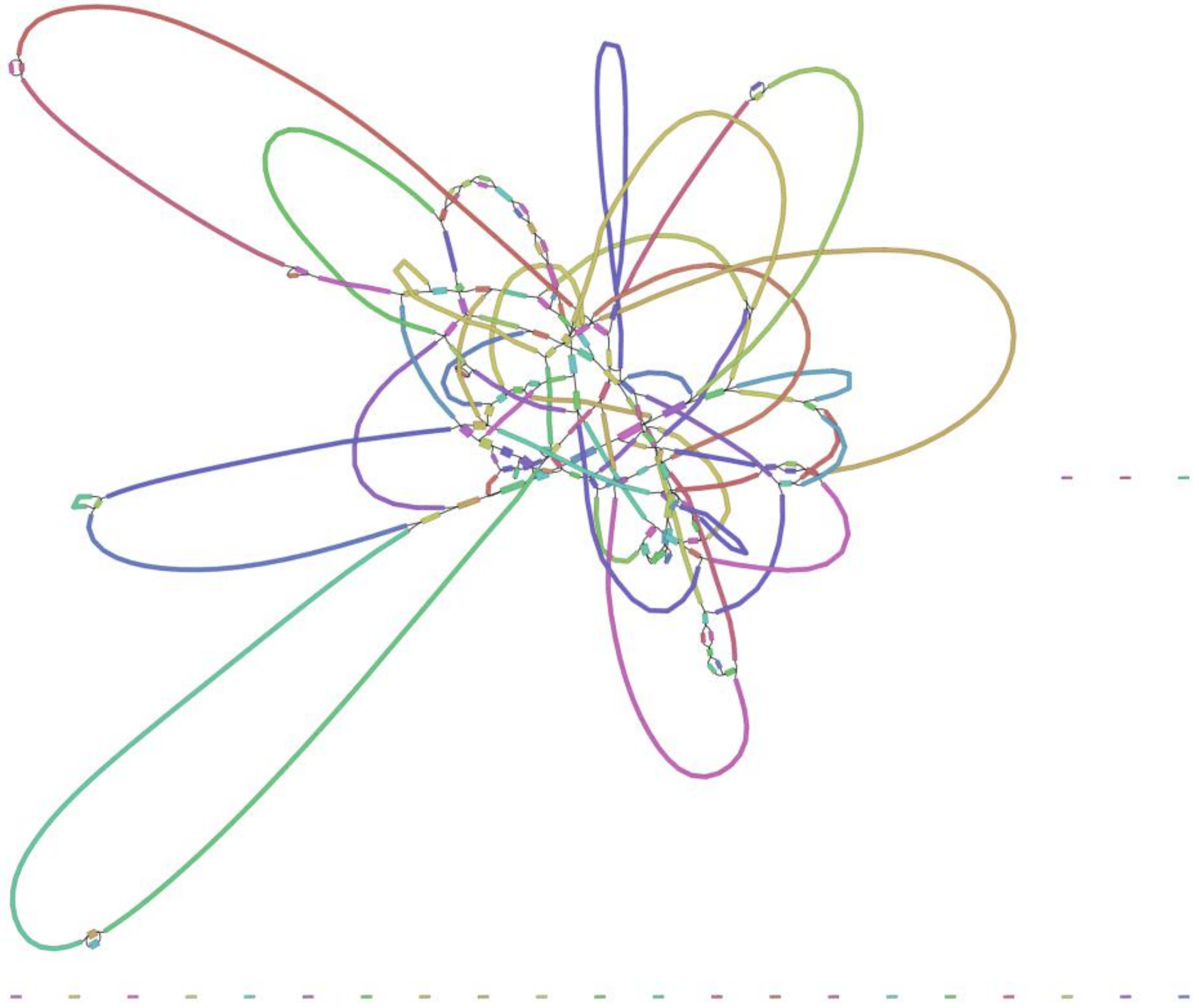
TAATAATAATCCTATCCTAGGTCGGGATCTAATAATAA   TAATAATAAGTAGTCAACTTCACTAATAATAA

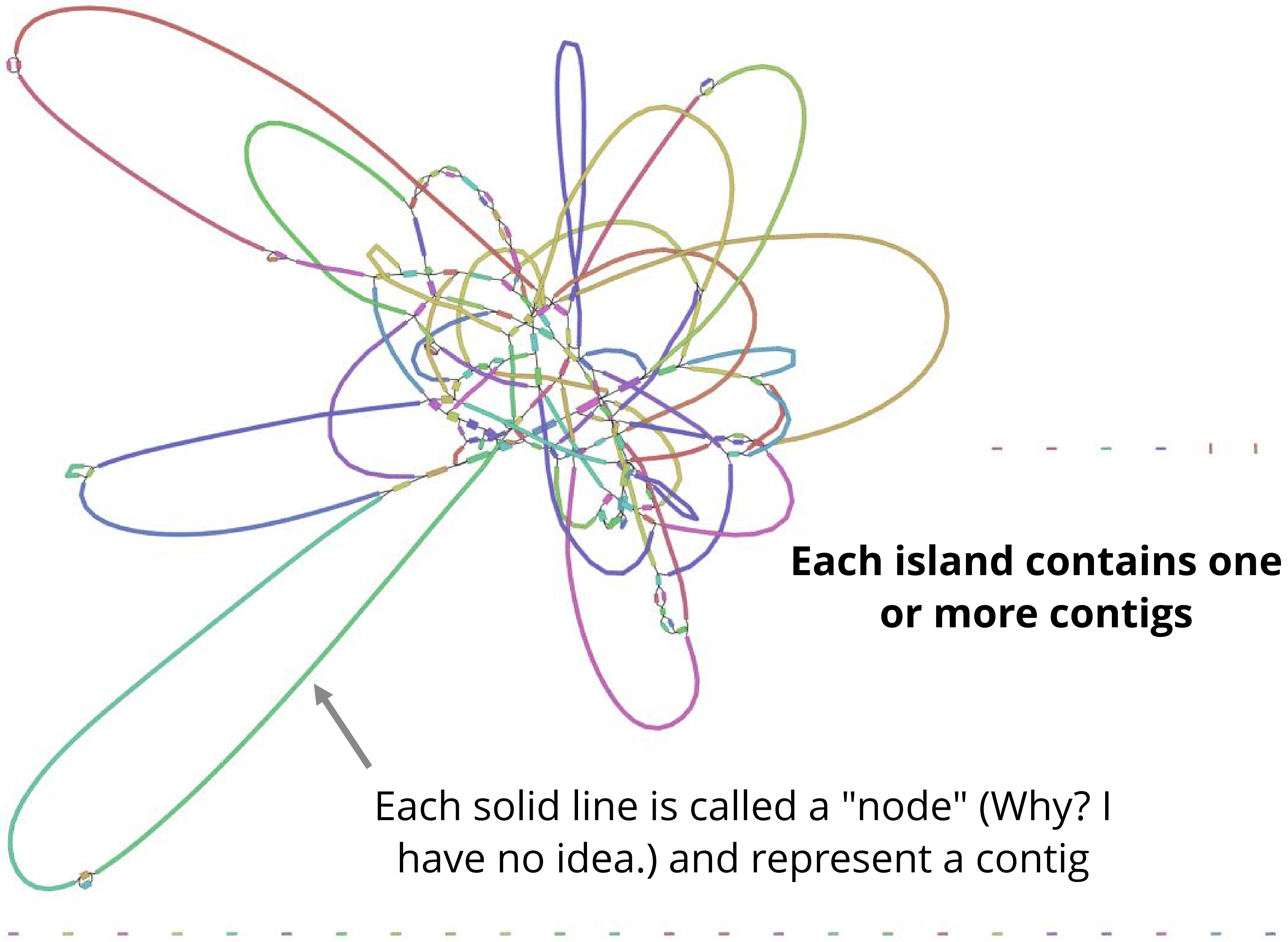
Contigs

TAATAATAATCCTATCCTAGGTCGGGATCTAATAATAANNNNNNNN TAATAATAAGTAGTCAACTTCACTAATAATAA

Scaffolds

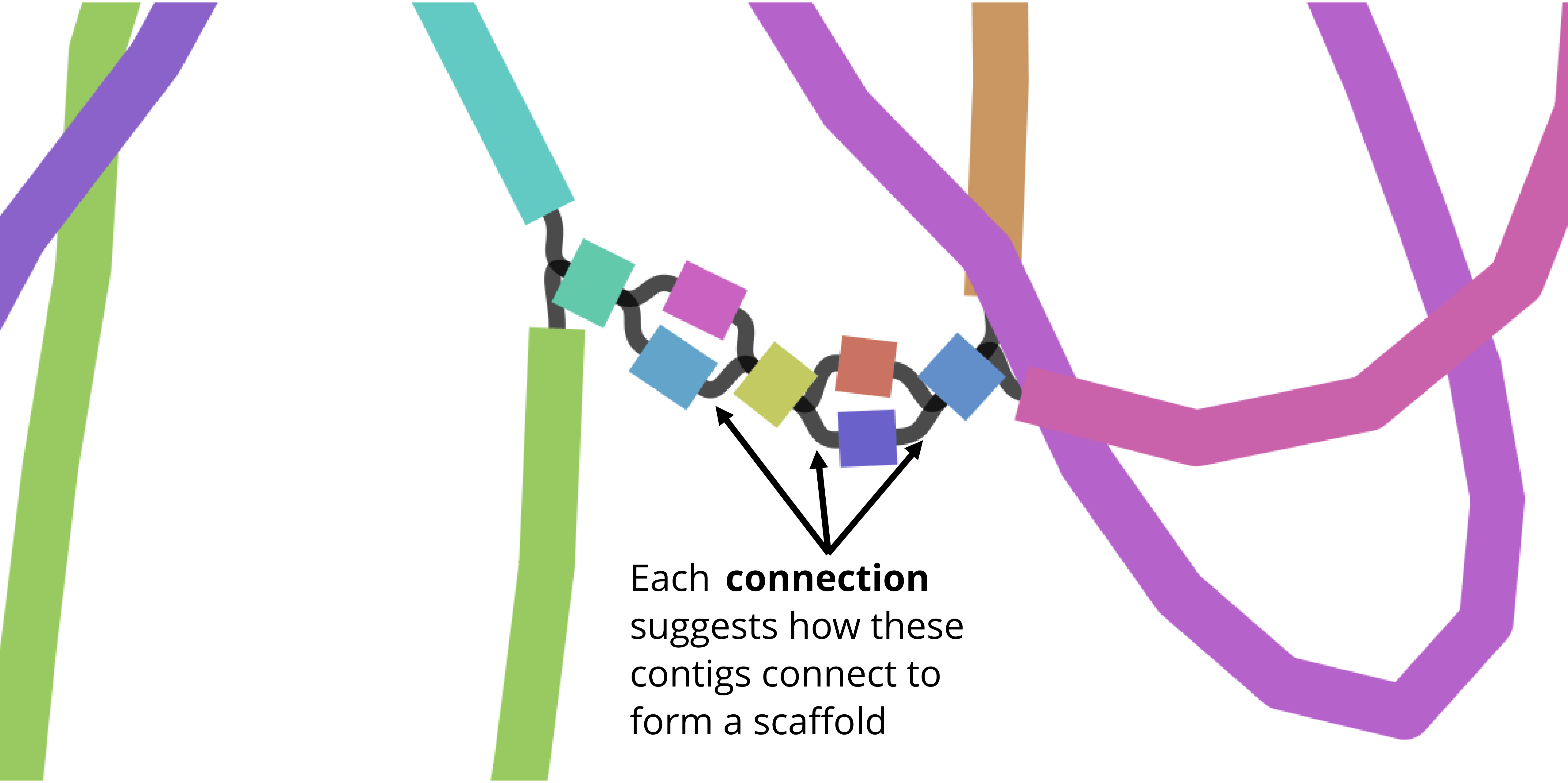
We can visualize this using an **assembly graph** from a tool called Bandage





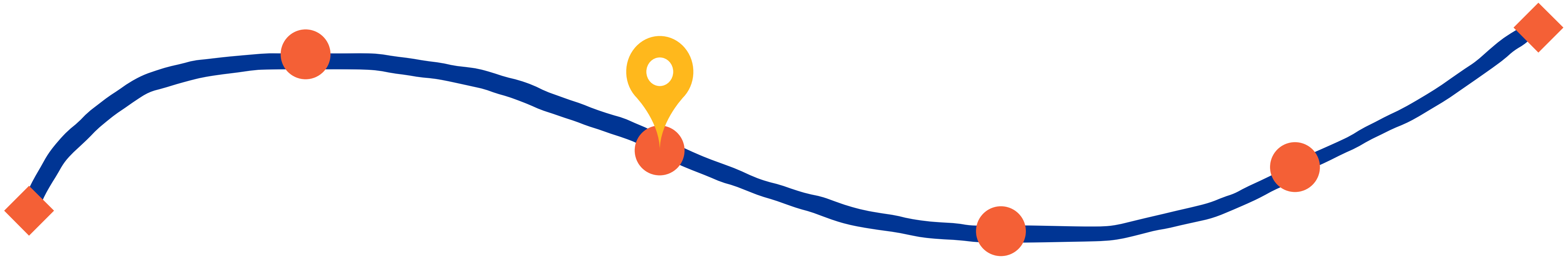
Each island contains one or more contigs

Each solid line is called a "node" (Why? I have no idea.) and represent a contig



Each **connection** suggests how these contigs connect to form a scaffold

After today, you should be able to

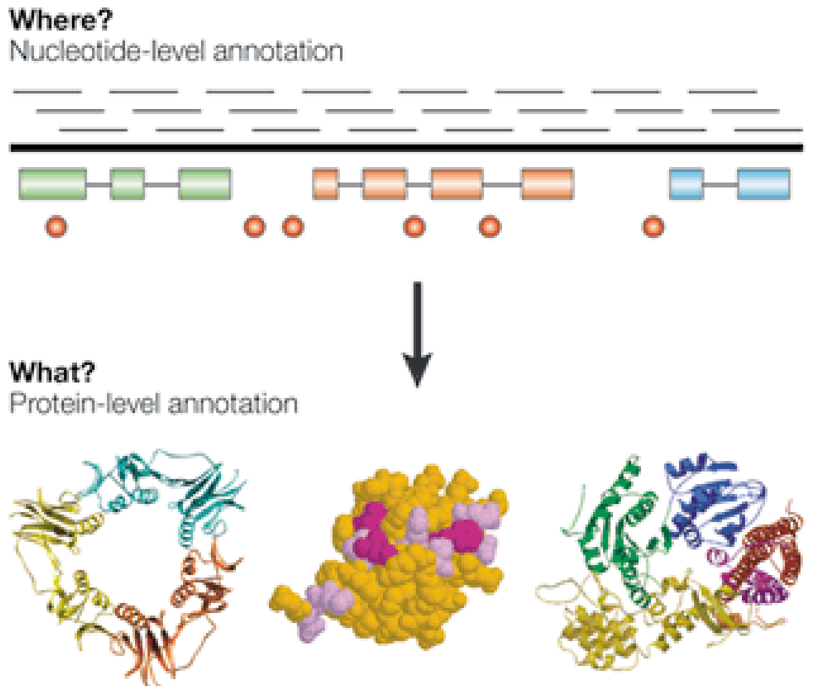


1. Explain the graph traversal and contig extraction process in genome assemblers.
2. Understand key output files and quality metrics of genome assembly results.
3. **Define gene annotation and describe its key components.**
4. Outline the main computational methods used in gene prediction and annotation.
5. Analyze and interpret basic gene annotation data and outputs.

Annotation is identifying the genetic elements and function in our contigs

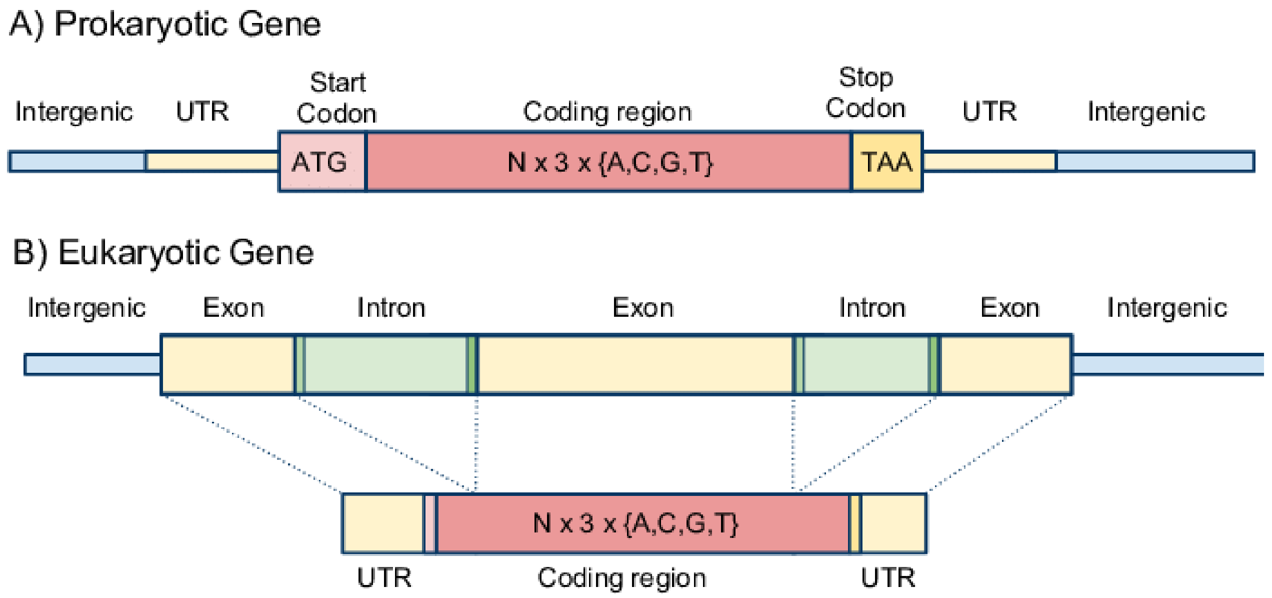
Structural annotation identifies critical genetic elements such as genes, promoters, and regulatory elements

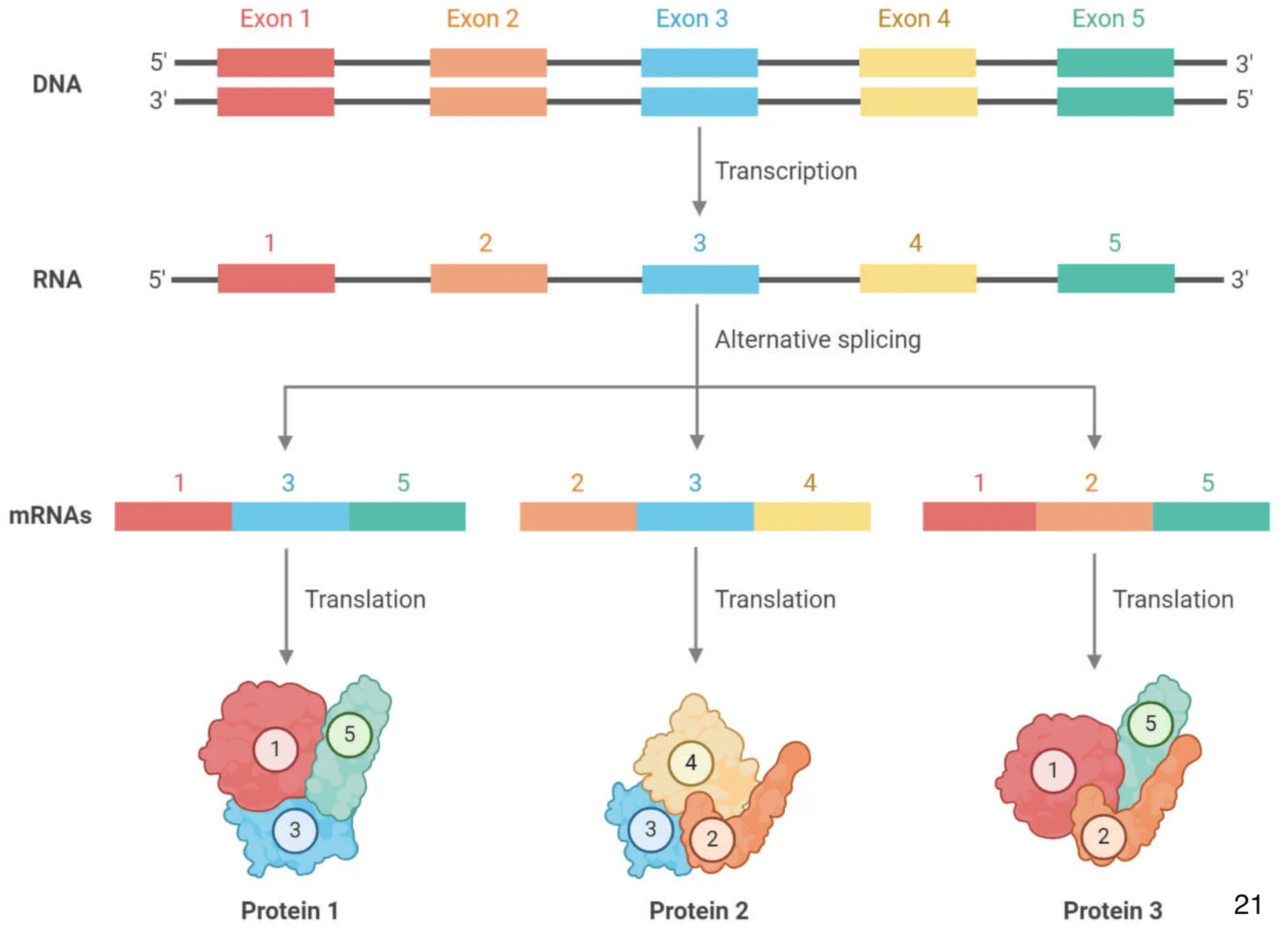
Functional annotation predicts the function of genetic elements



Eukaryote annotation is significantly more challenging that prokaryote

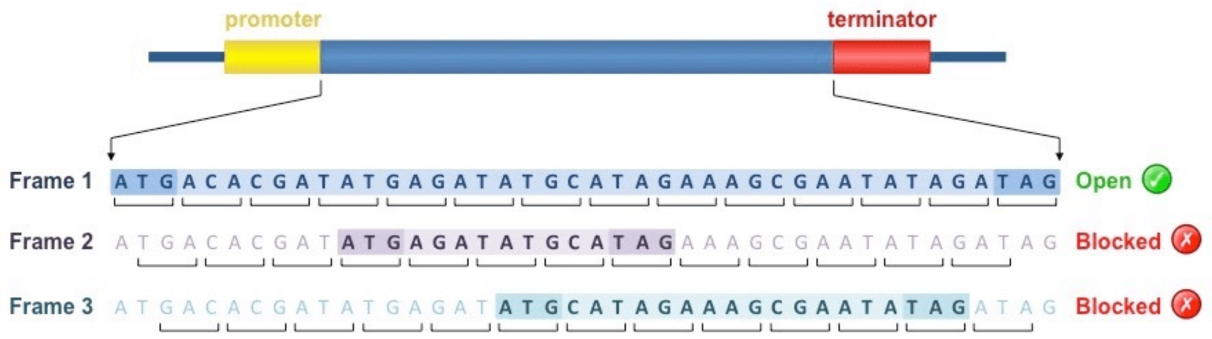
Introns and alternative splicing complicate annotation





Ab initio annotation for prokaryotes is tractable

Prokaryotes

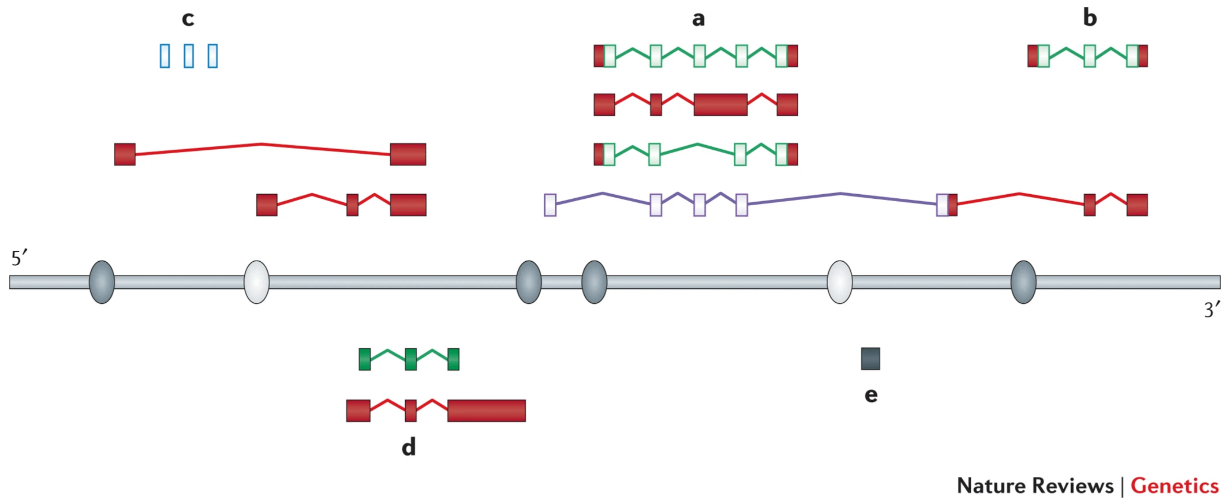


promoter
terminator
Frame 1 ATGACACGATATGAGATATGCATAGAAAGCGAATATAGATAG Open
Frame 2 ATGACACGATATGAGATATGCATAGAAAGCGAATATAGATAG Blocked
Frame 3 ATGACACGATATGAGATATGCATAGAAAGCGAATATAGATAG Blocked

Probabilistic models to identify open reading frames

Example: Prokka

Eukaryotes

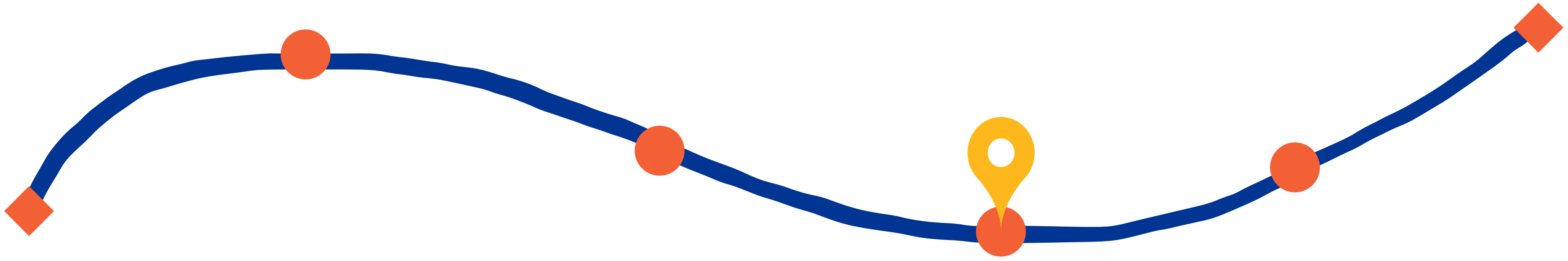


Accuracy demands supporting evidence like mRNA sequencing

Example: AUGUSTUS

We will focus on prokaryotes because eukaryotes are way more complicated

After today, you should be able to



1. Explain the graph traversal and contig extraction process in genome assemblers.
2. Understand key output files and quality metrics of genome assembly results.
3. Define gene annotation and describe its key components.
4. **Outline the main computational methods used in gene prediction and annotation.**
5. Analyze and interpret basic gene annotation data and outputs.

Identify open reading frames (ORF)

Seek the standard start codons: ATG, GTG or TTG

Seek stop codons based on the translation table

- TAA, TAG, and TGA for bacteria, archaea, and plant plastids (Code 11)

Score potential ORFs

$$S_{\mathrm{ORF}}(n) = 4.25\left[S_R(n) + S_T(n) + 0.4 S_U(n)\right] + S_C(n)$$

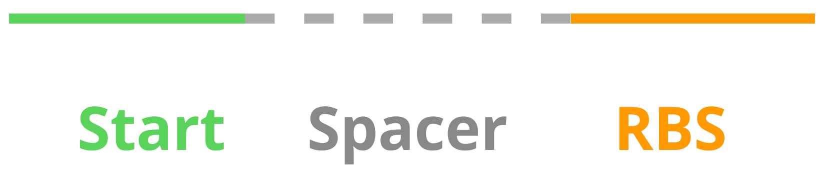
$S_R(n)$	$S_T(n)$	$S_U(n)$	$S_C(n)$
Ribosomal binding site motif score	**Start type score**	**Upstream score**	**Coding score**

Prokka uses prodigal for this

(I will use different notation than the paper.)

RBS score computed from dataset fitting

Search for RBS motif after start codon; choose whichever has the lowest bin number



Took **training data from 12 annotated genomes** $\longrightarrow$ Computed **frequency of RBS motif bin** in

- Entire sequence (Baseline) $B(n)$
- RBS frequency $R(n)$

$$S_R(n) = \log\left(\frac{R(n)}{B(n)}\right)$$

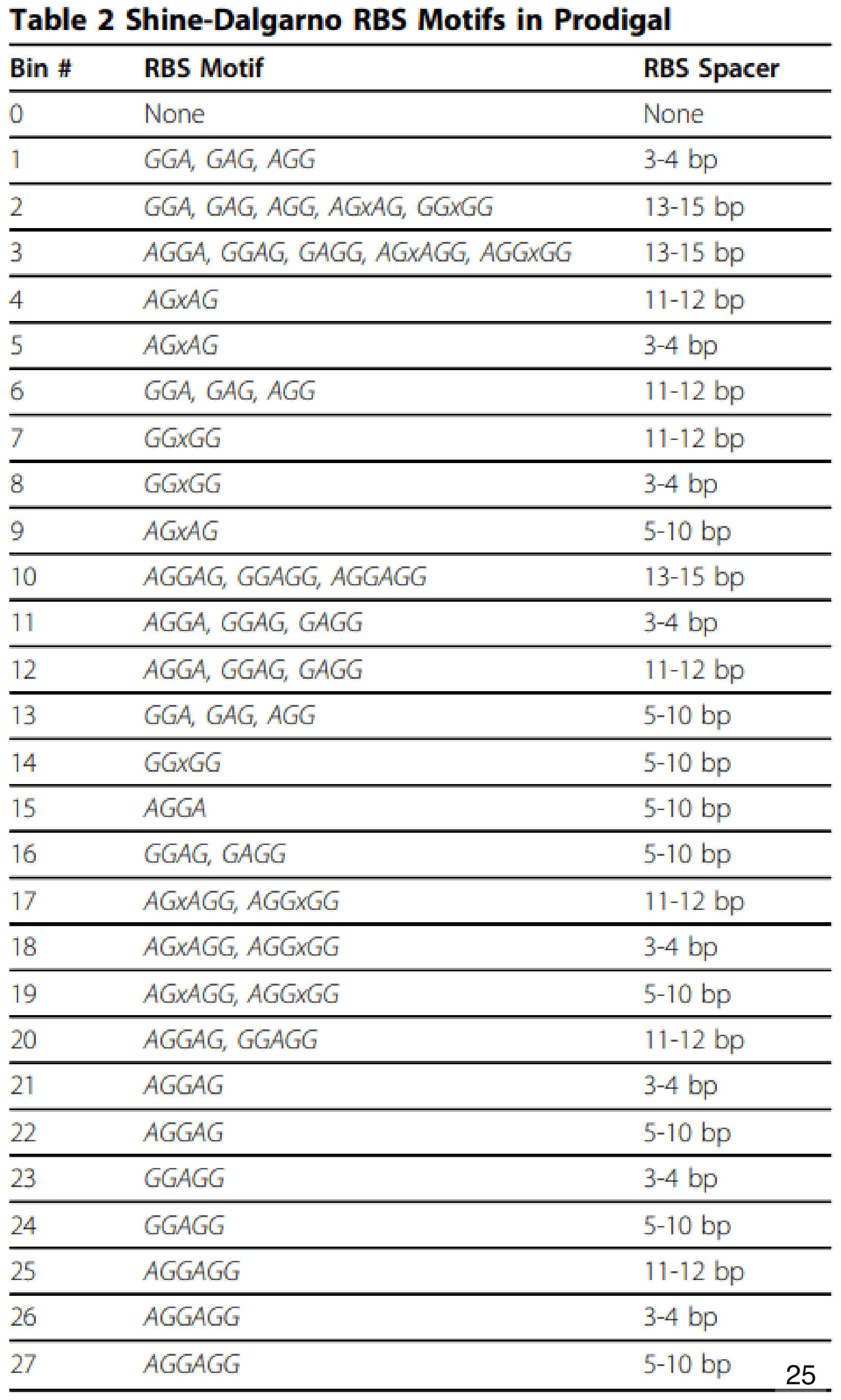


Table 2 Shine-Dalgarno RBS Motifs in Prodigal

Bin #	RBS Motif	RBS Spacer
0	None	None
1	*GGA, GAG, AGG*	3-4 bp
2	*GGA, GAG, AGG, AGxAG, GGxGG*	13-15 bp
3	*AGGA, GGAG, GAGG, AGxAGG, AGGxGG*	13-15 bp
4	*AGxAG*	11-12 bp
5	*AGxAG*	3-4 bp
6	*GGA, GAG, AGG*	11-12 bp
7	*GGxGG*	11-12 bp
8	*GGxGG*	3-4 bp
9	*AGxAG*	5-10 bp
10	*AGGAG, GGAGG, AGGAGG*	13-15 bp
11	*AGGA, GGAG, GAGG*	3-4 bp
12	*AGGA, GGAG, GAGG*	11-12 bp
13	*GGA, GAG, AGG*	5-10 bp
14	*GGxGG*	5-10 bp
15	*AGGA*	5-10 bp
16	*GGAG, GAGG*	5-10 bp
17	*AGxAGG, AGGxGG*	11-12 bp
18	*AGxAGG, AGGxGG*	3-4 bp
19	*AGxAGG, AGGxGG*	5-10 bp
20	*AGGAG, GGAGG*	11-12 bp
21	*AGGAG*	3-4 bp
22	*AGGAG*	5-10 bp
23	*GGAGG*	3-4 bp
24	*GGAGG*	5-10 bp
25	*AGGAGG*	11-12 bp
26	*AGGAGG*	3-4 bp
27	*AGGAGG*	5-10 bp

Start codon score given by similar RBS framework

Took **training data from 12 annotated genomes** $\longrightarrow$ Computed **frequency of start codon** in

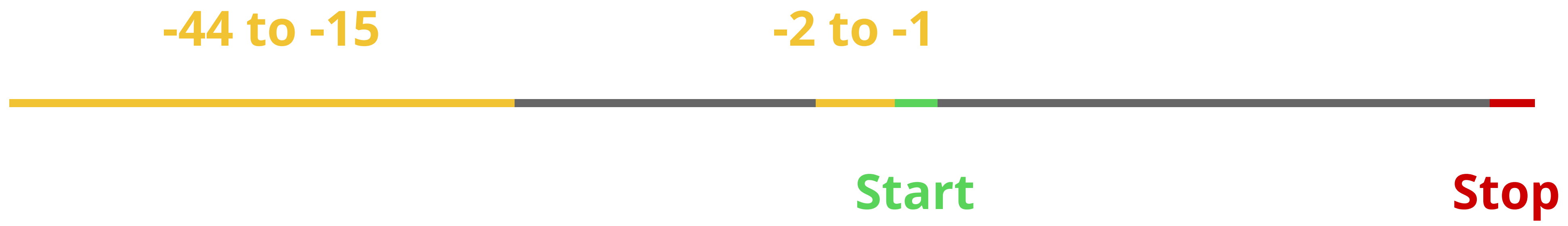
- Entire sequence (Baseline) $B(n)$
- Start codon frequency $T(n)$

$$S_T(n) = \log\left(\frac{T(n)}{B(n)}\right)$$

Upstream score based on base analysis

By analyzing base frequency in specific upstream regions, their annotation results improved

Essentially looking for promotors



$$S_U(n) = w_{\text{start}} \sum_{i \in P} p_i \left(\text{nuc}_i\right)$$

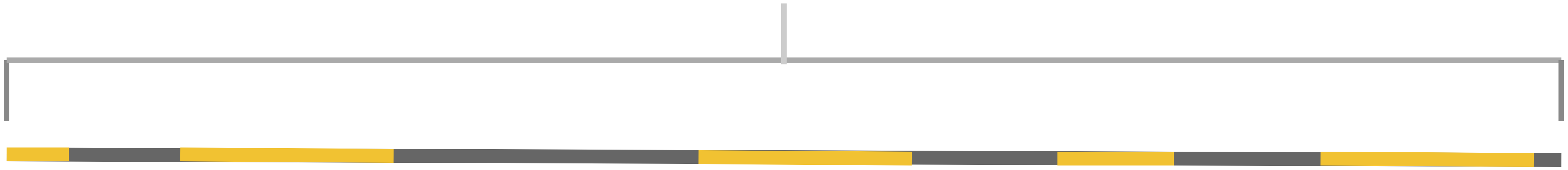
Coding score computed based on gene enrichment parameters

Computed **frequency of nucleotide hexamers called "words"** in

Example hexamers called "words"

ATGGCC	CAGCTG
ACTAGT	GGGCCC

$B(w)$ Compute probability of observing word within the whole genome



$G(w)$ Compute the probability of observing word within genes

$$C(w) = \log\left(\frac{G(w)}{B(w)}\right)$$

Word coding score

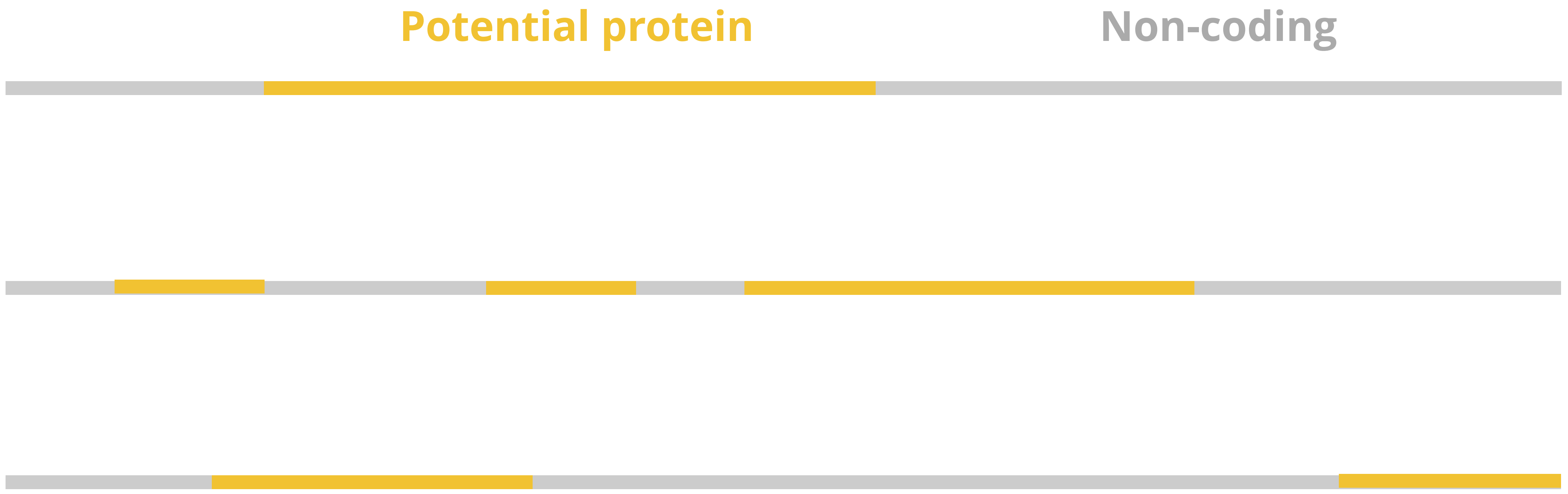
Coding score computed based on gene enrichment parameters

It can be thought of as
"How often does this word appear in genes?"

$$S_C(n_{\text{start}} \ldots n_{\text{stop}}) = \sum_{i=n_{\text{start}}}^{n_{\text{stop}}} C\left(w(i)\right)$$

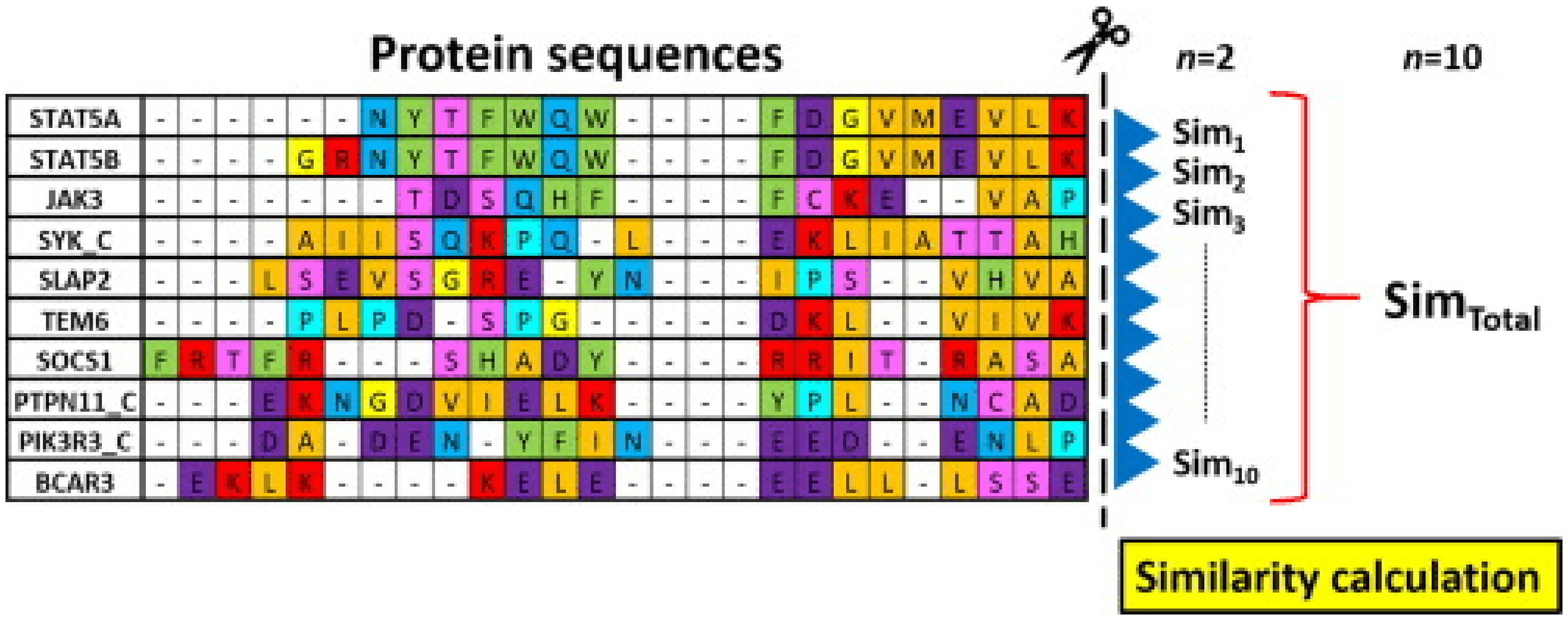
Gene coding score

Results: Sequences that likely encode for proteins

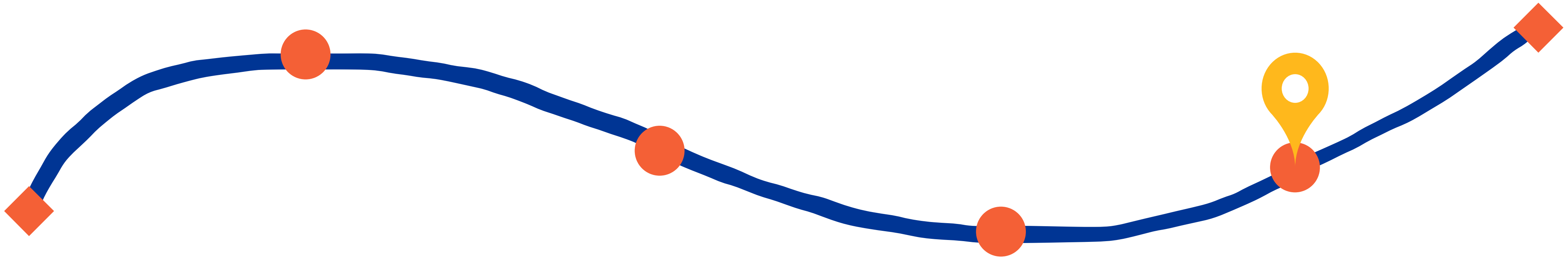


Functional annotation is normally based on protein database search

Similarity search will be our topic for Thursday



After today, you should be able to



1. Explain the graph traversal and contig extraction process in genome assemblers.
2. Understand key output files and quality metrics of genome assembly results.
3. Define gene annotation and describe its key components.
4. Outline the main computational methods used in gene prediction and annotation.
5. **Analyze and interpret basic gene annotation data and outputs.**

Prokka will provide several outputs

```
>ECNNONJI_02637 Dihydrofolate reductase
MTLSILVAHDLQRVIGFENQLPWHLPNDLKHVKKLSTGHTLVMGRKTFESIGKPLPNRRN
VVLTSDTSFNVEGVDVIHSIEDIYQLPGHVFIFGGQTLFEEMIDKVDDMYITVIEGKFRG
DTFFPPYTFEDWEVASSVEGKLDEKNTIPHTFLHLIRKK
```

Extension	Description
.gff	This is the master annotation in GFF3 format, containing both sequences and annotations. It can be viewed directly in Artemis or IGV.
.gbk	This is a standard Genbank file derived from the master .gff. If the input to prokka was a multi-FASTA, then this will be a multi-Genbank, with one record for each sequence.
.fna	Nucleotide FASTA file of the input contig sequences.
.faa	Protein FASTA file of the translated CDS sequences.
.ffn	Nucleotide FASTA file of all the prediction transcripts (CDS, rRNA, tRNA, tmRNA, misc_RNA)
.sqn	An ASN1 format "Sequin" file for submission to Genbank. It needs to be edited to set the correct taxonomy, authors, related publication etc.
.fsa	Nucleotide FASTA file of the input contig sequences, used by "tbl2asn" to create the .sqn file. It is mostly the same as the .fna file, but with extra Sequin tags in the sequence description lines.
.tbl	Feature Table file, used by "tbl2asn" to create the .sqn file.
.err	Unacceptable annotations - the NCBI discrepancy report.
.log	Contains all the output that Prokka produced during its run. This is a record of what settings you used, even if the --quiet option was enabled.
.txt	Statistics relating to the annotated features found.
.tsv	Tab-separated file of all features: locus_tag,ftype,len_bp,gene,EC_number,COG,product

Before the next class, you should

Lecture 05:
Gene annotation
Lecture 06:
Sequence alignment
Today
Thursday

- Finish A02, which is due Thursday at 11:59 pm.