genomics aotearoa

Metagenomics
Summer School 2024

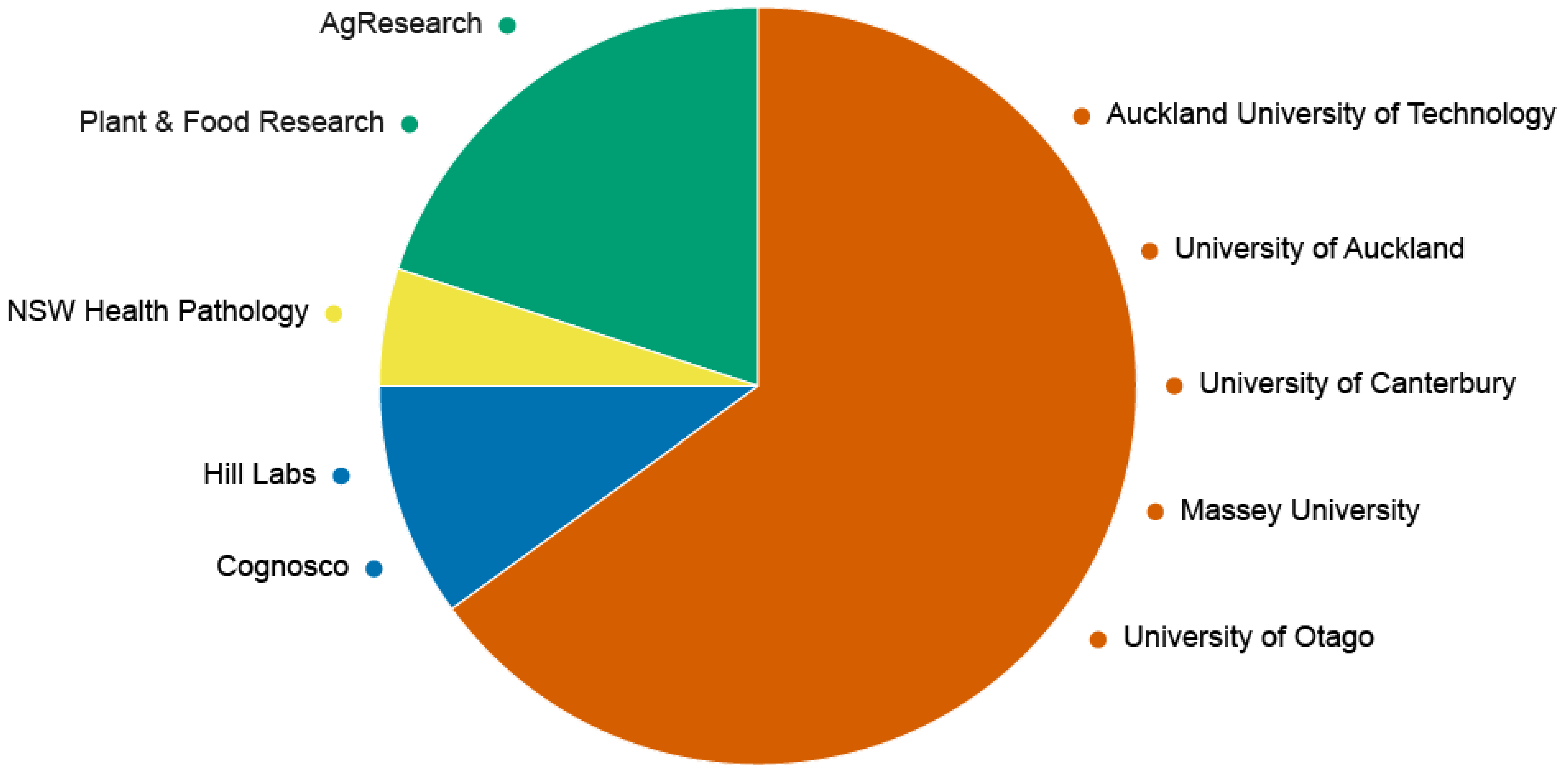
Day 1

Introductions
Metagenomics decision tree
Quality filtering WGS data
Genome assembly and evaluation

Welcome!

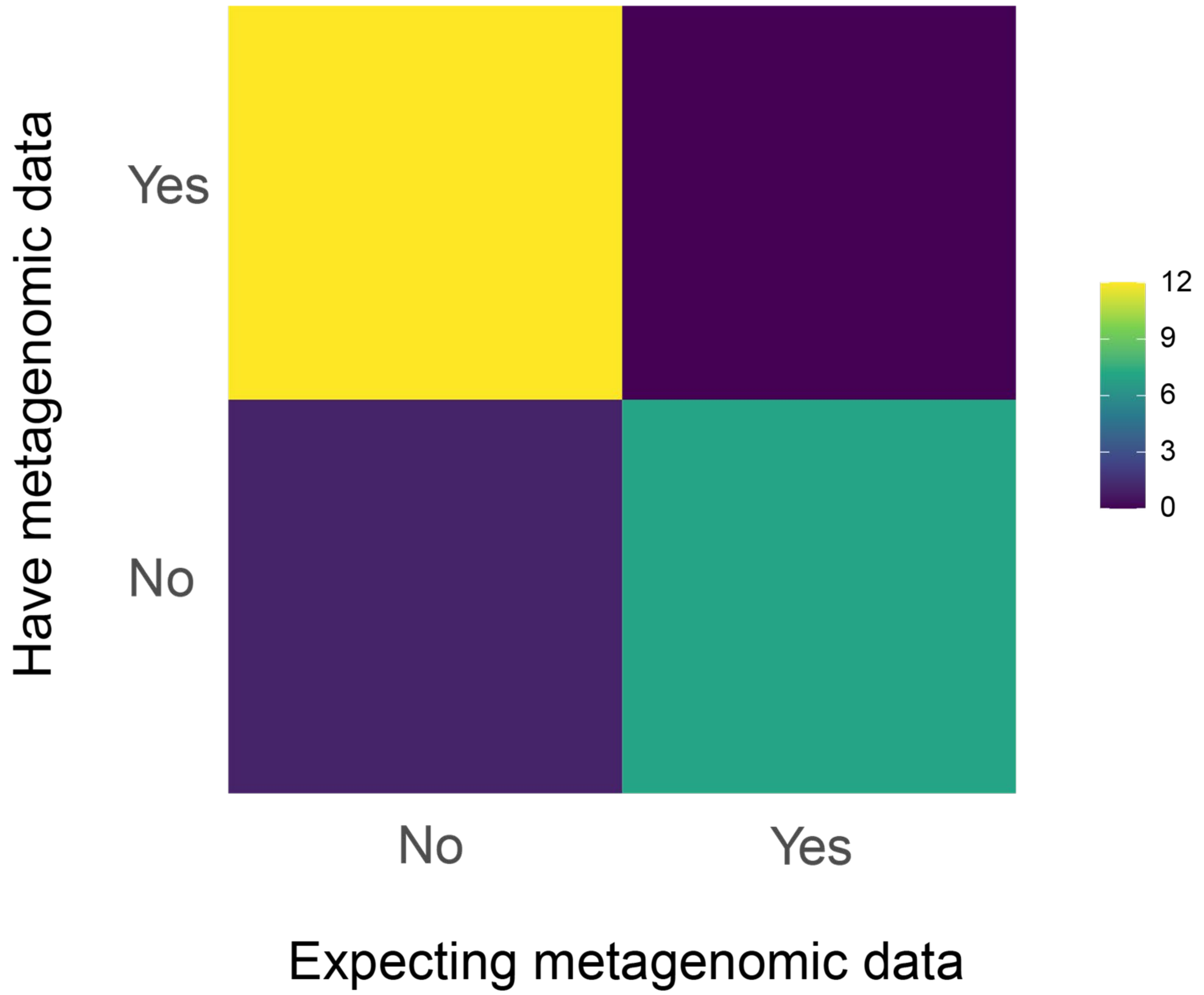
- **Housekeeping**
- **Google Doc for collaborative Q&A/comments**
  - https://tinyurl.com/mgss2024doc
- **Overview of attendees**
- **Any questions?**

Where are we from?

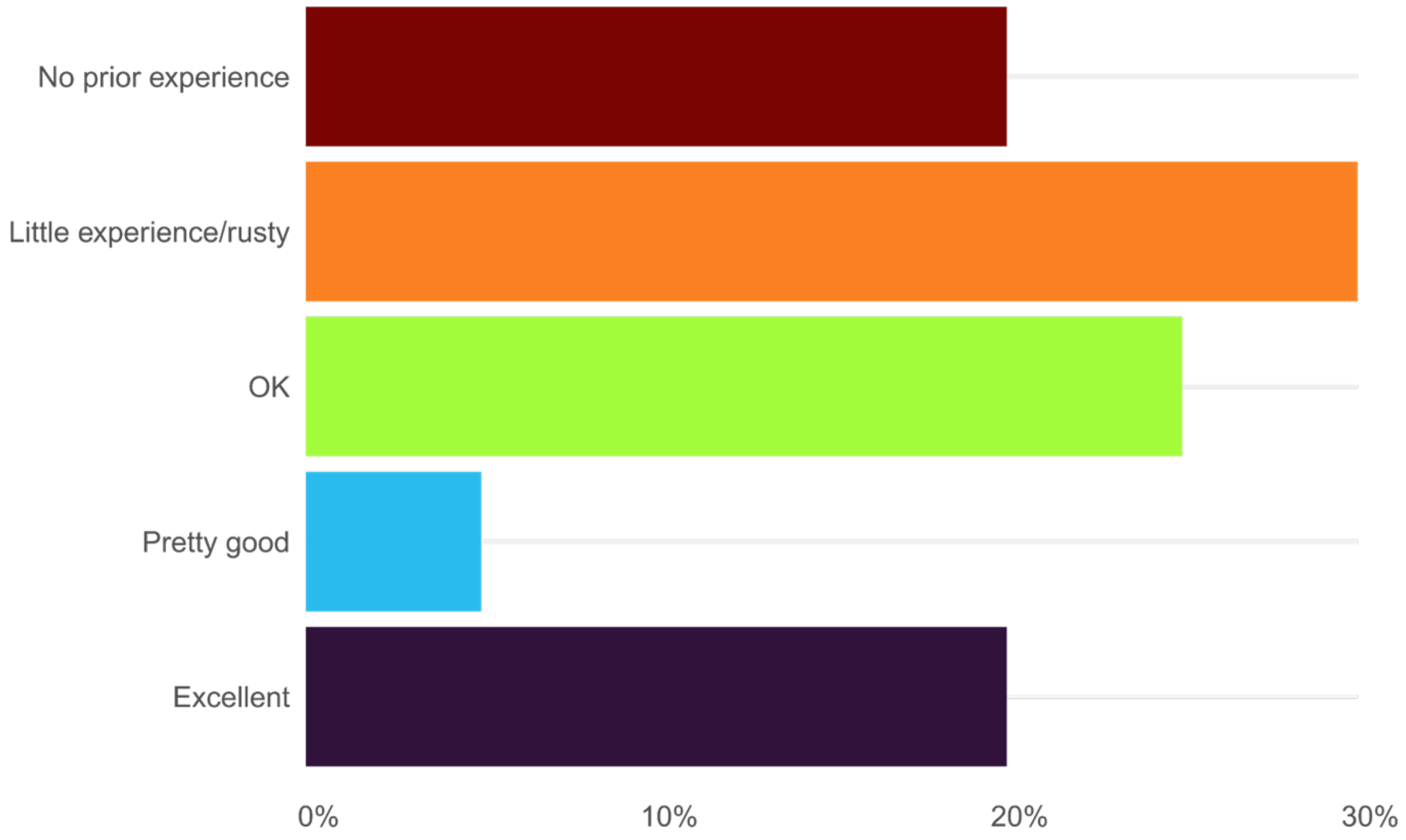


🟩 Crown Research Institute 🟨 Government 🟦 Industry 🟧 University

Do we have data?



Command line experience?



Genomics Aotearoa - Resources

Genomics Aotearoa – GitHub repositories
https://github.com/GenomicsAotearoa/

- Metagenomics Summer School material
- RNA seq workshop
- Environmental metagenomics
  - Metagenomic annotation and binning
- Methods and musings
  - Bin cluster refinement
  - Genome assembly ont
  - Metagenomic ont

Starting each session

1. Log in to the NeSI Jupyter hub via a browser

1. Open the workshop materials on GitHub

1. Optional: Open a (plain text) text editor for taking notes

Task: SLURM test

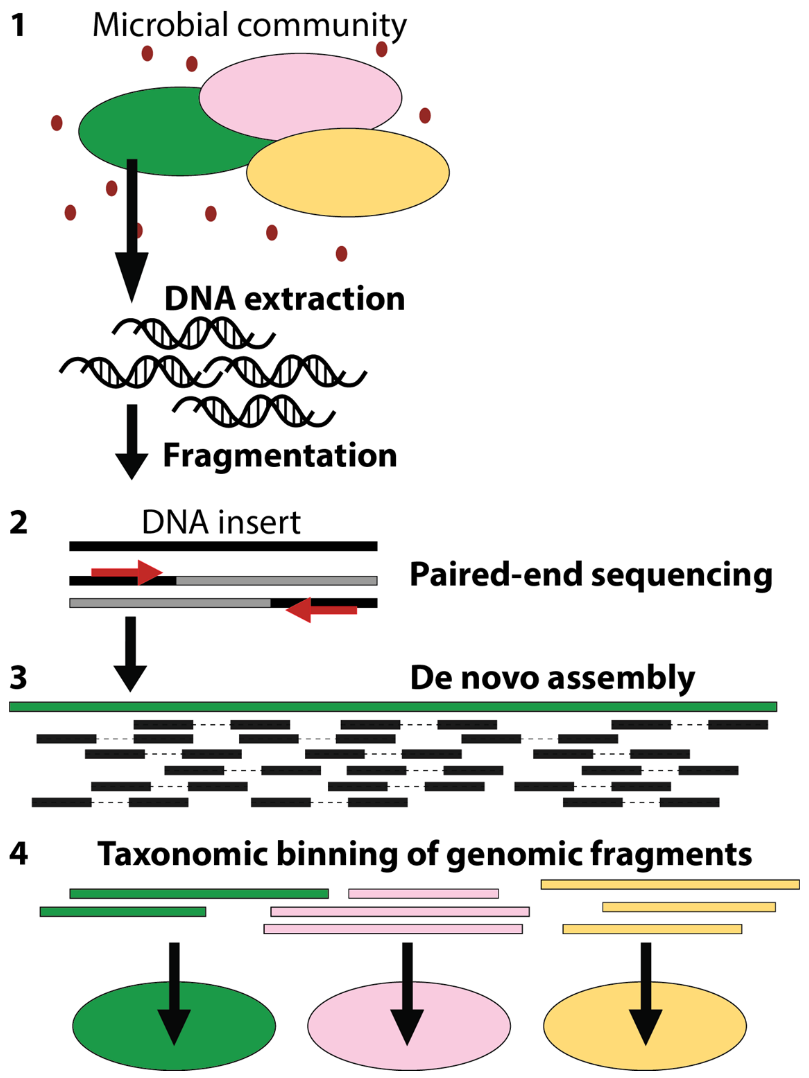
Go to Github MGSS webpage

Tasks:

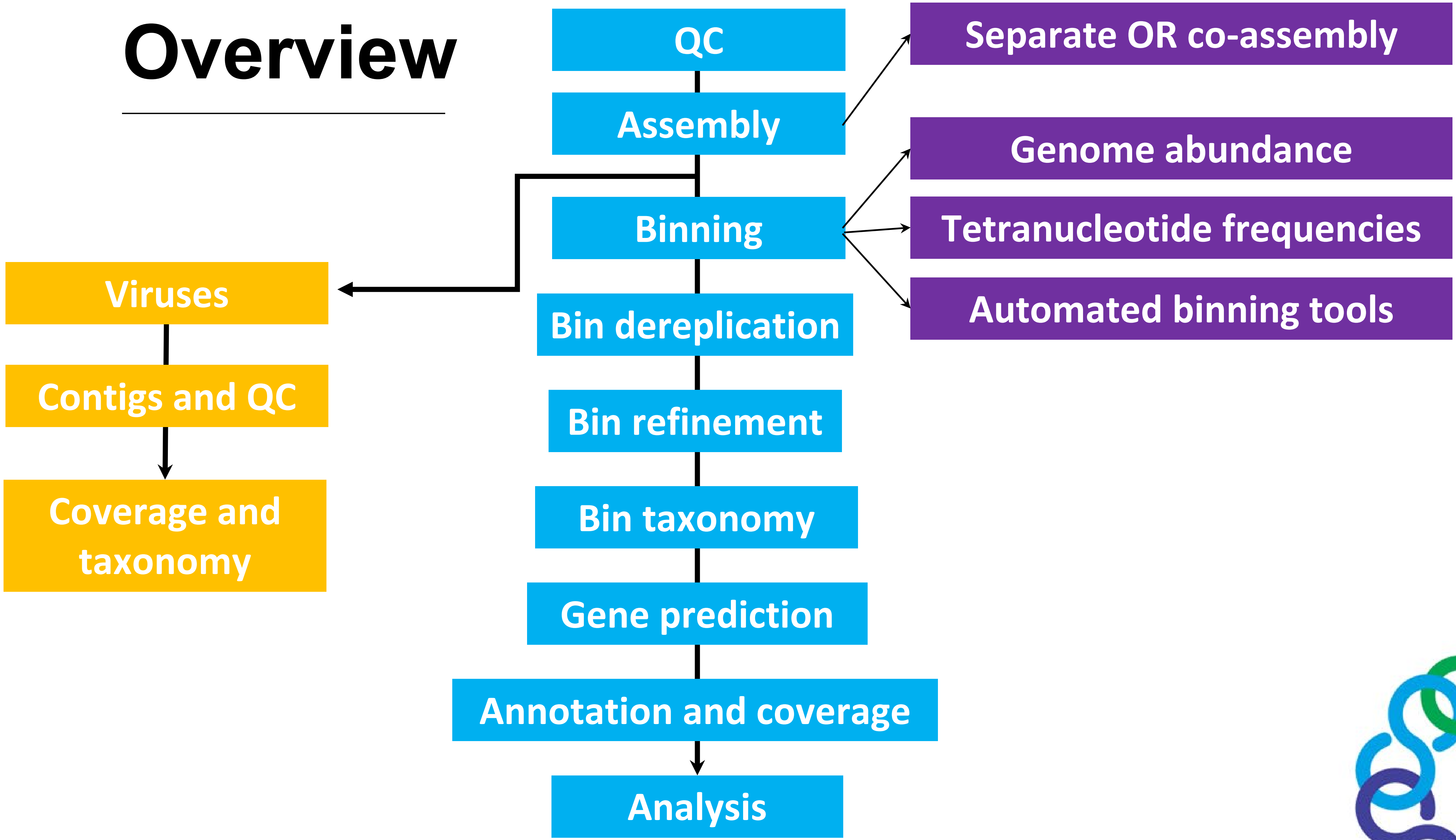
- **Submit the `bowtie-test.slurm` job**

Metagenomic decision tree(s)

Our goal: genome recovery

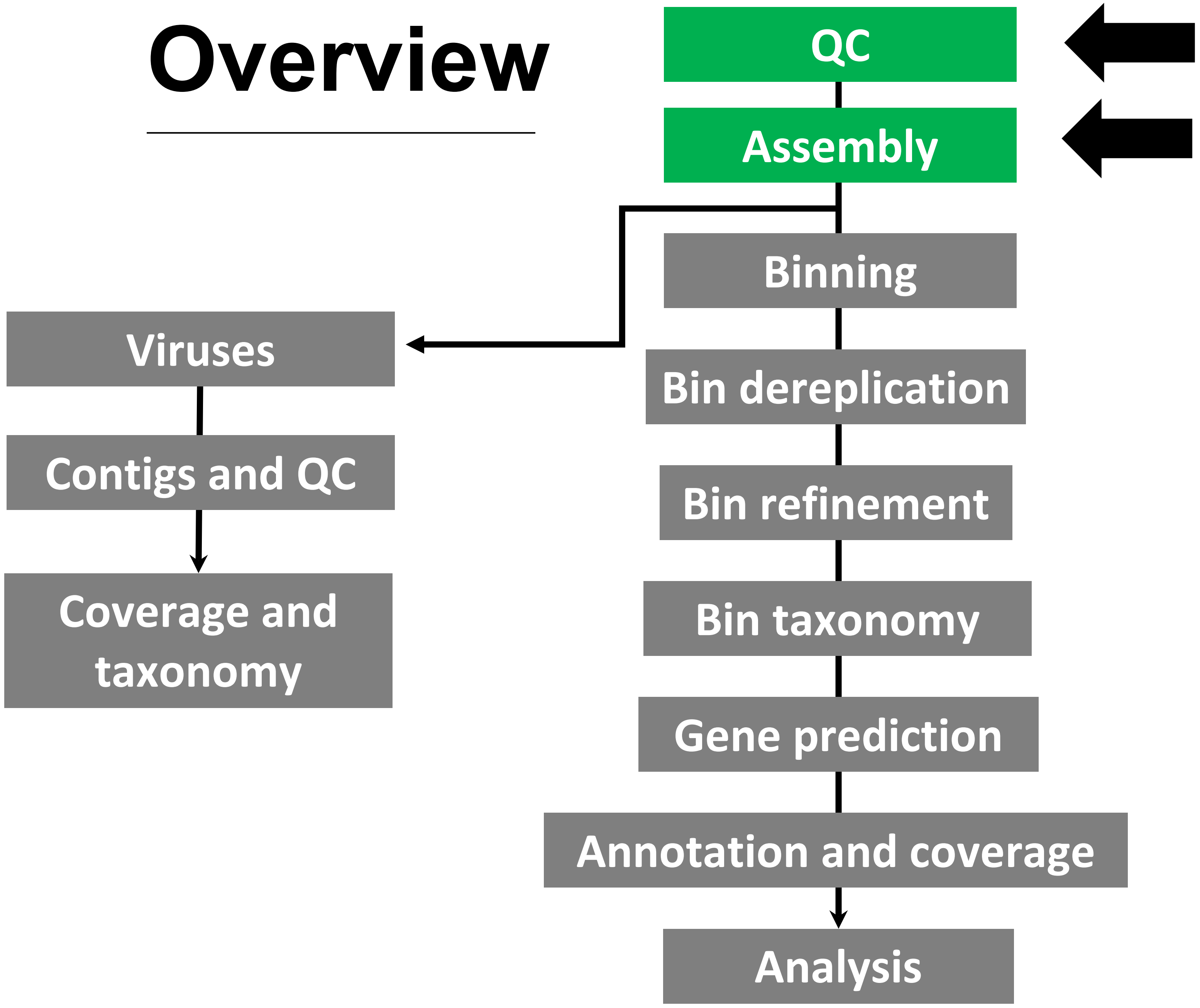


Overview



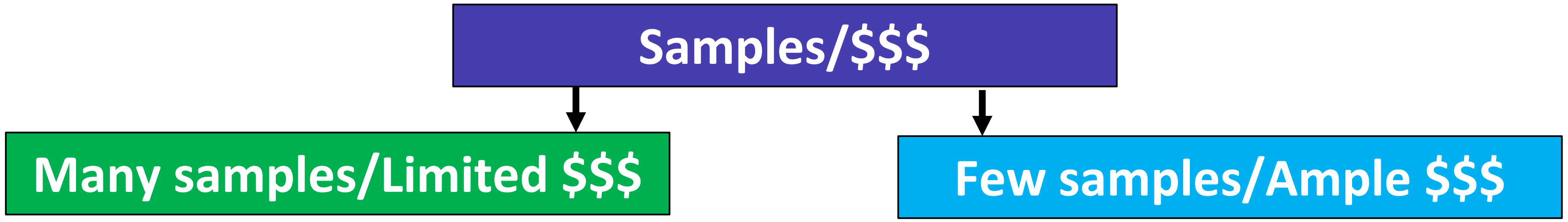
QC
Assembly
Binning
Bin dereplication
Bin refinement
Bin taxonomy
Gene prediction
Annotation and coverage
Analysis
Separate OR co-assembly
Genome abundance
Tetranucleotide frequencies
Automated binning tools
Viruses
Contigs and QC
Coverage and taxonomy

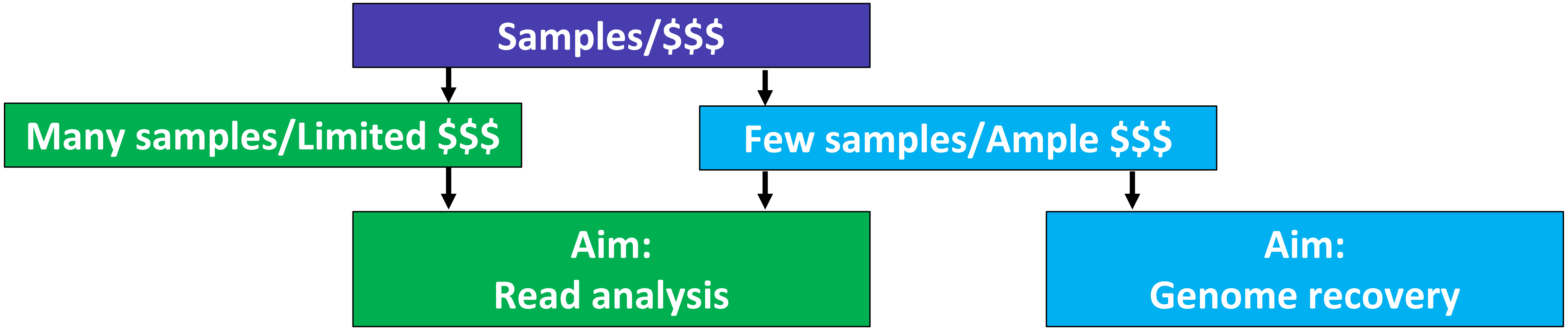
Overview

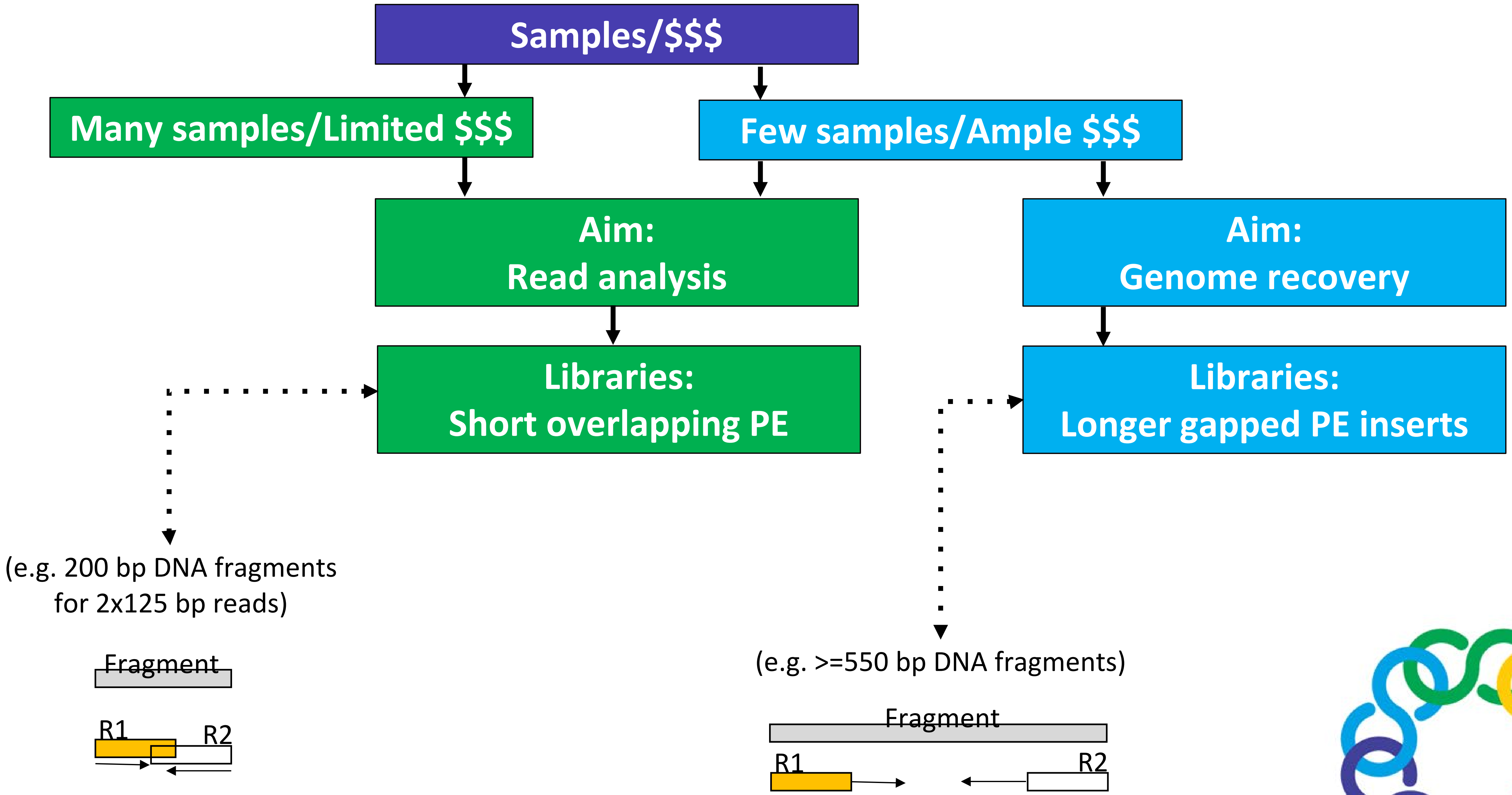

QC
Assembly
Binning
Bin dereplication
Bin refinement
Bin taxonomy
Gene prediction
Annotation and coverage
Analysis
Viruses
Contigs and QC
Coverage and taxonomy

Decision tree

- Starts with experimental design
- DNA extraction
- WGS library prep
- Amount of sequencing

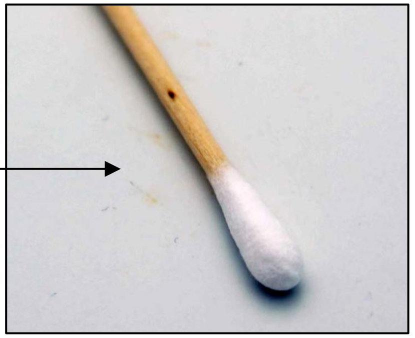

Samples/$$$
Many samples/Limited $$$
Few samples/Ample $$$


Samples/$$$
Many samples/Limited $$$
Few samples/Ample $$$
Aim:
Read analysis
Aim:
Genome recovery


Samples/$$$
Many samples/Limited $$$
Few samples/Ample $$$
Aim:
Read analysis
Aim:
Genome recovery
Libraries:
Short overlapping PE
Libraries:
Longer gapped PE inserts
(e.g. 200 bp DNA fragments
for 2x125 bp reads)
Fragment
R1
R2
(e.g. >=550 bp DNA fragments)
Fragment
R1
R2

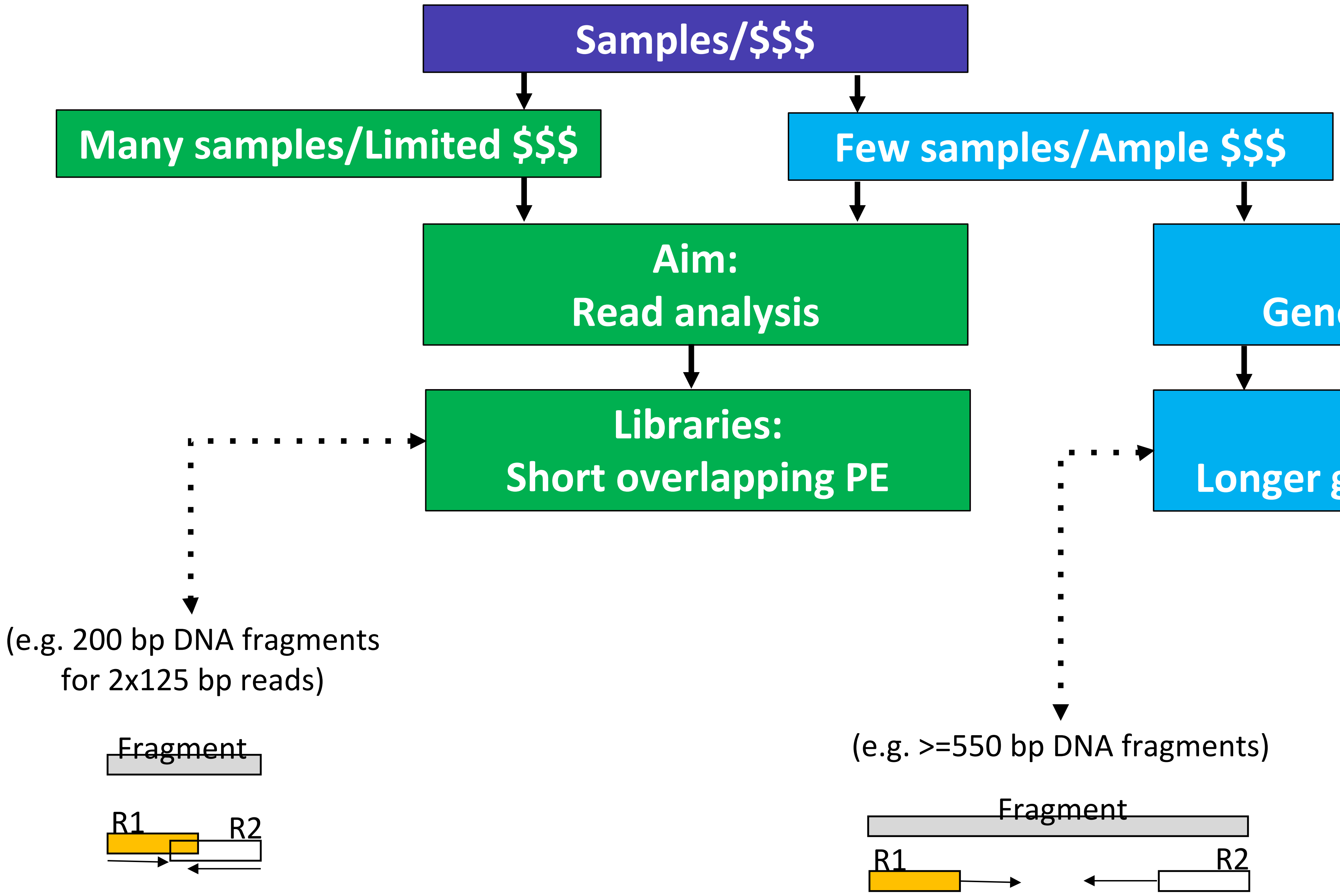
DNA input

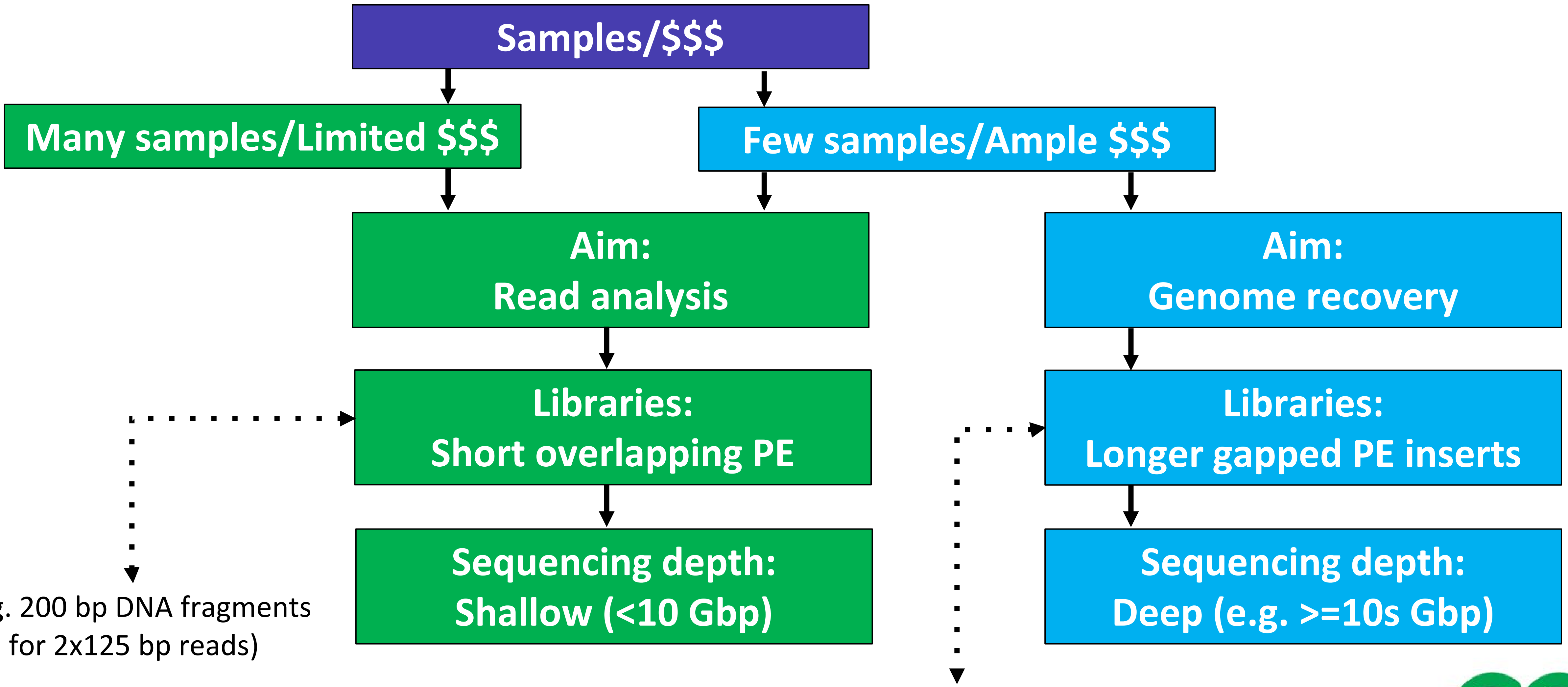
- **Very low inputs (e.g. nanograms) for Nextera library prep = enzymatic fragmentation with broad size distributions**



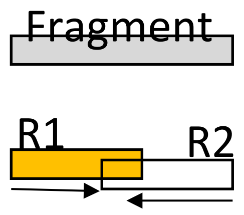
- **High inputs (e.g. 100s ng) for TruSeq = physical fragmentation with defined size selection**

Tends to yield sequences of larger inserts

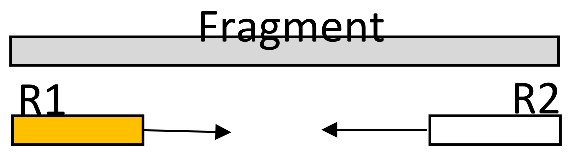

Samples/$$$
Many samples/Limited $$$
Few samples/Ample $$$
Aim:
Read analysis
Aim:
Genome recovery
Libraries:
Short overlapping PE
Libraries:
Longer gapped PE inserts
(e.g. 200 bp DNA fragments
for 2x125 bp reads)
Fragment
R1
R2
(e.g. >=550 bp DNA fragments)
Fragment
R1
R2



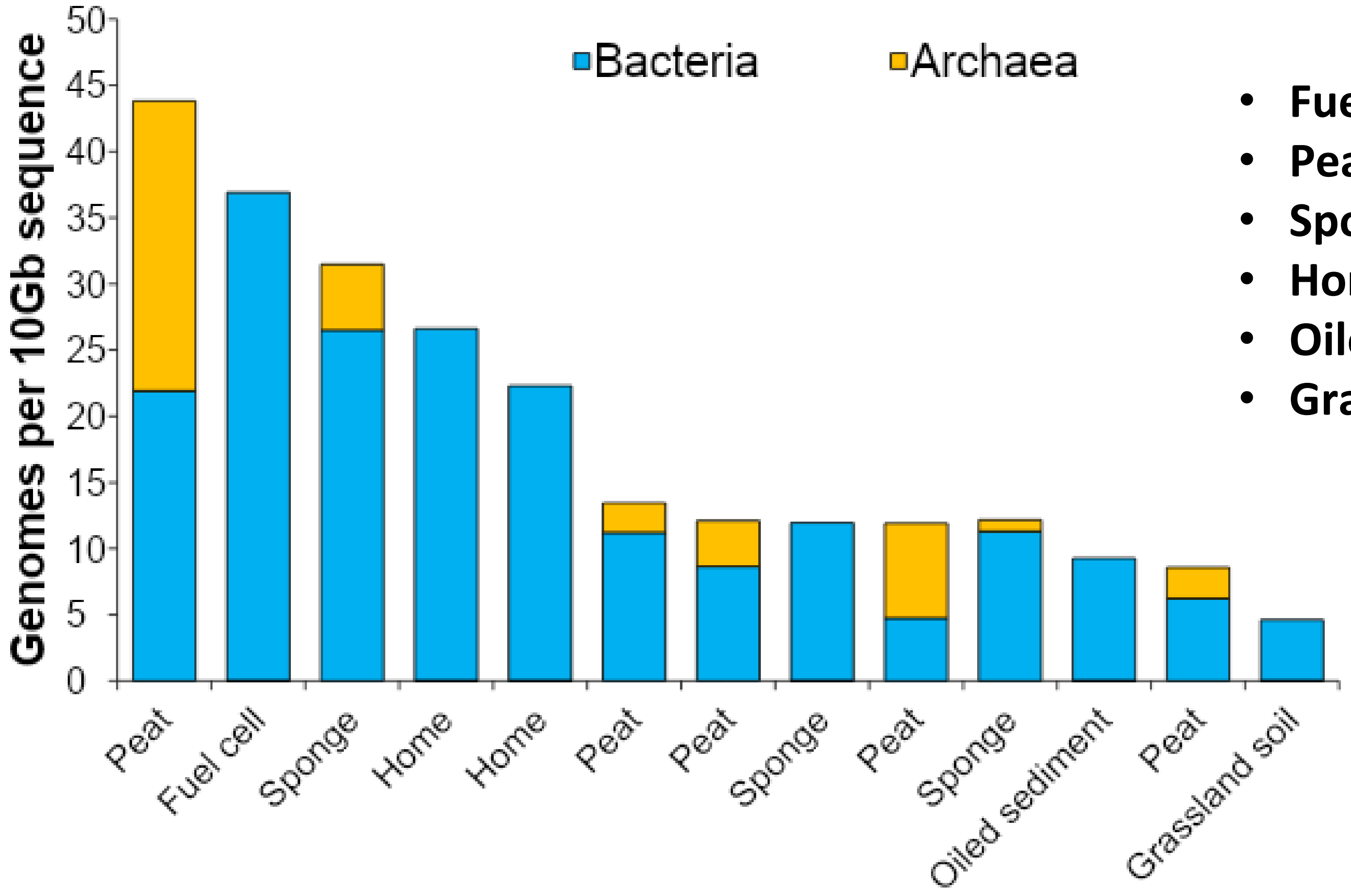
(e.g. 200 bp DNA fragments for 2x125 bp reads)



(e.g. >=550 bp DNA fragments)

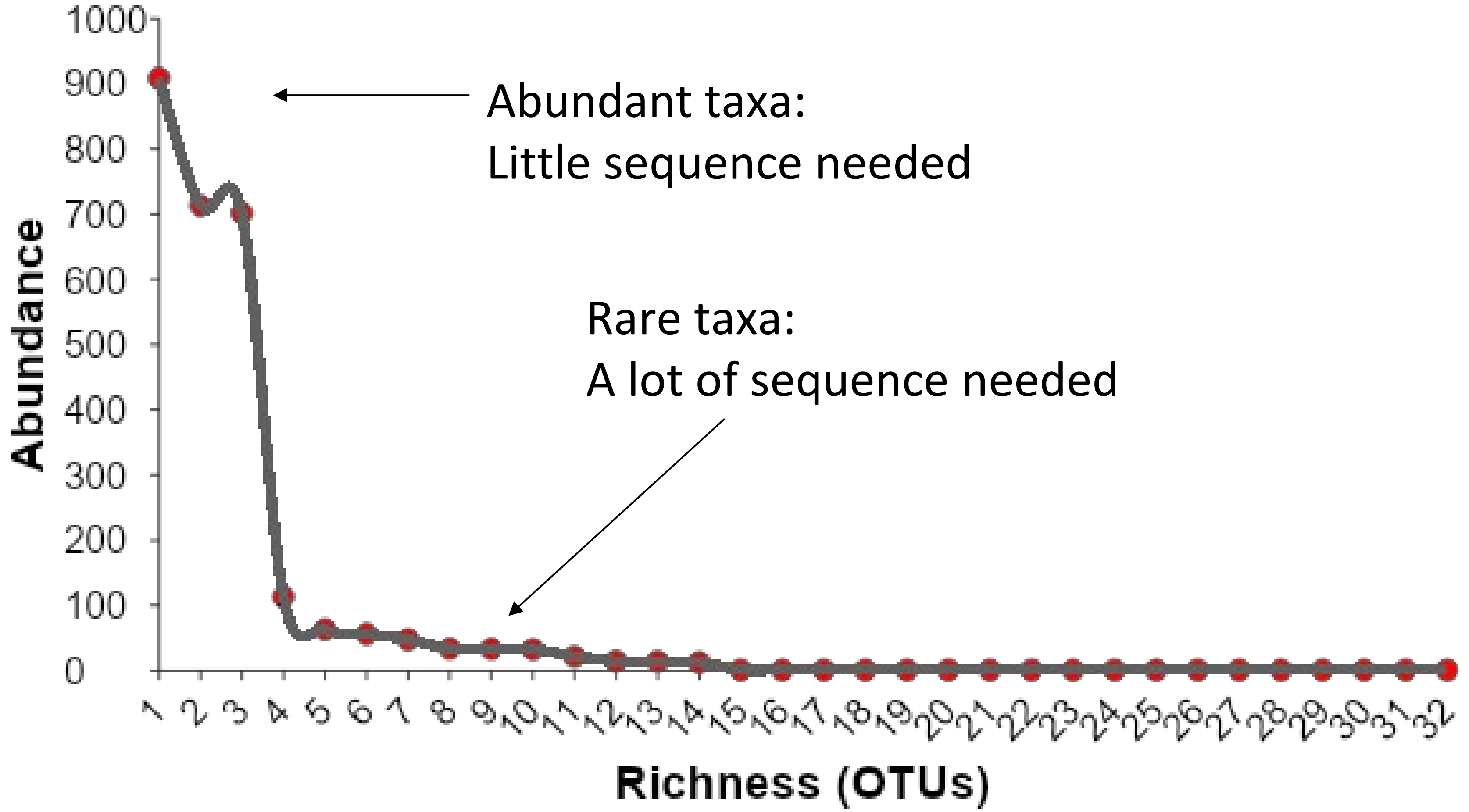


Genome recovery per environment



- Fuel cell microbiome
- Peat (boreal)
- Sponge microbiome
- Home microbiome
- Oiled sediment (seafloor)
- Grassland soil

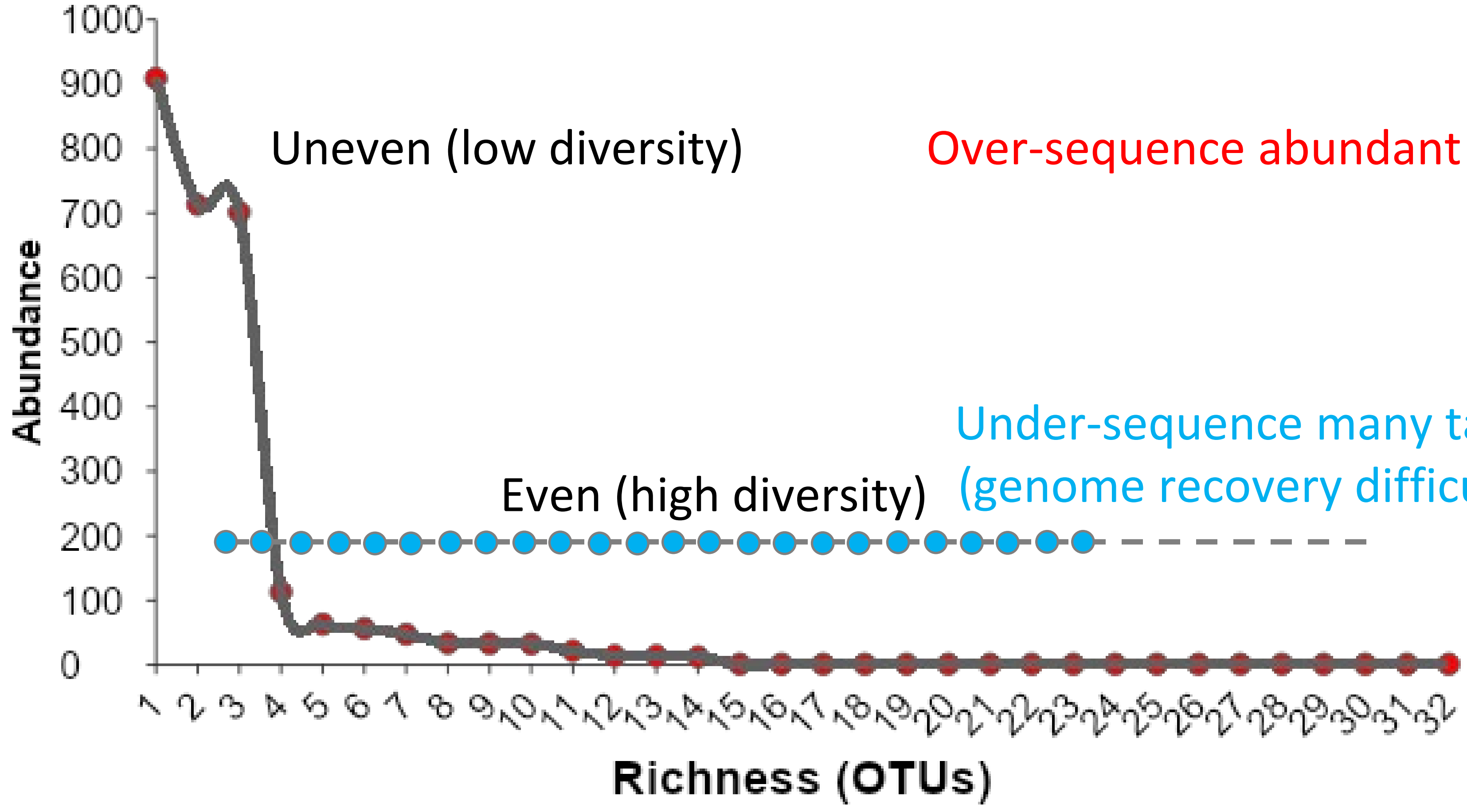
Estimate sequencing depth



Abundant taxa:
Little sequence needed

Rare taxa:
A lot of sequence needed

Community structure matters

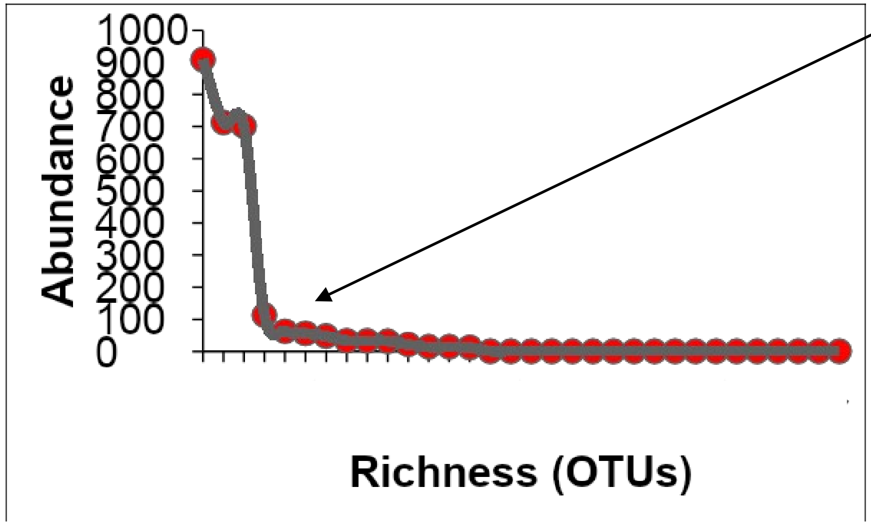


Over-sequence abundant taxa

Under-sequence many taxa (genome recovery difficult)

Estimate sequencing depth

- **Estimate generously**
- **Determine/guesstimate relative abundance of rarest target organism**
- **Determine/guesstimate the average genome size**
- **Factor in larger eukaryote genomes**
- **Decide the minimum desired coverage (e.g. 30x)**



e.g., 5% relative abundance = 5% of sequence data

Prokaryotic genome sizes

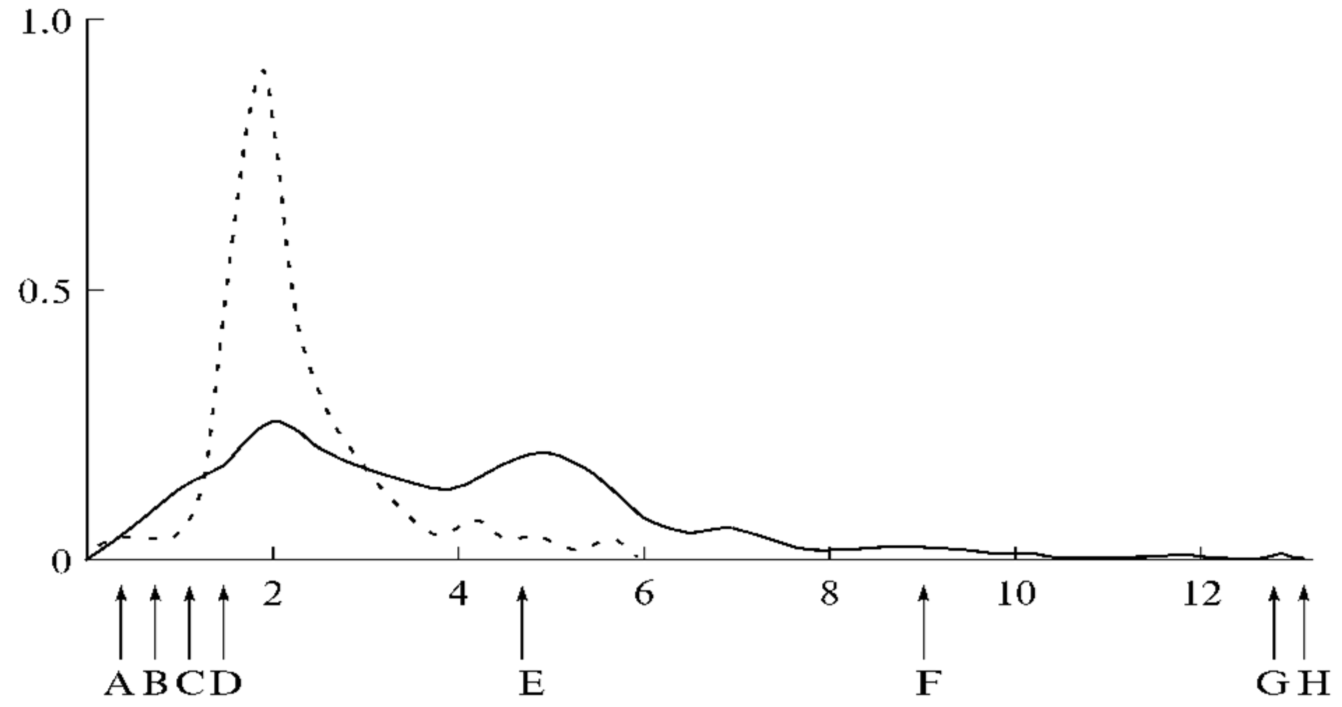
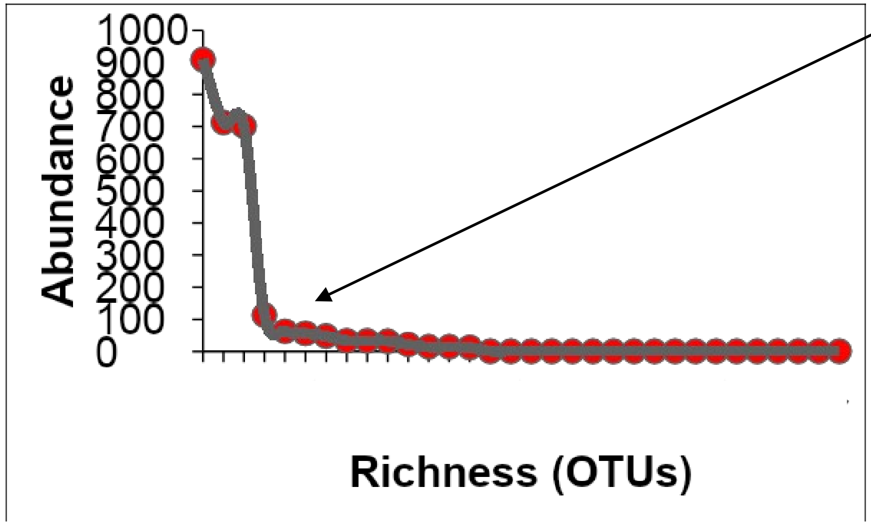


Fig. 1. Ranges of bacterial and archaeal genome sizes. Abscissa shows genome size, Mbp; ordinate shows number of genomes; solid line indicates bacterial genomes; dashed line indicates archaeal genomes; A, *C. ruddii* genome; B, *N. equitans* genome; C, minimal size for free-living microorganisms; D, major peak for genome sizes of bacterial and archaeal genomes; E, minor peak for bacterial genomes; F, *Nostoc punctiforme* genome; G, *Sorangium cellulosum* genome; and H, Van Nimwegen limit.

(Smirnov 2010 Molecular Genetics, Microbiology and Virology)

Estimate sequencing depth

- Estimate generously
- Determine/guesstimate relative abundance of rarest target organism
- Determine/guesstimate the average genome size
- Factor in larger eukaryote genomes
- Decide the minimum desired coverage (e.g. 30x)



e.g., 5% relative abundance = 5% of sequence data

Mock parameters:

- **Bacterial genome 5 Mbp long**
- **5% abundance (need 100/5 or 20x)**
- **30x coverage**

5 Mbp x 20 x 30 = 3,000 Mbp (or 3 Gbp)

When you have so many genomes

You need a:

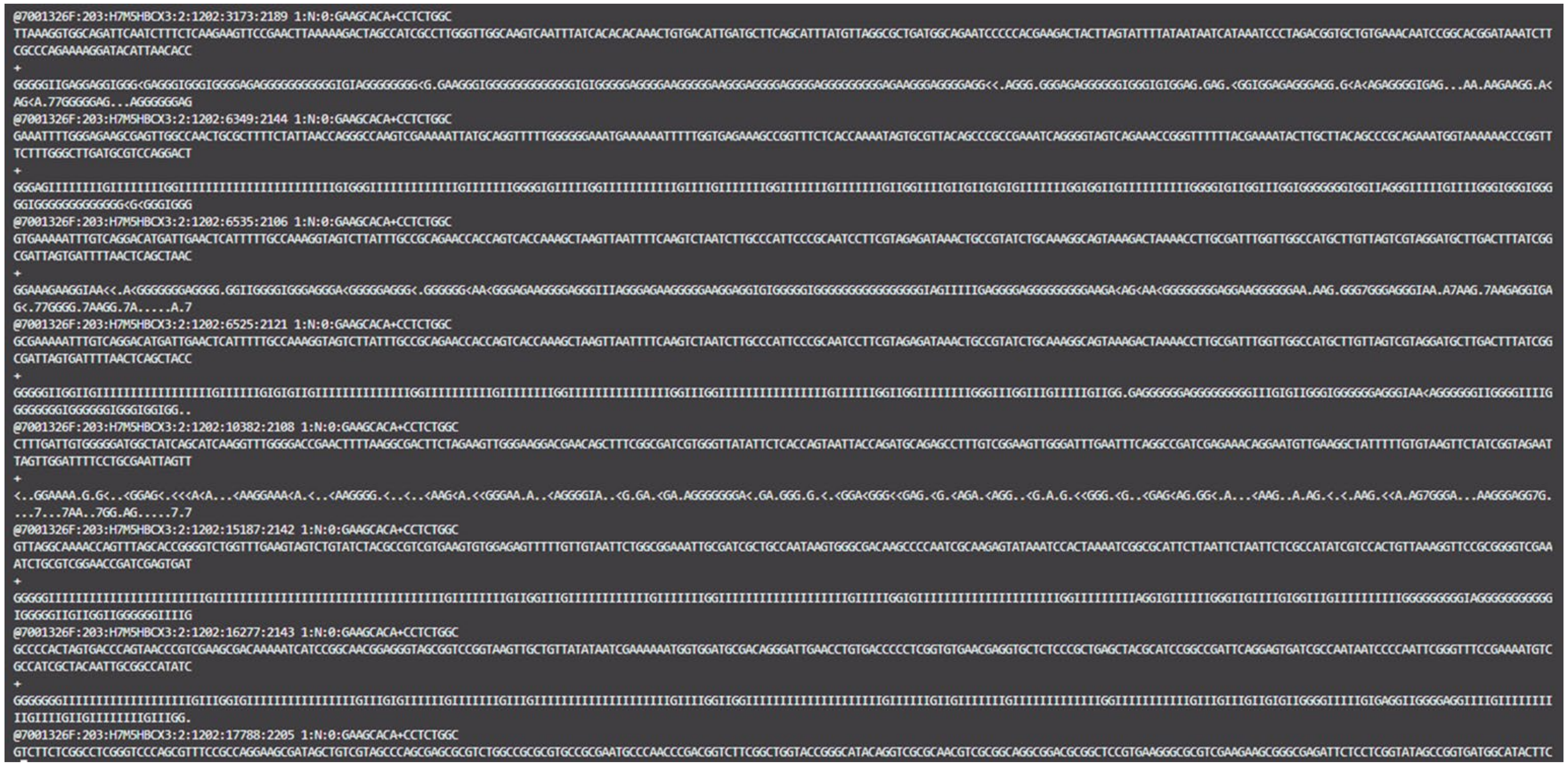
- Clear goal
- Question
- Hypothesis to test

Q&A

What are your research questions?

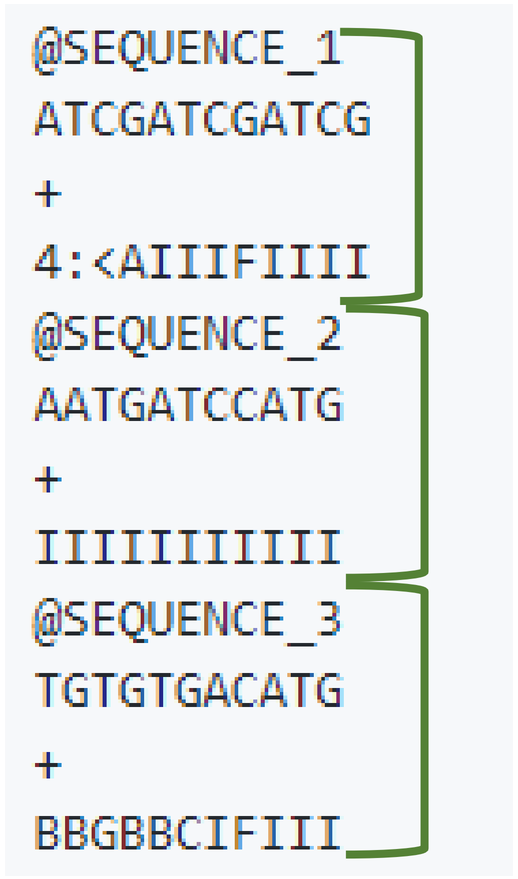
Quality control/filtering raw reads

The FastQ data format



```
@7001326F:203:H7M5HBCX3:2:1202:3173:2189 1:N:0:GAAGCACA+CCTCTGGC
TTAAAGGTGGCAGATTCAATCTTTCTCAAGAAGTTCCGAACTTAAAAAGACTAGCCATCGCCTTGGGTTGGCAAGTCAATTTATCACACACAAACTGTGACATTGATGCTTCAGCATTTATGTTAGGCGCTGATGGCAGAATCCCCCACGAAGACTACTTAGTATTTTATAATAATCATAAATCCCTAGACGGTGCTGTGAAACAATCCGGCACGGATAAATCTT
CGCCCAGAAAAGGATACATTAACACC
+
GGGGGIIGAGGAGGIGGG<GAGGGIGGGIGGGGAGAGGGGGGGGGGGIGIAGGGGGGGG<G.GAAGGGIGGGGGGGGGGGGIGIGGGGGAGGGGAAGGGGGAAGGGAGGGGAGGGGAGGGGGGGGGAGAAGGGAGGGGAGG<<.AGGG.GGGAGAGGGGGGIGGGIGIGGAG.GAG.<GGIGGAGAGGGAGG.G<A<AGAGGGGIGAG...AA.AAGAAGG.A<
AG<A.77GGGGGAG...AGGGGGGAG
@7001326F:203:H7M5HBCX3:2:1202:6349:2144 1:N:0:GAAGCACA+CCTCTGGC
GAAATTTTGGGAGAAGCGAGTTGGCCAACTGCGCTTTTCTATTAACCAGGGCCAAGTCGAAAAATTATGCAGGTTTTTGGGGGGAAATGAAAAAATTTTTGGTGAGAAAGCCGGTTTCTCACCAAAATAGTGCGTTACAGCCCGCCGAAATCAGGGGTAGTCAGAAACCGGGTTTTTTACGAAAATACTTGCTTACAGCCCGCAGAAATGGTAAAAAACCCGGTT
TCTTTGGGCTTGATGCGTCCAGGACT
+
GGGAGIIIIIIIIGIIIIIIIIGGIIIIIIIIIIIIIIIIIIIIIIIIIGIGGGIIIIIIIIIIIIIGIIIIIIIGGGGIGIIIIIGGIIIIIIIIIIIGIIIIGIIIIIIIGGIIIIIIIGIIIIIIIGIIGGIIIIGIIGIIGIGIGIIIIIIIGGIGGIIGIIIIIIIIIIIGGGGIGIIGGIIIGGIGGGGGGGIGGIIAGGGIIIIIGIIIIGGGIGGGIGGG
GGIGGGGGGGGGGGGG<G<GGGIGGG
@7001326F:203:H7M5HBCX3:2:1202:6535:2106 1:N:0:GAAGCACA+CCTCTGGC
GTGAAAAATTTGTCAGGACATGATTGAACTCATTTTTGCCAAAGGTAGTCTTATTTGCCGCAGAACCACCAGTCACCAAAGCTAAGTTAATTTTCAAGTCTAATCTTGCCCATTCCCGCAATCCTTCGTAGAGATAAACTGCCGTATCTGCAAAGGCAGTAAAGACTAAAACCTTGCGATTTGGTTGGCCATGCTTGTTAGTCGTAGGATGCTTGACTTTATCGG
CGATTAGTGATTTTAACTCAGCTAAC
+
GGAAAGAAGGIAA<<.A<GGGGGGGAGGGG.GGIIGGGGIGGAGGGA<GGGGAGGG<.GGGGGG<AA<GGGAGAAGGGGAGGGIIIAGGGAGAAGGGGAAGGAGGIGIGGGGGIGGGGGGGGGGGGGGGIAGIIIIIGAGGGGAGGGGGGGGGAAGA<AG<AA<GGGGGGGGAGGAAGGGGGGAA.AAG.GGG7GGGAGGGIAA.A7AAG.7AAGAGGIGA
G<.77GGGG.7AAGG.7A.....A.7
@7001326F:203:H7M5HBCX3:2:1202:6525:2121 1:N:0:GAAGCACA+CCTCTGGC
GCGAAAAATTTGTCAGGACATGATTGAACTCATTTTTGCCAAAGGTAGTCTTATTTGCCGCAGAACCACCAGTCACCAAAGCTAAGTTAATTTTCAAGTCTAATCTTGCCCATTCCCGCAATCCTTCGTAGAGATAAACTGCCGTATCTGCAAAGGCAGTAAAGACTAAAACCTTGCGATTTGGTTGGCCATGCTTGTTAGTCGTAGGATGCTTGACTTTATCGG
CGATTAGTGATTTTAACTCAGCTACC
+
GGGGGIIGGIIGIIIIIIIIIIIIIIIIIGIIIIIIGIGIGIIGIIIIIIIIIIIIIIIGGIIIIIIIIIIIGIIIIIIIIGGIIIIIIIIIIIIIIIIGGIIIGGIIIIIIIIIIIIIIIIIGIIIIIIGGIIGGIIIIIIIIIGGGIIIGGIIIGIIIIIGIIGG.GAGGGGGGAGGGGGGGGGIIIGIGIIGGGIGGGGGGAGGGIAA<AGGGGGGIIGGGGIIIIG
GGGGGGGIGGGGGGIGGGIGGIGG..
@7001326F:203:H7M5HBCX3:2:1202:10382:2108 1:N:0:GAAGCACA+CCTCTGGC
CTTTGATTGTGGGGGATGGCTATCAGCATCAAGGTTTGGGGACCGAACTTTTAAGGCGACTTCTAGAAGTTGGGAAGGACGAACAGCTTTCGGCGATCGTGGGTTATATTCTCACCAGTAATTACCAGATGCAGAGCCTTTGTCGGAAGTTGGGATTTGAATTTCAGGCCGATCGAGAAACAGGAATGTTGAAGGCTATTTTTGTGTAAGTTCTATCGGTAGAAT
TAGTTGGATTTTCCTGCGAATTAGTT
+
<..GGAAAA.G.G<..<GGAG<.<<<A<A...<AAGGAAA<A.<..<AAGGGG.<..<..<AAG<A.<<GGGAA.A..<AGGGGIA..<G.GA.<GA.AGGGGGGGA<.GA.GGG.G.<.<GGA<GGG<<GAG.<G.<AGA.<AGG..<G.A.G.<<GGG.<G..<GAG<AG.GG<.A...<AAG..A.AG.<.<.AAG.<<A.AG7GGGA...AAGGGAGG7G.
...7...7AA..7GG.AG.....7.7
@7001326F:203:H7M5HBCX3:2:1202:15187:2142 1:N:0:GAAGCACA+CCTCTGGC
GTTAGGCAAAACCAGTTTAGCACCGGGGTCTGGTTTGAAGTAGTCTGTATCTACGCCGTCGTGAAGTGTGGAGAGTTTTTGTTGTAATTCTGGCGGAAATTGCGATCGCTGCCAATAAGTGGGCGACAAGCCCCAATCGCAAGAGTATAAATCCACTAAAATCGGCGCATTCTTAATTCTAATTCTCGCCATATCGTCCACTGTTAAAGGTTCCGCGGGGTCGAA
ATCTGCGTCGGAACCGATCGAGTGAT
+
GGGGGIIIIIIIIIIIIIIIIIIIIIIIGIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIGIIIIIIIIGIIGGIIIGIIIIIIIIIIIIIGIIIIIIIGGIIIIIIIIIIIIIIIIIIIIIIGIIIIIGGIGIIIIIIIIIIIIIIIIIIIIIIIGGIIIIIIIIIIAGGIGIIIIIIGGGIIGIIIIGIGGIIIGIIIIIIIIIIIGGGGGGGGGIAGGGGGGGGGGG
IGGGGGIIGIIGGIIGGGGGGIIIIG
@7001326F:203:H7M5HBCX3:2:1202:16277:2143 1:N:0:GAAGCACA+CCTCTGGC
GCCCCACTAGTGACCCAGTAACCCGTCGAAGCGACAAAAATCATCCGGCAACGGAGGGTAGCGGTCCGGTAAGTTGCTGTTATATAATCGAAAAAATGGTGGATGCGACAGGGATTGAACCTGTGACCCCCTCGGTGTGAACGAGGTGCTCTCCCGCTGAGCTACGCATCCGGCCGATTCAGGAGTGATCGCCAATAATCCCCAATTCGGGTTTCCGAAAATGTC
GCCATCGCTACAATTGCGGCCATATC
+
GGGGGGGIIIIIIIIIIIIIIIIIIIGIIIGGIGIIIIIIIIIIIIIIIIIGIIIGIGIIIIIIGIIIIIIIGIIIGIIIIIIIIIIIIIIIIIIIIIIGIIIIGGIIGGIIIIIIIIIIIIIIIIIIIIIIGIIIIIIGIIGIIIIIIIGIIIIIIIIIIIIIIIGGIIIIIIIIIIIIGIIIGIIIGIIGIGIIGGGGIIIIIGIGAGGIIGGGGAGGIIIIGIIIIIIII
IIGIIIIGIIGIIIIIIIIIGIIIGG.
@7001326F:203:H7M5HBCX3:2:1202:17788:2205 1:N:0:GAAGCACA+CCTCTGGC
GTCTTCTCGGCCTCGGGTCCCAGCGTTTCCGCCAGGAAGCGATAGCTGTCGTAGCCCAGCGAGCGCGTCTGGCCGCGCGTGCCGCGAATGCCCAACCCGACGGTCTTCGGCTGGTACCGGGCATACAGGTCGCGCAACGTCGCGGCAGGCGGACGCGGCTCCGTGAAGGGCGCGTCGAAGAAGCGGGCGAGATTCTCCTCGGTATAGCCGGTGATGGCATACTTC
```

The FastQ data format



```
@SEQUENCE_1
ATCGATCGATCG
+
4:<AIIIFIIII
@SEQUENCE_2
AATGATCCATG
+
IIIIIIIIIII
@SEQUENCE_3
TGTGTGACATG
+
BBGBBCIFIII
```

Each sequence is represented by four lines

1. Sequence name
2. Sequence content
3. Spacer line (+, or +Sequence name)
4. Quality information

The FastQ data format

- **What does the quality score even mean?**
  - **It represents the probability of a nucleotide position being *<u>incorrectly</u>* called**

$$Q = -10\, log_{10} p$$

Q	p	Prob. correct
0	1	0
10	0.1	0.9
20	0.01	0.99
30	0.001	0.999
40	0.0001	0.9999

How each Q value is encoded varies between sequencing platforms

<u>Generally</u> we work with the **Illumina 1.8+** ([Phred+33]) standard

The FastQ data format

- **What does the quality score even mean?**
  - **It represents the probability of a nucleotide position being *incorrectly* called**

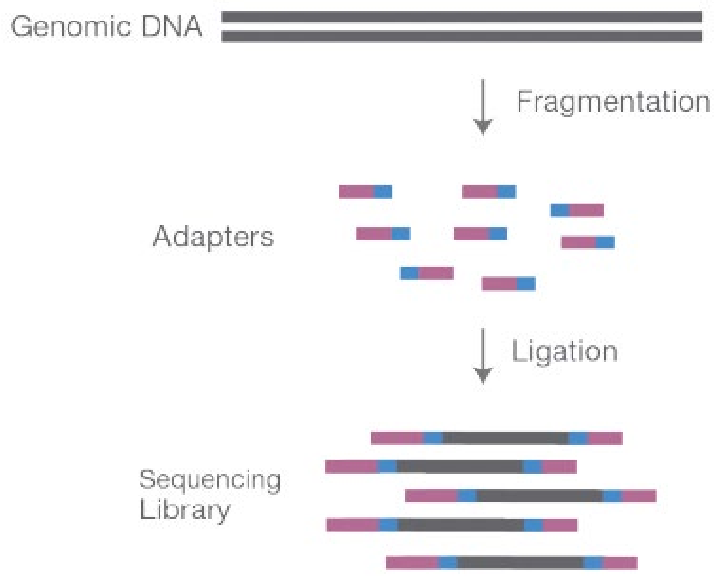
$$Q = -10\ log_{10}p$$

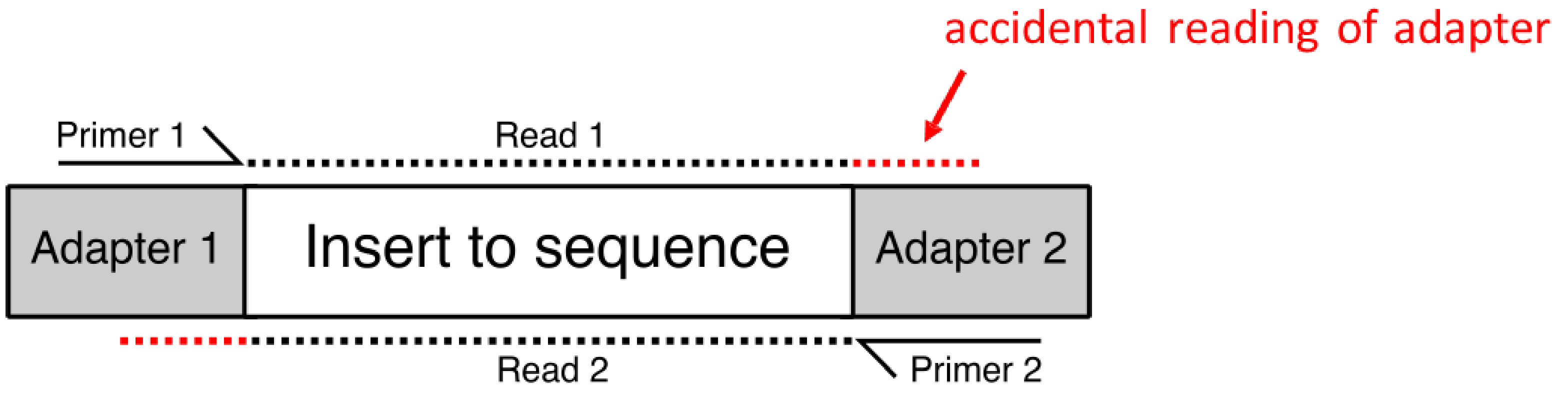
Q	p	Prob. correct
0	1	0
10	0.1	0.9
20	0.01	0.99
30	0.001	0.999
40	0.0001	0.9999

How each Q value is encoded varies between sequencing platforms

```
(33):  !"#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHI
```

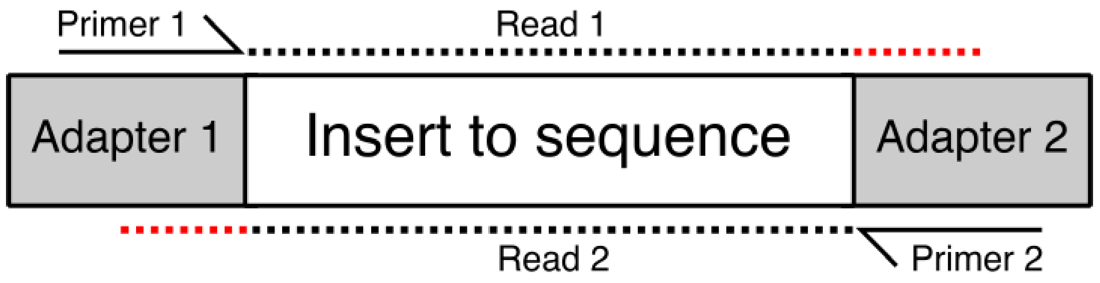
Quality filtering WGS data





Quality filtering WGS data

- Identify potential problems that occurred during sequencing
  - e.g. Adapter read-through
  - e.g. Rapid drop off in sequence quality
- Remove barcode and adapter regions
- Remove low-quality regions of reads



Task: Quality filtering

Go to Github MGSS webpage

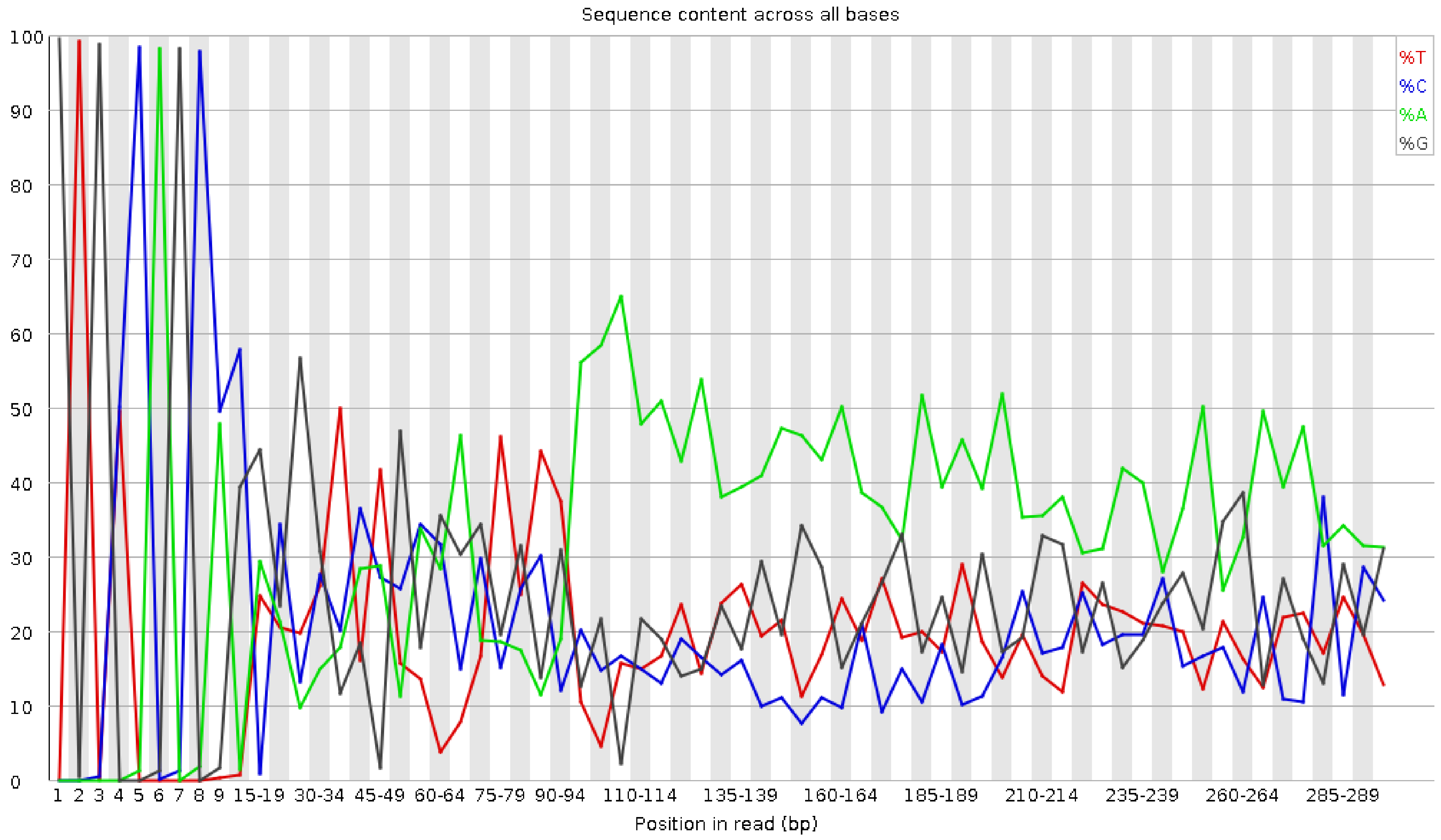
Tasks:

- **Visualisation with FastQC**
  - Inspecting FASTQ files
  - Identifying regions of concern
- **Read trimming and adapter removal with Trimmomatic**
  - Removing adapter sequences
  - Removing low-quality regions
- **Diagnosing poor libraries**
- **(Optional) Filtering out host DNA**

Diagnosing poor libraries

- Does the **sequencing length** match what you ordered from the facility?
- If the sequences are shorter than expected, is adapter read-through a concern?
- What does the **sequence quality** look like for the whole length of the run? Are there any expected/unexpected regions of quality degradation?
- Are **adapters and/or barcodes removed**?
  - Look at the Per base sequence content to diagnose this.
- Is there **unexpected sequence duplication**?
  - This can occur when low-input library preparations are used.
- Are **over-represented k-mers** present?
  - This can be a sign of adapter and barcode contamination.

Common issues with WGS data



Common issues with WGS data

Do I need to remove adapters?	Yes.
I don't know if adapters have been removed or not	Check the per-nucleotide distributions You will see 100% skews if they remain.
What's the lowest Q to allow when trimming?	Assembly is a self-correcting process, so you can be surprisingly lenient.
What if my GC skew is outside of the expected range?	FastQC is calibrated to genome data where you expect GC conservation. Metagenomes do not adhere to this assumption.

Filtering out host DNA

Metagenome data derived from microbial communities associated with a host should ideally be filtered to remove any reads originating from host DNA. This may improve the quality and efficiency of downstream data processing

Important for submission to databases e.g. NCBI

- Ethics for human host DNA
- Taonga species in Aotearoa

Task: Quality filtering

Go to Github MGSS webpage

Tasks:

✓ - **Visualisation with *FastQC***
    - Inspecting FASTQ files
    - Identifying regions of concern

✓ - **Read trimming and adapter removal with Trimmomatic**
    - Removing adapter sequences
    - Removing low-quality regions

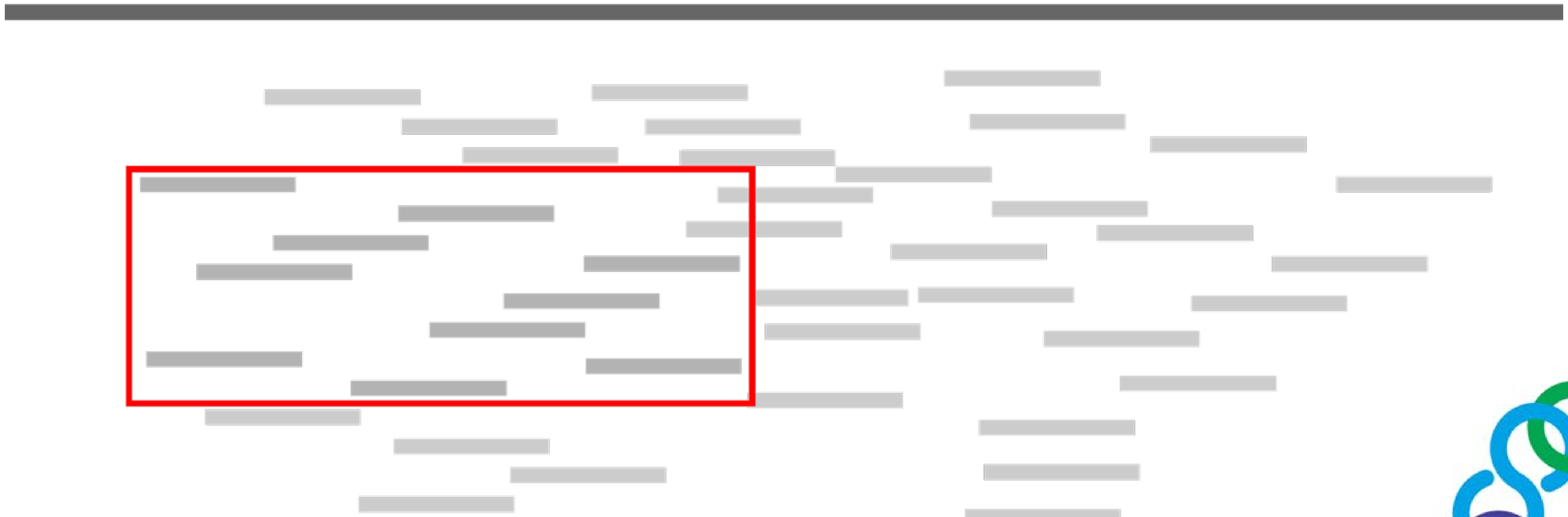
✓ - **Diagnosing poor libraries**

- **(Optional) Filtering out host DNA**

Assembly

Genome assembly

Overlap-Consensus-Layout (OCL) assembly



Genome assembly

Overlap-Consensus-Layout (OCL) assembly

```
TTGAAGAGTT
GGCTCAGATT
TTTGATCATG
AACGCTGGCG
AAGAGTTTGA
GATTGAACGC
CTCAGATTGA
TGAAGAGTTT
ACGCTGGCGC
TCATGGCTCA
```

Genome assembly

Overlap-Consensus-Layout (OCL) assembly

```
TTGAAGAGTTTGATCATGGCTCAGATTGAACGCTGGCGC
TTGAAGAGTT       GGCTCAGATT AACGCTGGCG
        TTTGATCATG     GATTGAACGC
   AAGAGTTTGA         CTCAGATTGAACGCTGGCGC
 TGAAGAGTTT   TCATGGCTCA
```

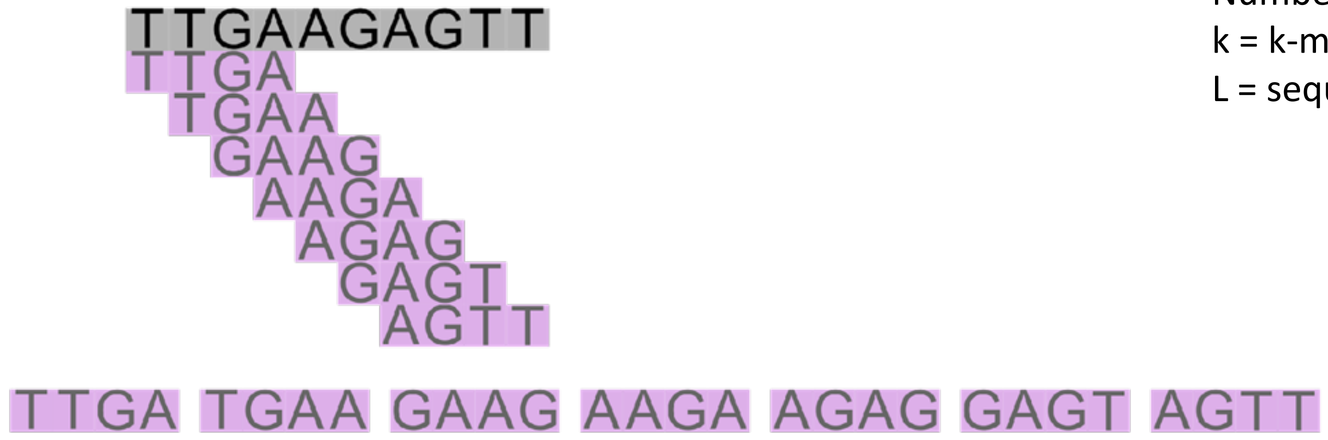
The problem for *de novo* assembly?

$$N.comparisons = \frac{(n)(n-1)}{2} = \frac{(10)(10-1)}{2} = 45$$

Genome assembly

De Bruijn graph assembly

Break reads into shorter *k*-mers

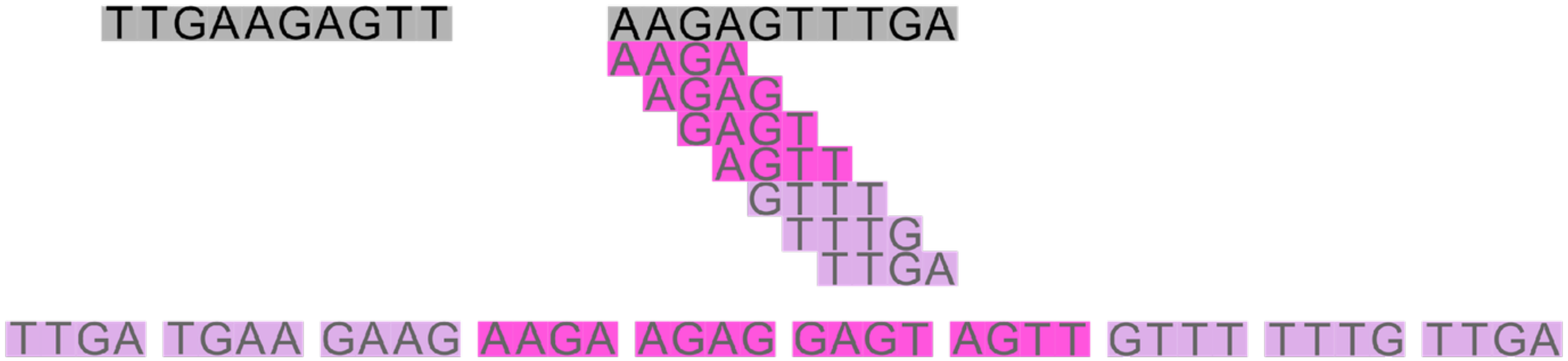


Number kmers per sequence = (L – k) + 1
k = k-mer length
L = sequence length

Genome assembly

De Bruijn graph assembly

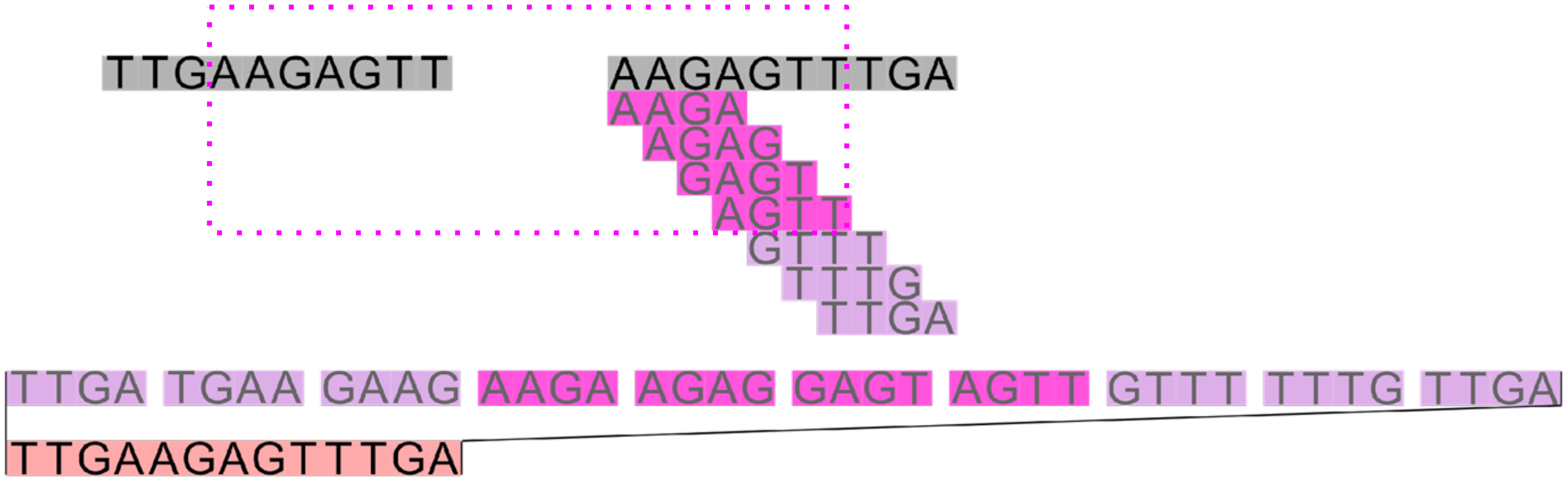
Identify sequences of shared *k*-mers



Genome assembly

De Bruijn graph assembly

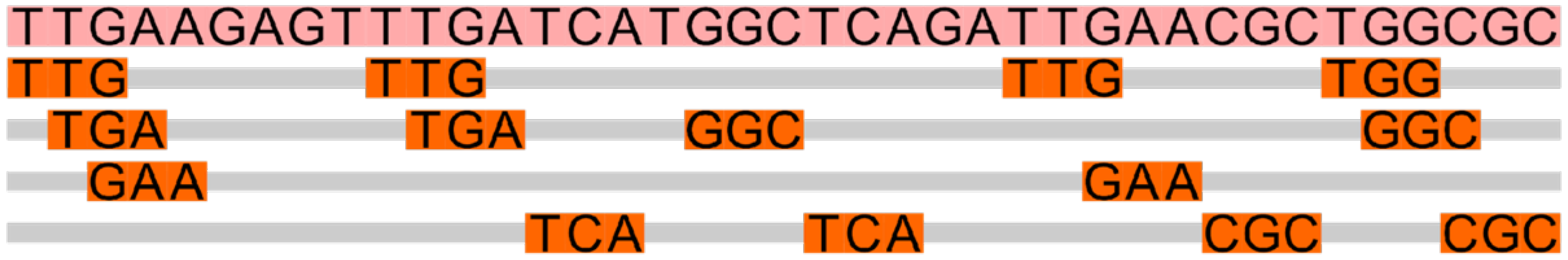
Identify sequences of shared *k*-mers



Genome assembly

De Bruijn graph assembly

Problem #1 – *k*-mers are short?

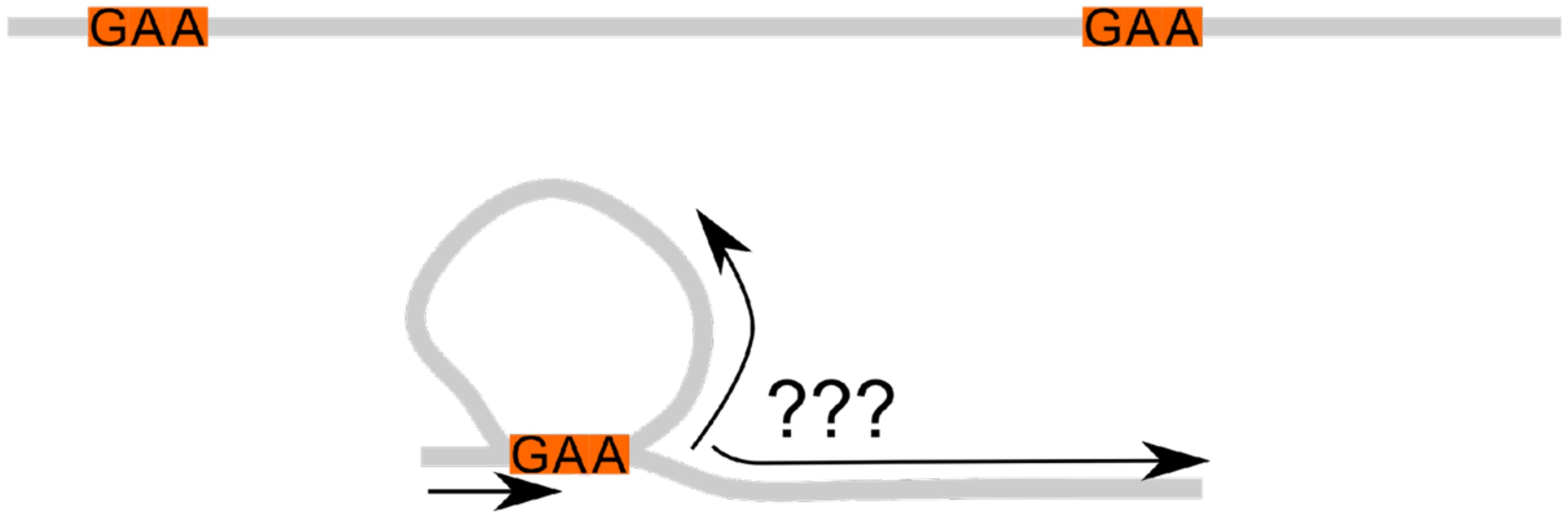


Genome assembly

De Bruijn graph assembly

Problem #1 – *k*-mers are short?

TTGAAGAGTTTGATCATGGCTCAGATTGAACGCTGGCGC



Genome assembly

De Bruijn graph assembly

Problem #2 – *k*-mers are long?

```
TTGAAGAGTT          AAGAGTTTGA
TTGAAGAG            AAGAGTTT
 TGAAGAGT            AGAGTTTG
  GAAGAGTT            GAGTTTGA
```

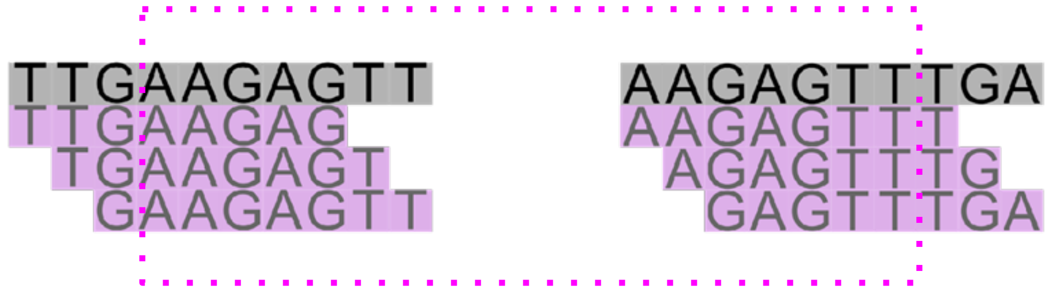
```
TTGAAGAG TGAAGAGT GAAGAGTT

    AAGAGTTT AGAGTTTG GAGTTTGA
```

Genome assembly

De Bruijn graph assembly

Problem #2 – *k*-mers are long?



TTGAAGAG TGAAGAGT GAAGAGTT

AAGAGTTT AGAGTTTG GAGTTTGA

De Bruijn graph assembly

We want a range of *k*-mer sizes

- Short *k*-mers yield higher coverage
- Long *k*-mers assemble longer contigs (jump repeat regions)

Other considerations for picking *k*-mer sizes

- Size cannot be longer than read length
- Always pick odd *k*-mer sizes
- The more sizes you use, the longer assembly will take

K-mers	N. contigs	Longest contig	N50 >2kbp	L50 >2kbp
21, 33, 55	4,239,806	660,812	6,782	12,906
43, 55, 77, 99, 121	2,519,669	1,022,083	7,990	12,673
21, 43, 55, 77, 99, 121	3,388,682	1,022,083	7,789	13,327

De Bruijn graph assembly

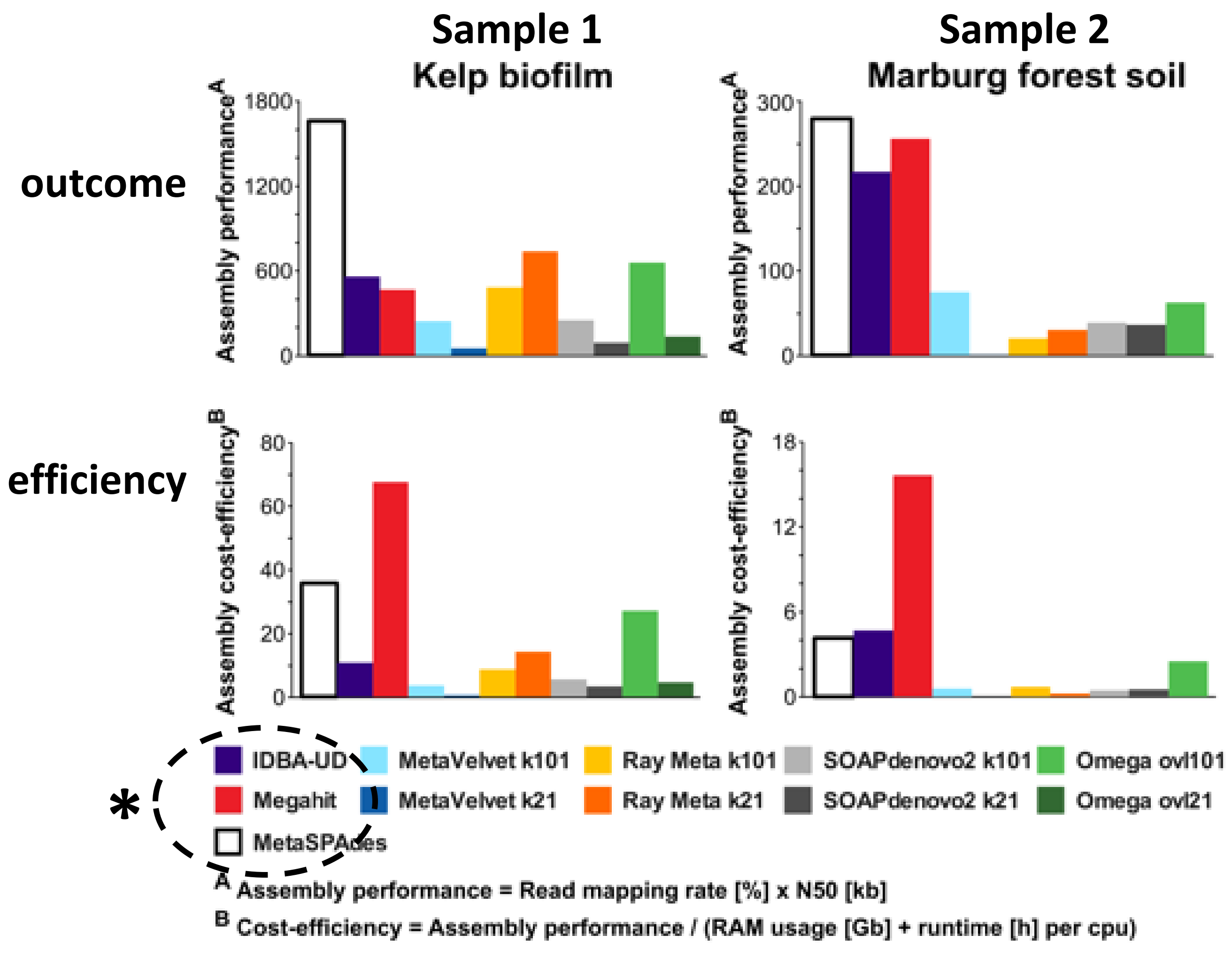
We want a range of *k*-mer sizes

- Short *k*-mers yield higher coverage

K-mers	N. contigs	Longest contig	N50 >2kbp	L50 >2kbp
21, 33, 55	4,239,806	660,812	6,782	12,906
43, 55, 77, 99, 121	2,519,669	1,022,083	7,990	12,673
21, 43, 55, 77, 99, 121	3,388,682	1,022,083	7,789	13,327

- Size cannot be longer than read length
- Always pick odd *k*-mer sizes
- The more sizes you use, the longer assembly will take

Which assembler is best?



(Vollmers et al., 2017, PLOS ONE)

Outcomes vary by dataset.

Assembly optimization generally requires empirically testing:

- Assemblers
- Parameters

Which assembler is best?

There are three good options

- SPAdes
- MegaHIT
- IDBA-UD

Which assembler is best?

There are three good options

- SPAdes
- MegaHIT
- IDBA-UD

In conclusion, it can be said that the choice of assembler should depend on the data at hand and on the exact research question asked. Generally, the best assembly is performed by multi k-mer assemblers such as metaSPAdes, Megahit and IDBA-UD. If micro diversity is not a major issue, and the primary research goal is to bin and reconstruct representative bacterial genomes from a given environment, metaSPAdes should clearly be the assembler of choice. This assembler yields the best contig size statistics while capturing a high degree of community diversity, even at high complexity and low read coverage. If mico diversity is however an issue, or if the degree of captured diversity is far more important than contig lengths, then IDBA-UD or Megahit should be preferred.

Vollmers *et al.* 2017 (https://doi.org/ 10.1371/journal.pone.0169662)

Which assembler is best?

There are three good options

- SPAdes
- MegaHIT
- IDBA-UD

In conclusion, it can be said that the choice of assembler should depend on the data at hand and on the exact research question asked. Generally, the best assembly is performed by multi k-mer assemblers such as metaSPAdes, Megahit and IDBA-UD. If micro diversity is not a major issue, and the primary research goal is to bin and reconstruct representative bacterial genomes from a given environment, metaSPAdes should clearly be the assembler of choice. This assembler yields the best contig size statistics while capturing a high degree of community diversity, even at high complexity and low read coverage. If mico diversity is however an issue, or if the degree of captured diversity is far more important than contig lengths, then IDBA-UD or Megahit should be preferred.

Vollmers *et al.* 2017 (https://doi.org/ 10.1371/journal.pone.0169662)

What are some key considerations?

Biological

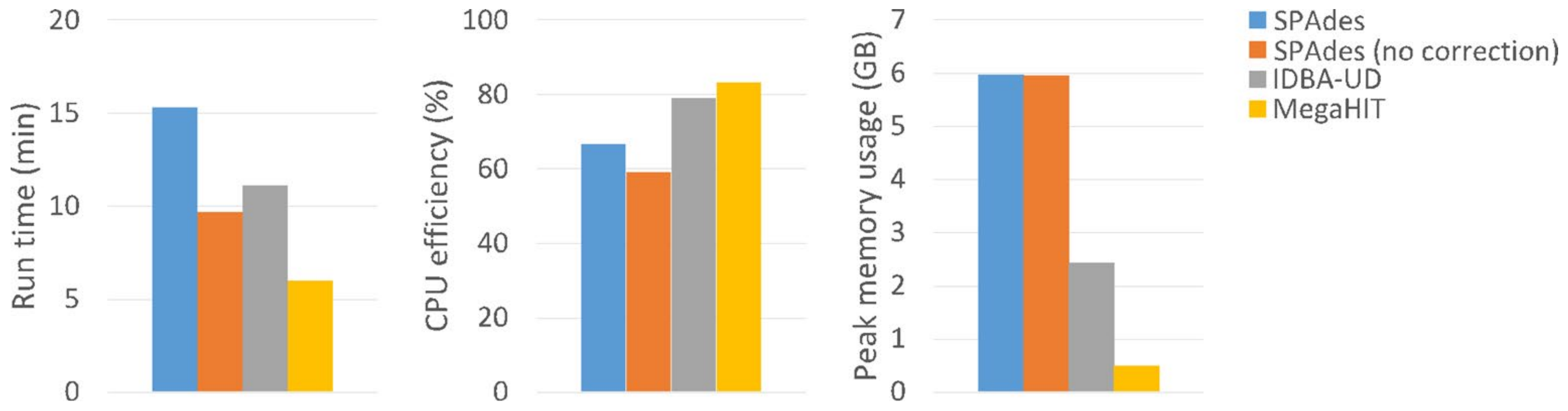
1. What is your hypothesis?
2. What do you want from the data?

Computational and resource

1. How much data do you have?
2. What are your computational resources?
3. What are your ***time*** resources?

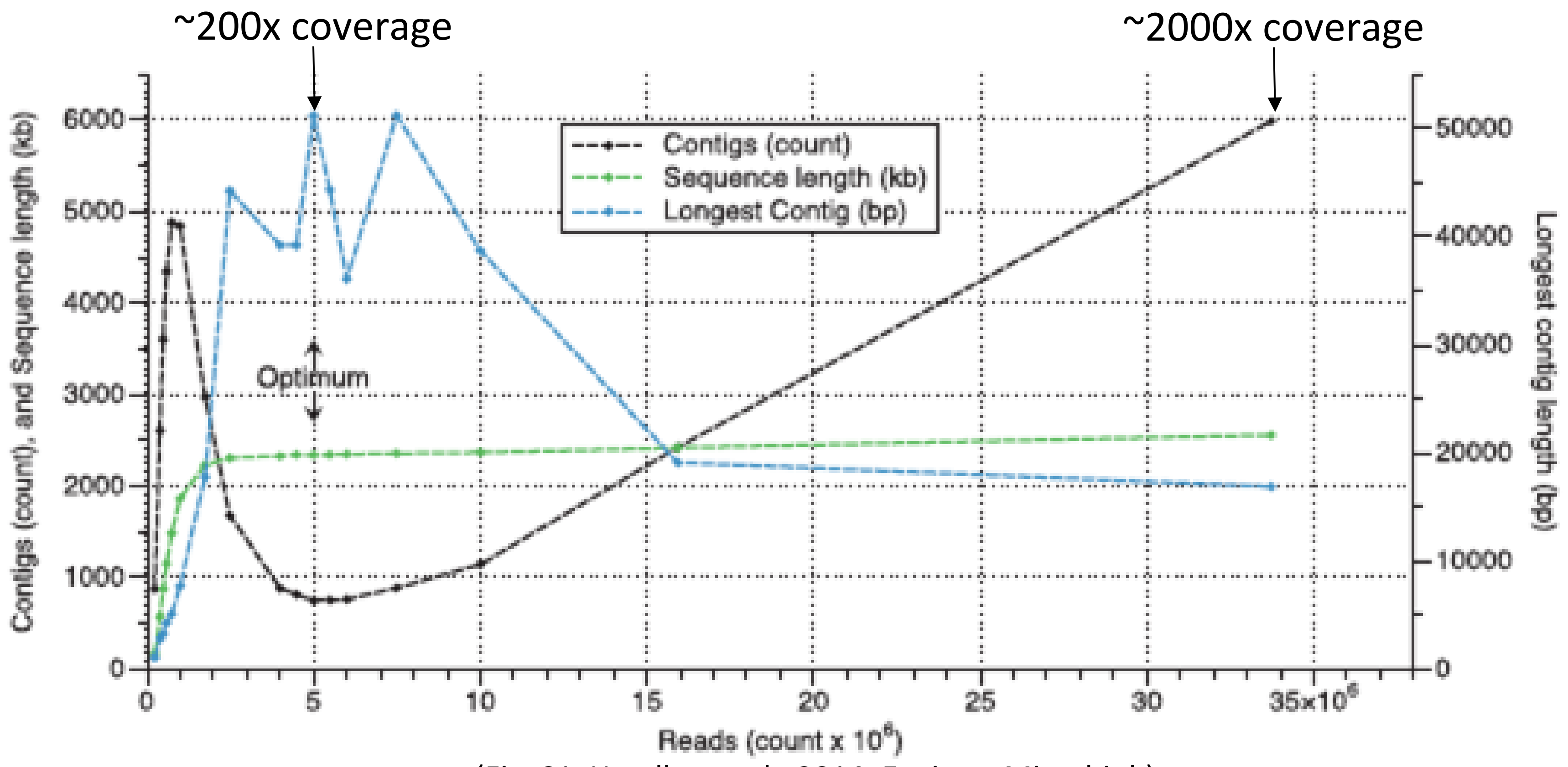
Genome assembly

What are some key considerations?



Too much data?

- Consider testing sub-samples when coverage is very high, e.g. 100s or 1000s
- Example: abundant groundwater genome at 2000x coverage in full dataset
- Empirical testing of subsample sizes identified assembly sweet spot



(Fig. S1, Handley et al., 2014, Environ. Microbiol.)

Task: Assembly

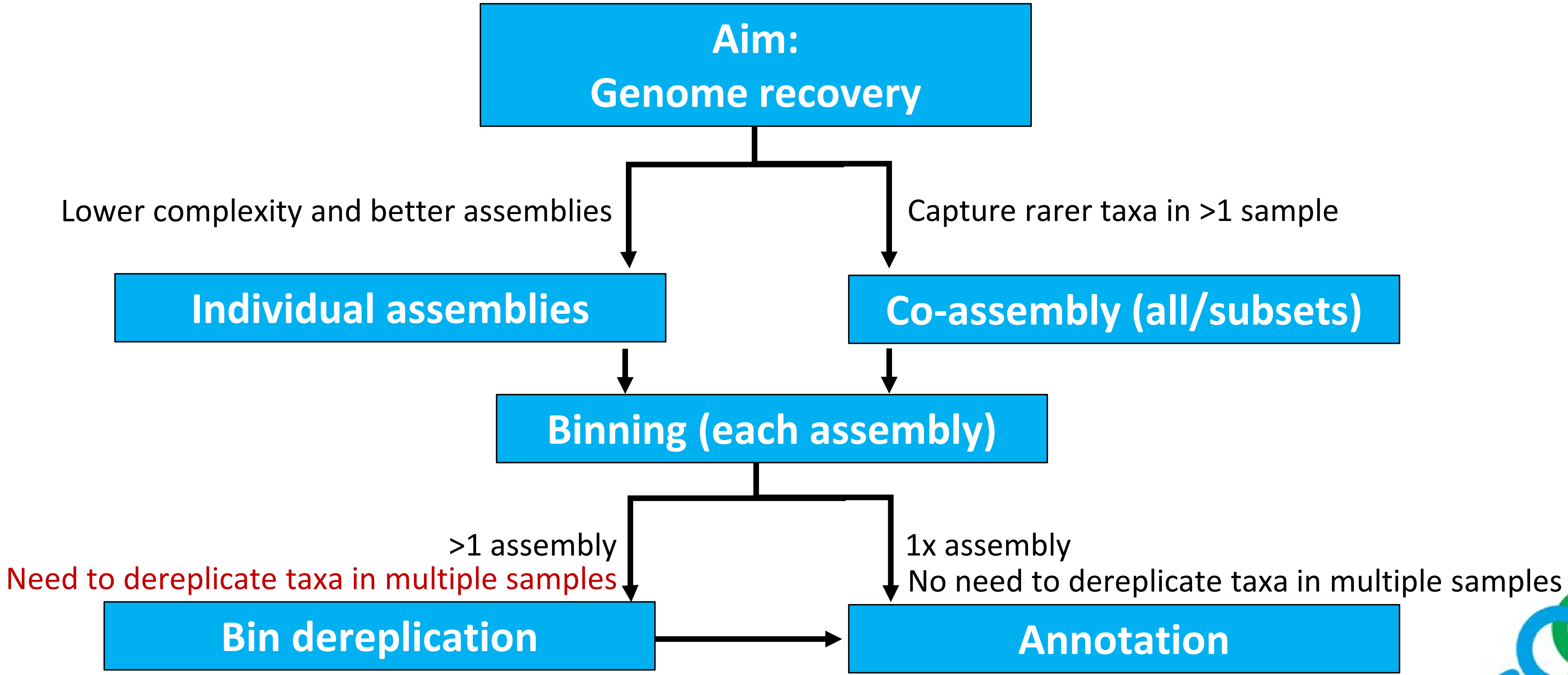
Go to Github MGSS webpage

Tasks:

- **Preparing data for assembly (Run IDBA_UD assembly)**
- **Exploring assembler options**
  - Configure the basic parameters for assembly
- **Submitting jobs to NeSI via slurm**
  - Prepare an assembly job to run under slurm
- **Run SPAdes and IDBA_UD assembly**
- **(Optional) Submitting variant assemblies to NeSI**

Other assembly considerations and Assembly evaluation

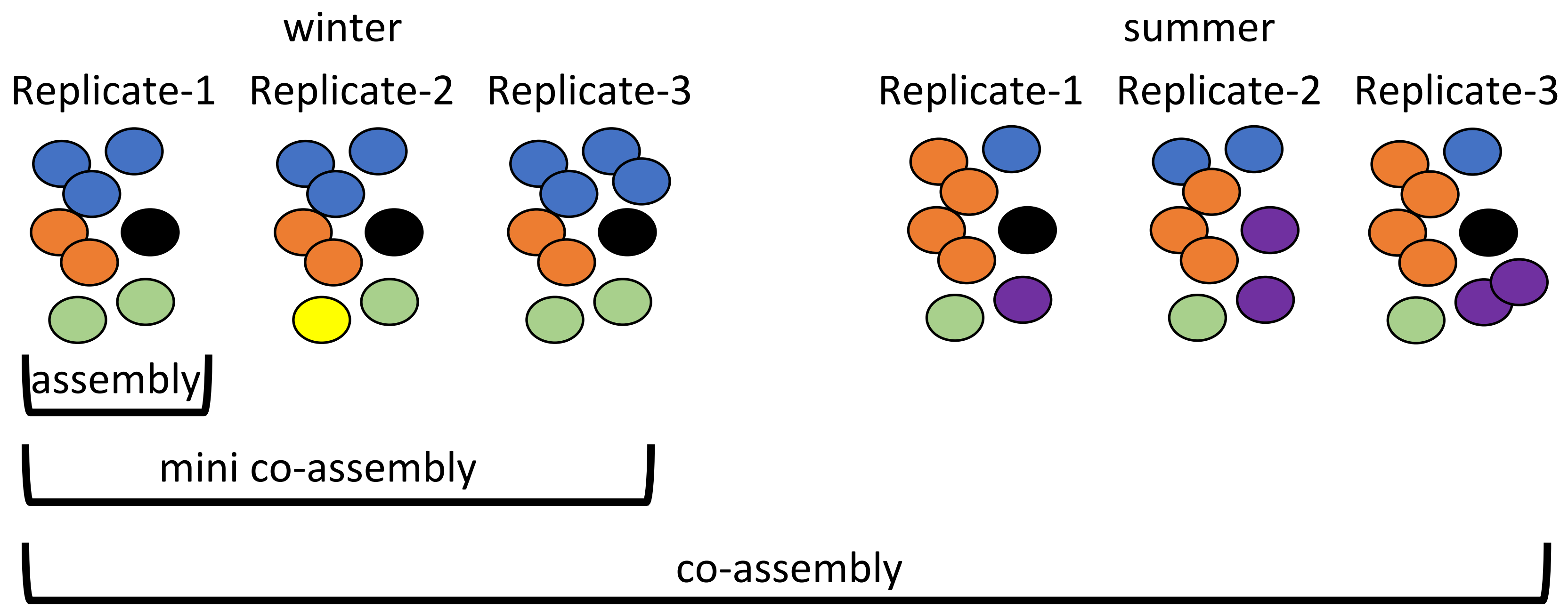
Other assembly considerations


Aim:
Genome recovery
Lower complexity and better assemblies
Capture rarer taxa in >1 sample
Individual assemblies
Co-assembly (all/subsets)
Binning (each assembly)
>1 assembly
Need to dereplicate taxa in multiple samples
1x assembly
No need to dereplicate taxa in multiple samples
Bin dereplication
Annotation

Other assembly considerations

Assembly options:

- **Assemble each community separately**
- **Combine reads and assemble all together (co-assembly)**
- **Combine only reads from the same season (mini co-assemblies)**



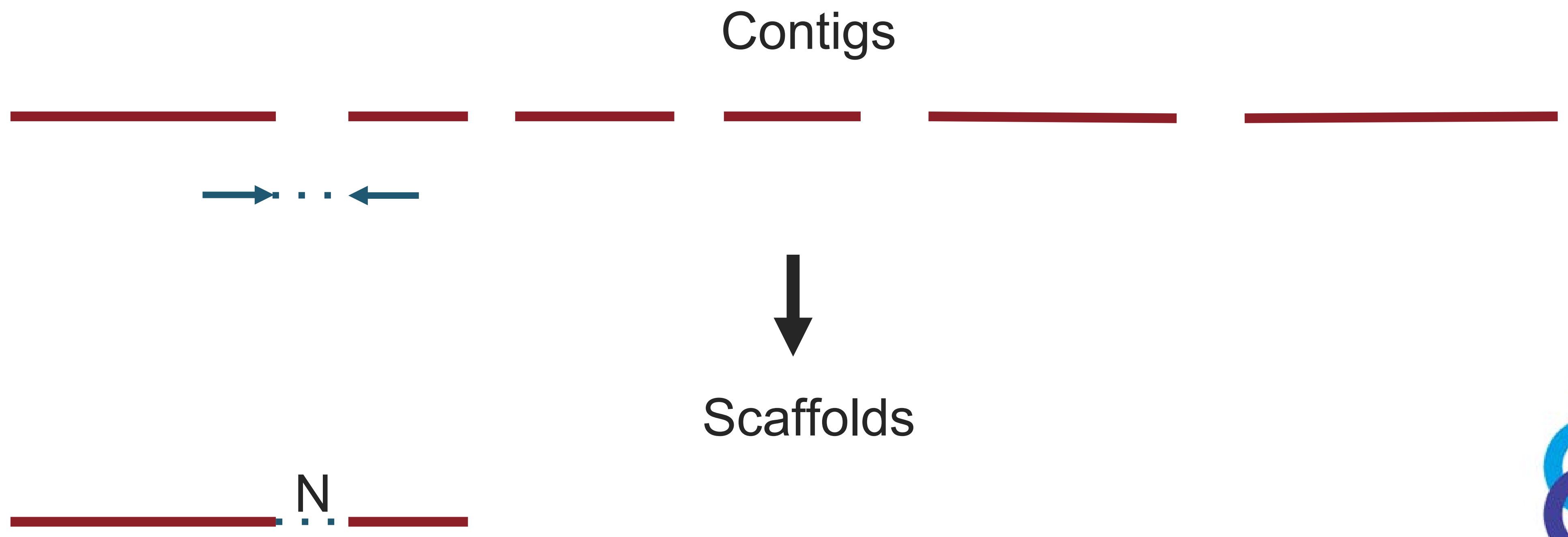
Assembly evaluation

Contigs vs Scaffolds

Assembly evaluation

Contigs vs Scaffolds

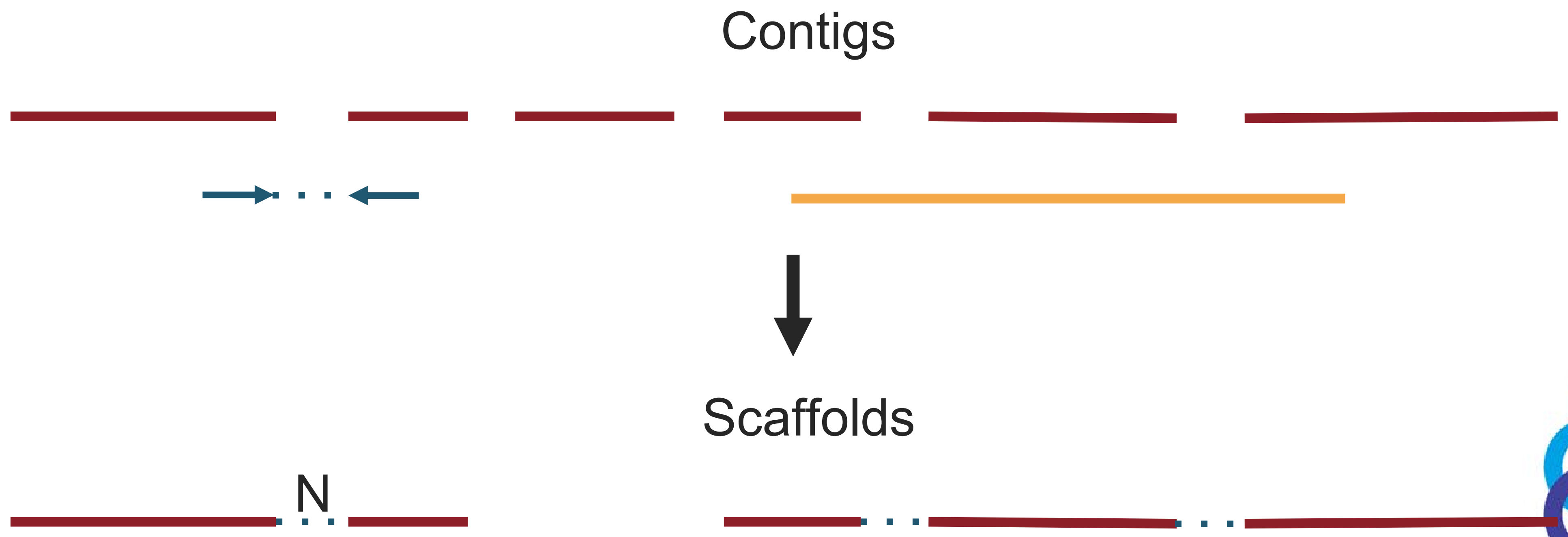
- Overlapping insert



Assembly evaluation

Contigs vs Scaffolds

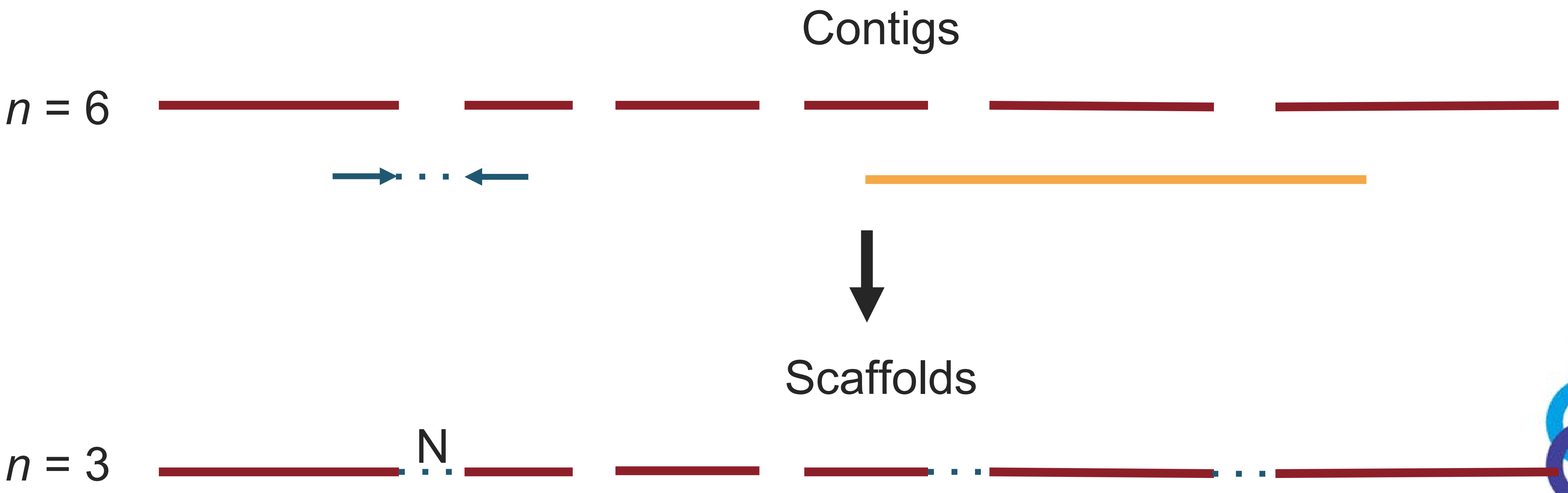
- Overlapping insert
- Long read sequencing (hybrid assembly)



Assembly evaluation

Contigs vs Scaffolds

- Overlapping insert
- Long read sequencing (hybrid assembly)



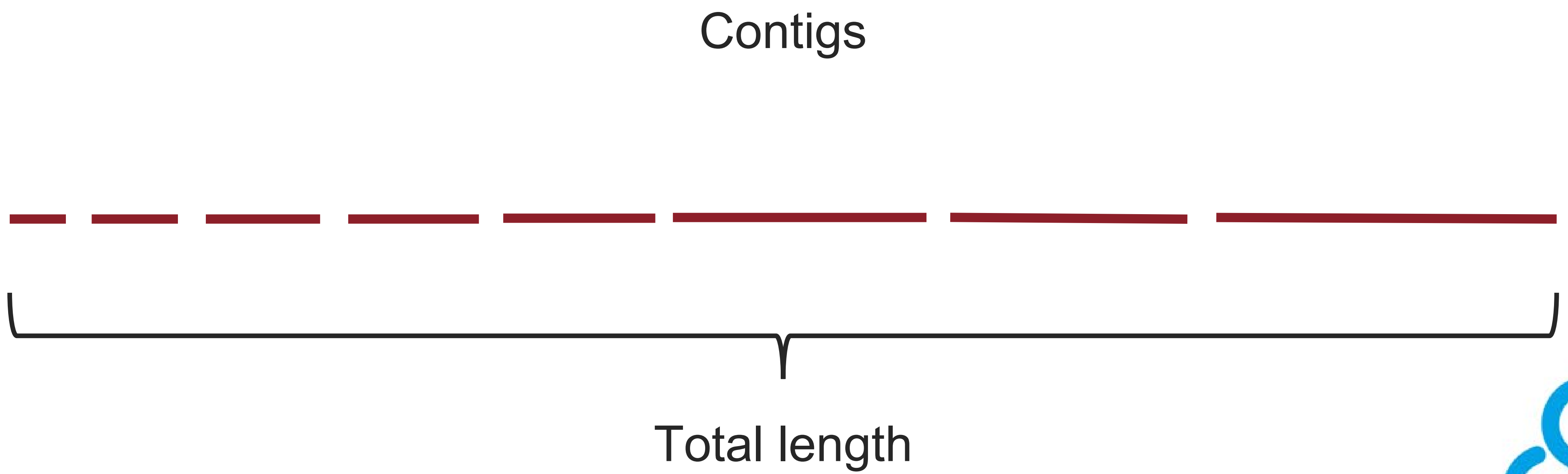
Assembly evaluation

Parameters to use in evaluation:

- Number of contigs (less is more)
- Total length of contigs (= amount assembled)
- Total length of contigs usable (e.g. >1,000 bp, or at least the length of one bacterial gene)
- Length distribution of contigs
- N50 (minimum contig length at 50% of the total genome length)
- Recovery of particular genomes (determined at later stage)

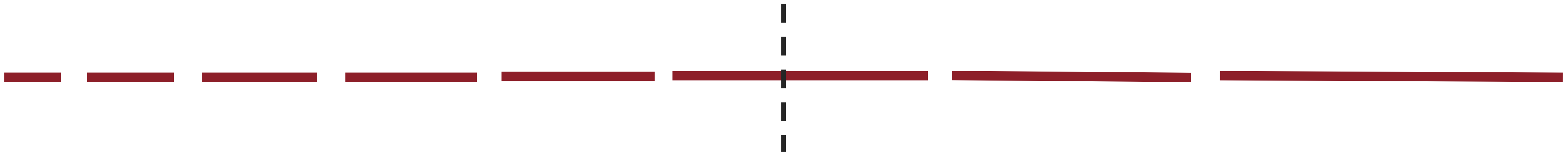
Assembly evaluation

N50 vs L50



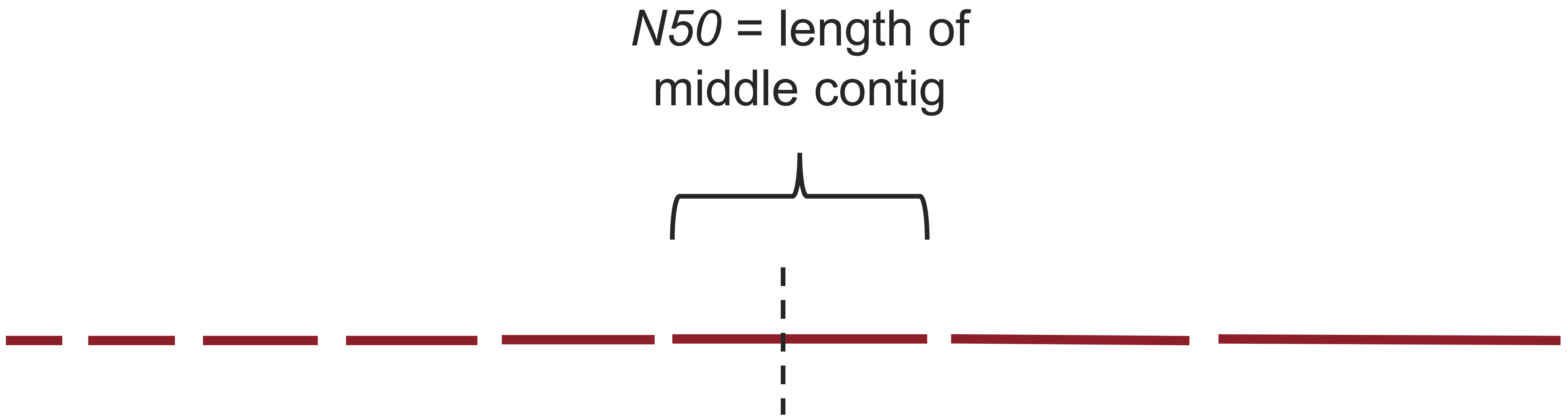
Assembly evaluation

N50 vs L50



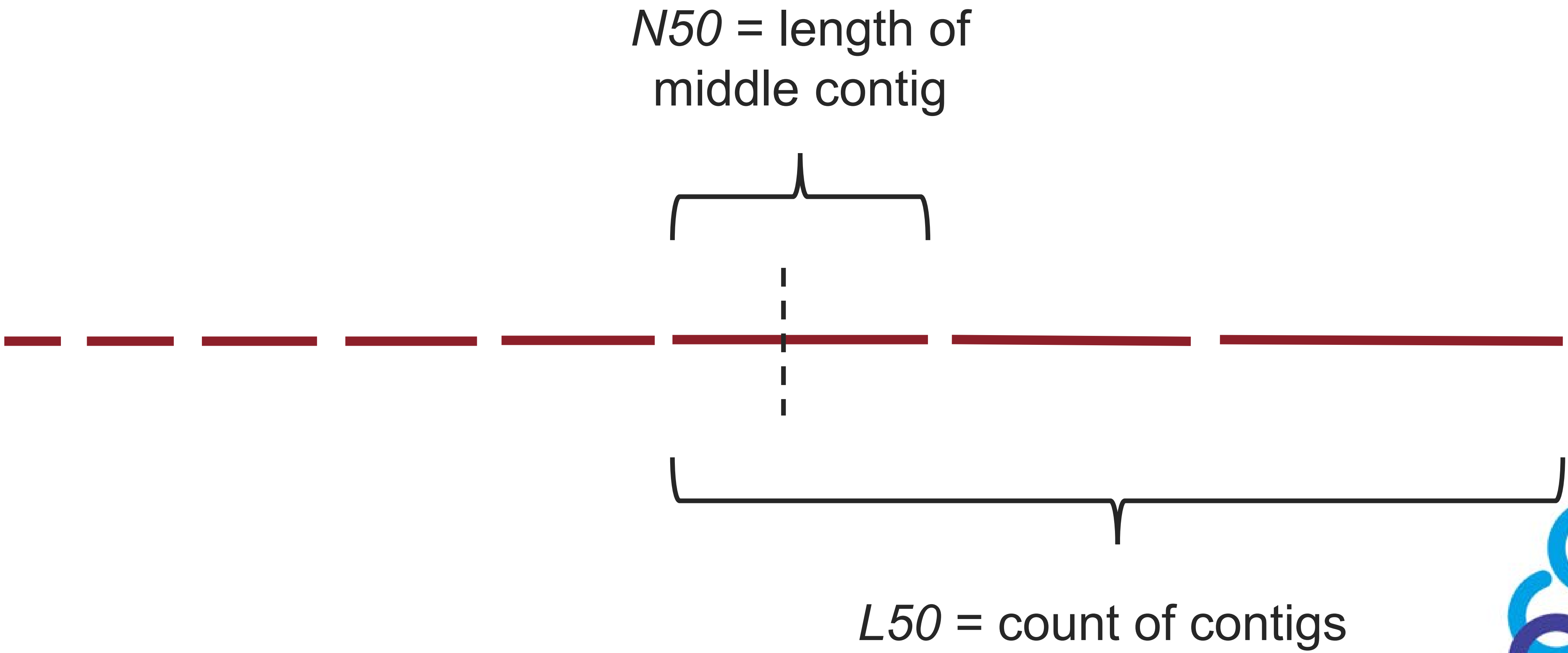
Assembly evaluation

N50 vs L50



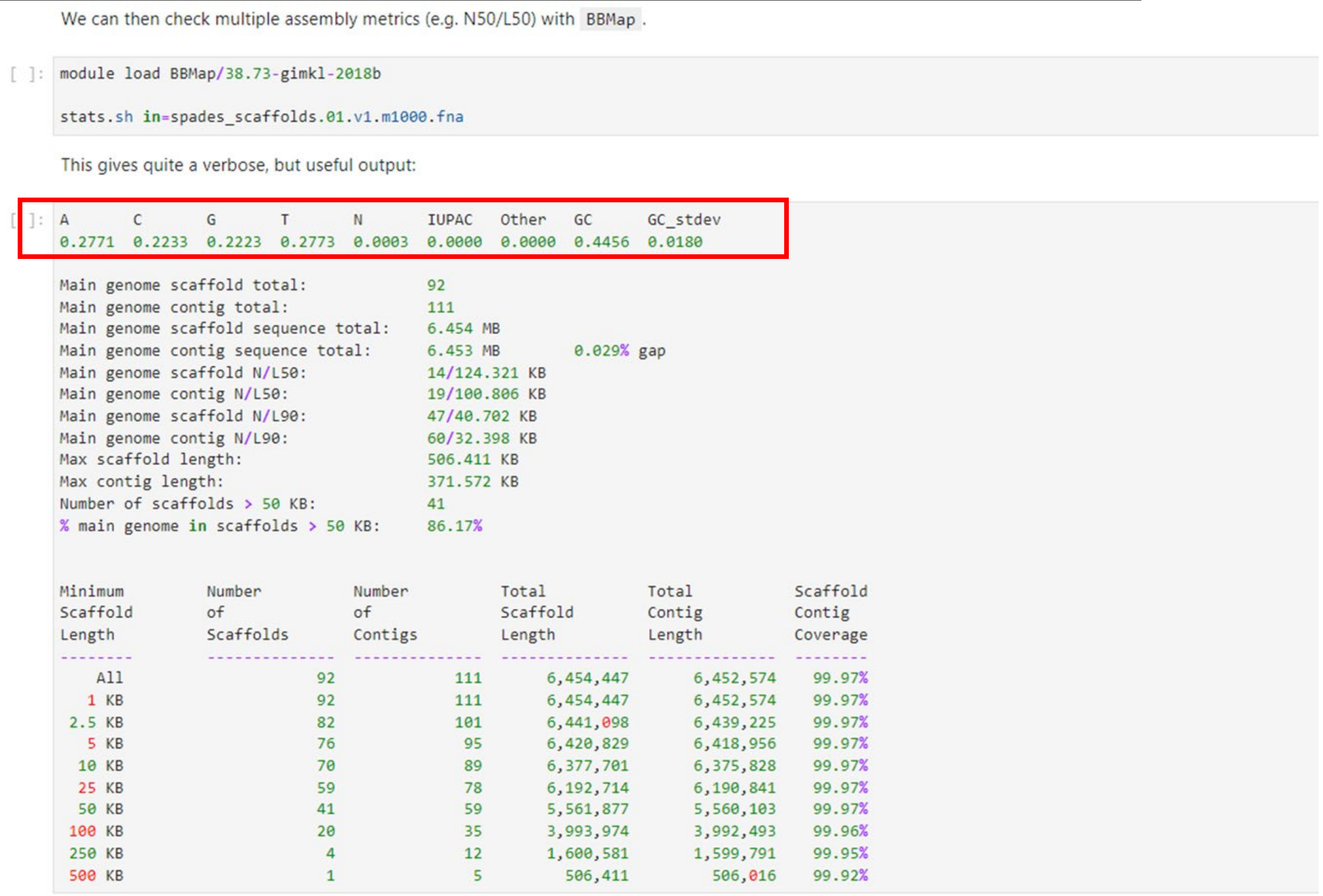
Assembly evaluation

N50 vs L50



Assembly evaluation

Assembly evaluation



We can then check multiple assembly metrics (e.g. N50/L50) with `BBMap`.

```
[ ]: module load BBMap/38.73-gimkl-2018b

     stats.sh in=spades_scaffolds.01.v1.m1000.fna
```

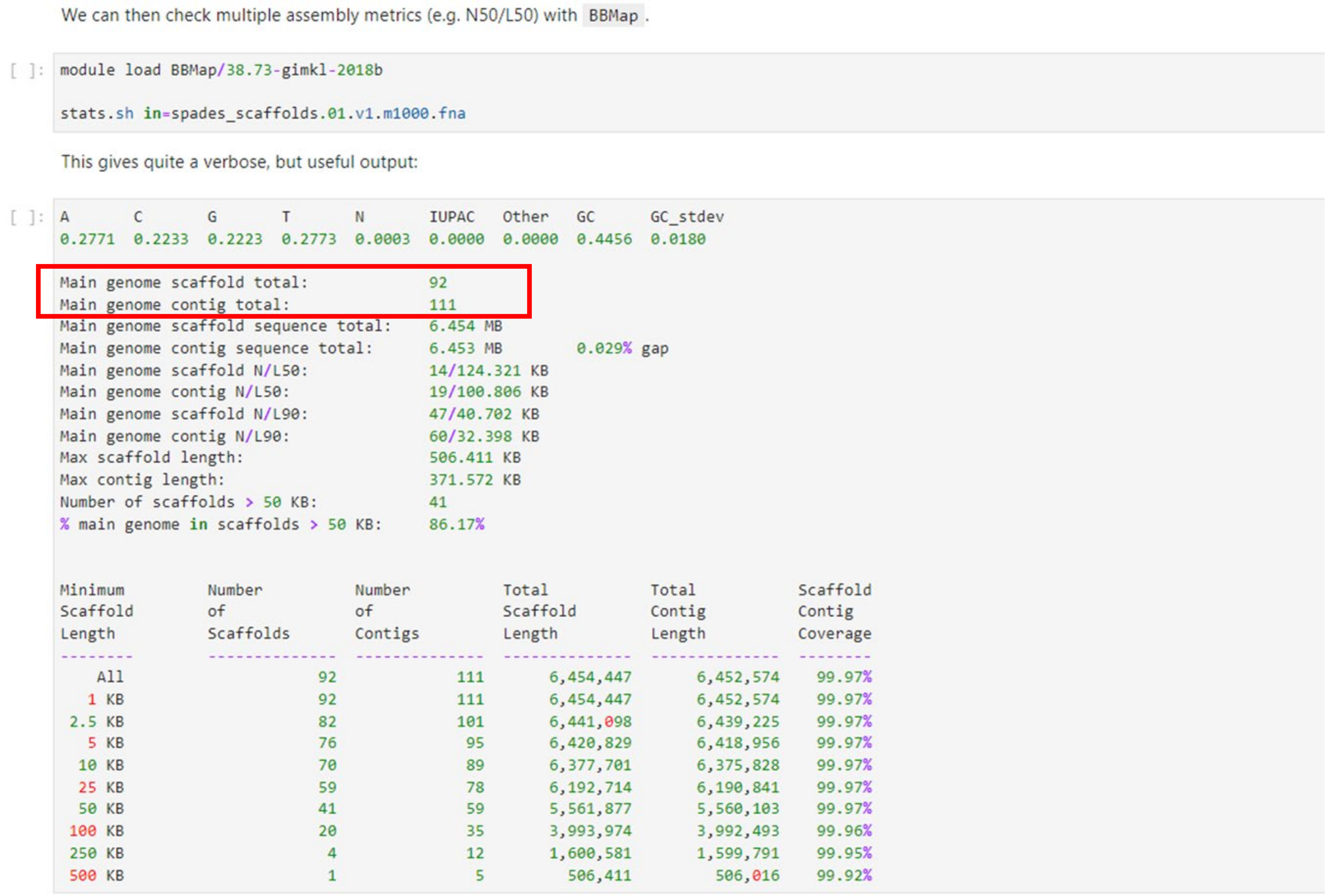
This gives quite a verbose, but useful output:

```
[ ]: A       C       G       T       N       IUPAC   Other   GC      GC_stdev
     0.2771  0.2233  0.2223  0.2773  0.0003  0.0000  0.0000  0.4456  0.0180

     Main genome scaffold total:             92
     Main genome contig total:               111
     Main genome scaffold sequence total:    6.454 MB
     Main genome contig sequence total:      6.453 MB        0.029% gap
     Main genome scaffold N/L50:             14/124.321 KB
     Main genome contig N/L50:               19/100.806 KB
     Main genome scaffold N/L90:             47/40.702 KB
     Main genome contig N/L90:               60/32.398 KB
     Max scaffold length:                    506.411 KB
     Max contig length:                      371.572 KB
     Number of scaffolds > 50 KB:            41
     % main genome in scaffolds > 50 KB:     86.17%


     Minimum         Number          Number          Total           Total           Scaffold
     Scaffold        of              of              Scaffold        Contig          Contig
     Length          Scaffolds       Contigs         Length          Length          Coverage
     --------        --------------  --------------  --------------  --------------  --------
         All                     92             111       6,454,447       6,452,574    99.97%
        1 KB                     92             111       6,454,447       6,452,574    99.97%
      2.5 KB                     82             101       6,441,098       6,439,225    99.97%
        5 KB                     76              95       6,420,829       6,418,956    99.97%
       10 KB                     70              89       6,377,701       6,375,828    99.97%
       25 KB                     59              78       6,192,714       6,190,841    99.97%
       50 KB                     41              59       5,561,877       5,560,103    99.97%
      100 KB                     20              35       3,993,974       3,992,493    99.96%
      250 KB                      4              12       1,600,581       1,599,791    99.95%
      500 KB                      1               5         506,411         506,016    99.92%
```

Assembly evaluation



We can then check multiple assembly metrics (e.g. N50/L50) with `BBMap`.

```
[ ]: module load BBMap/38.73-gimkl-2018b

     stats.sh in=spades_scaffolds.01.v1.m1000.fna
```

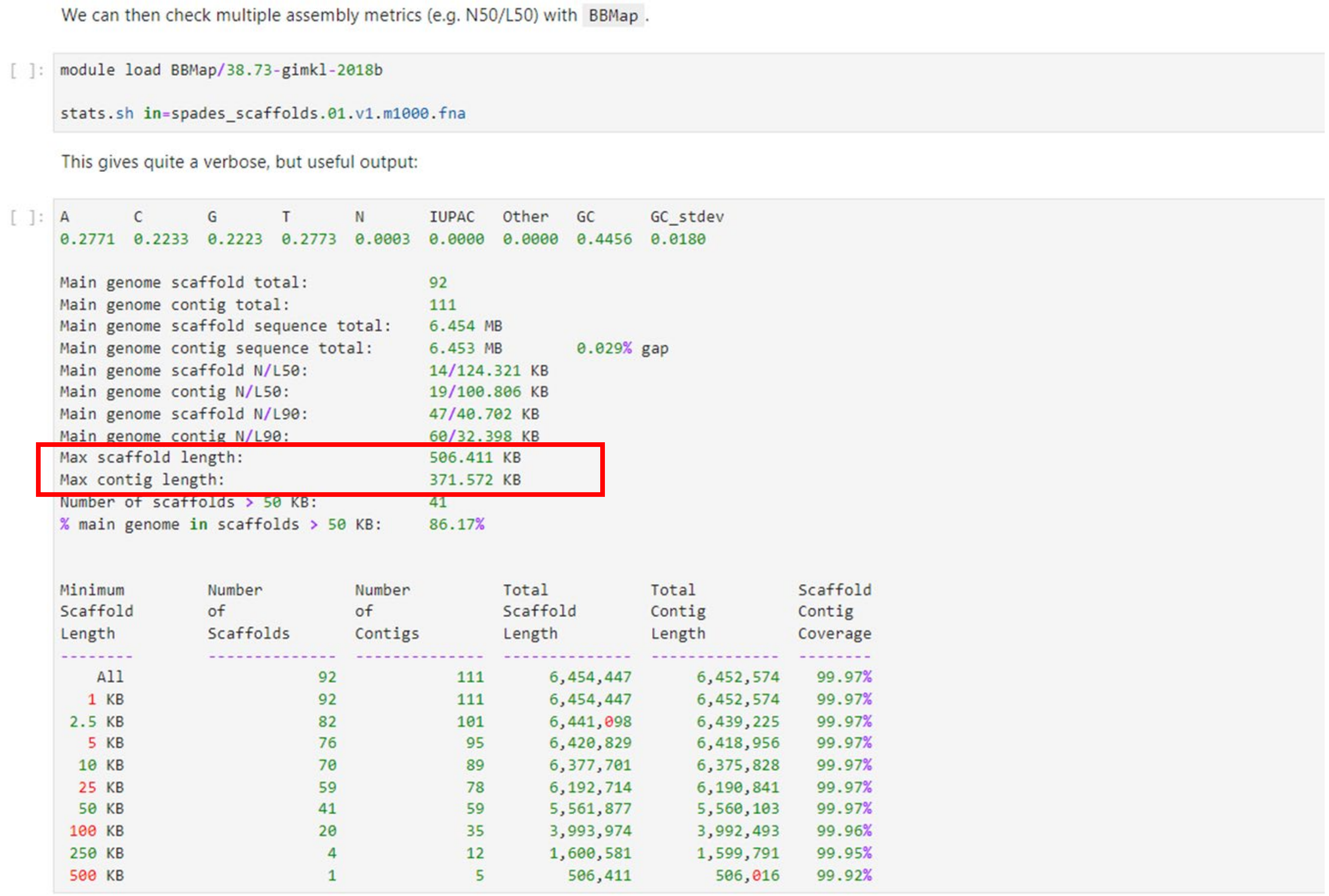
This gives quite a verbose, but useful output:

```
[ ]: A       C       G       T       N       IUPAC   Other   GC      GC_stdev
     0.2771  0.2233  0.2223  0.2773  0.0003  0.0000  0.0000  0.4456  0.0180

     Main genome scaffold total:             92
     Main genome contig total:               111
     Main genome scaffold sequence total:    6.454 MB
     Main genome contig sequence total:      6.453 MB        0.029% gap
     Main genome scaffold N/L50:             14/124.321 KB
     Main genome contig N/L50:               19/100.806 KB
     Main genome scaffold N/L90:             47/40.702 KB
     Main genome contig N/L90:               60/32.398 KB
     Max scaffold length:                    506.411 KB
     Max contig length:                      371.572 KB
     Number of scaffolds > 50 KB:            41
     % main genome in scaffolds > 50 KB:     86.17%


     Minimum         Number          Number          Total           Total           Scaffold
     Scaffold        of              of              Scaffold        Contig          Contig
     Length          Scaffolds       Contigs         Length          Length          Coverage
     --------        --------------  --------------  --------------  --------------  --------
         All                     92             111       6,454,447       6,452,574    99.97%
        1 KB                     92             111       6,454,447       6,452,574    99.97%
      2.5 KB                     82             101       6,441,098       6,439,225    99.97%
        5 KB                     76              95       6,420,829       6,418,956    99.97%
       10 KB                     70              89       6,377,701       6,375,828    99.97%
       25 KB                     59              78       6,192,714       6,190,841    99.97%
       50 KB                     41              59       5,561,877       5,560,103    99.97%
      100 KB                     20              35       3,993,974       3,992,493    99.96%
      250 KB                      4              12       1,600,581       1,599,791    99.95%
      500 KB                      1               5         506,411         506,016    99.92%
```

Assembly evaluation



We can then check multiple assembly metrics (e.g. N50/L50) with `BBMap`.

```
[ ]: module load BBMap/38.73-gimkl-2018b

     stats.sh in=spades_scaffolds.01.v1.m1000.fna
```

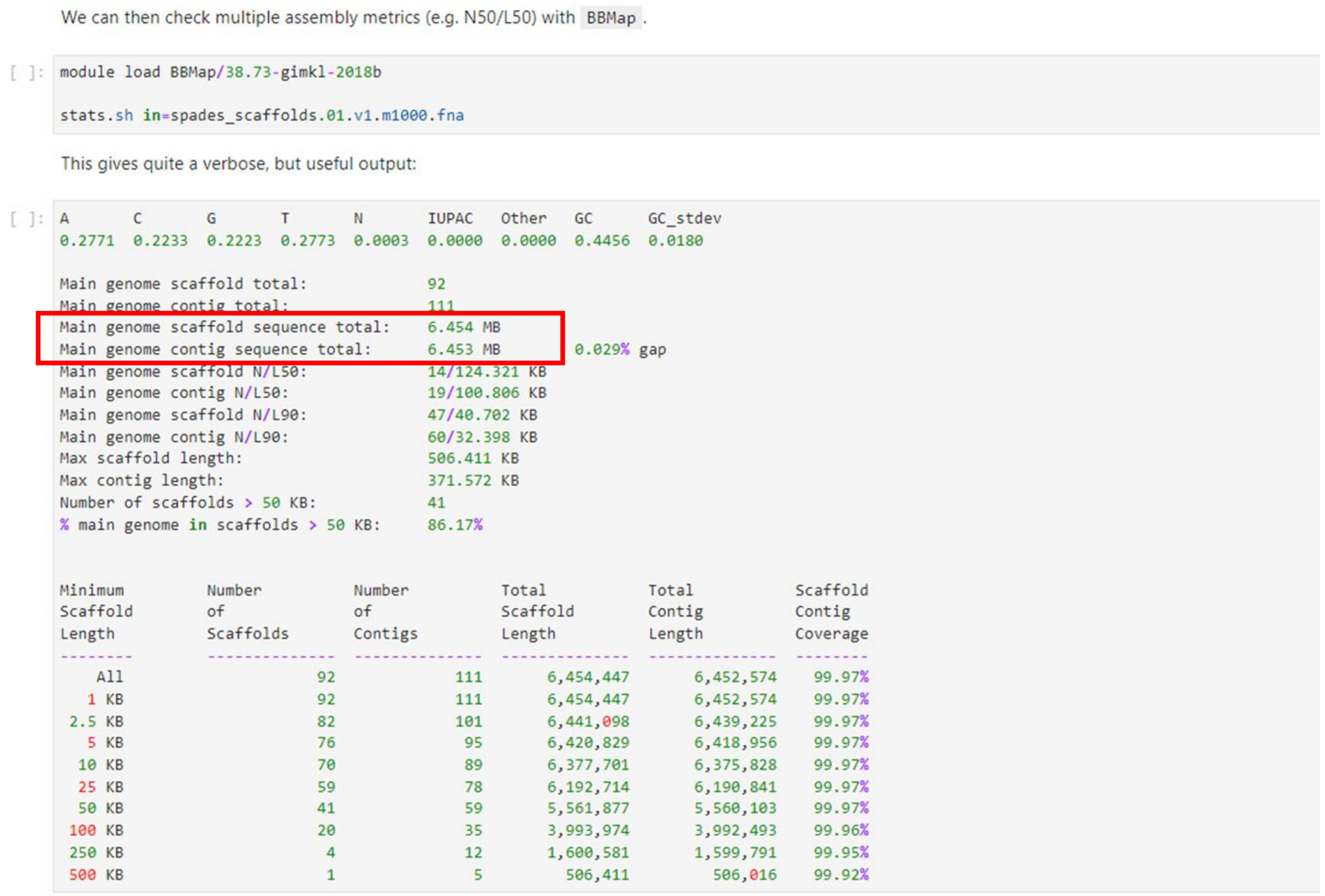
This gives quite a verbose, but useful output:

```
[ ]: A       C       G       T       N       IUPAC   Other   GC      GC_stdev
     0.2771  0.2233  0.2223  0.2773  0.0003  0.0000  0.0000  0.4456  0.0180

     Main genome scaffold total:             92
     Main genome contig total:               111
     Main genome scaffold sequence total:    6.454 MB
     Main genome contig sequence total:      6.453 MB        0.029% gap
     Main genome scaffold N/L50:             14/124.321 KB
     Main genome contig N/L50:               19/100.806 KB
     Main genome scaffold N/L90:             47/40.702 KB
     Main genome contig N/L90:               60/32.398 KB
     Max scaffold length:                    506.411 KB
     Max contig length:                      371.572 KB
     Number of scaffolds > 50 KB:            41
     % main genome in scaffolds > 50 KB:     86.17%


     Minimum         Number          Number          Total           Total           Scaffold
     Scaffold        of              of              Scaffold        Contig          Contig
     Length          Scaffolds       Contigs         Length          Length          Coverage
     --------        --------------  --------------  --------------  --------------  --------
         All                     92             111       6,454,447       6,452,574    99.97%
        1 KB                     92             111       6,454,447       6,452,574    99.97%
      2.5 KB                     82             101       6,441,098       6,439,225    99.97%
        5 KB                     76              95       6,420,829       6,418,956    99.97%
       10 KB                     70              89       6,377,701       6,375,828    99.97%
       25 KB                     59              78       6,192,714       6,190,841    99.97%
       50 KB                     41              59       5,561,877       5,560,103    99.97%
      100 KB                     20              35       3,993,974       3,992,493    99.96%
      250 KB                      4              12       1,600,581       1,599,791    99.95%
      500 KB                      1               5         506,411         506,016    99.92%
```

Assembly evaluation



We can then check multiple assembly metrics (e.g. N50/L50) with `BBMap`.

```
[ ]: module load BBMap/38.73-gimkl-2018b

     stats.sh in=spades_scaffolds.01.v1.m1000.fna
```

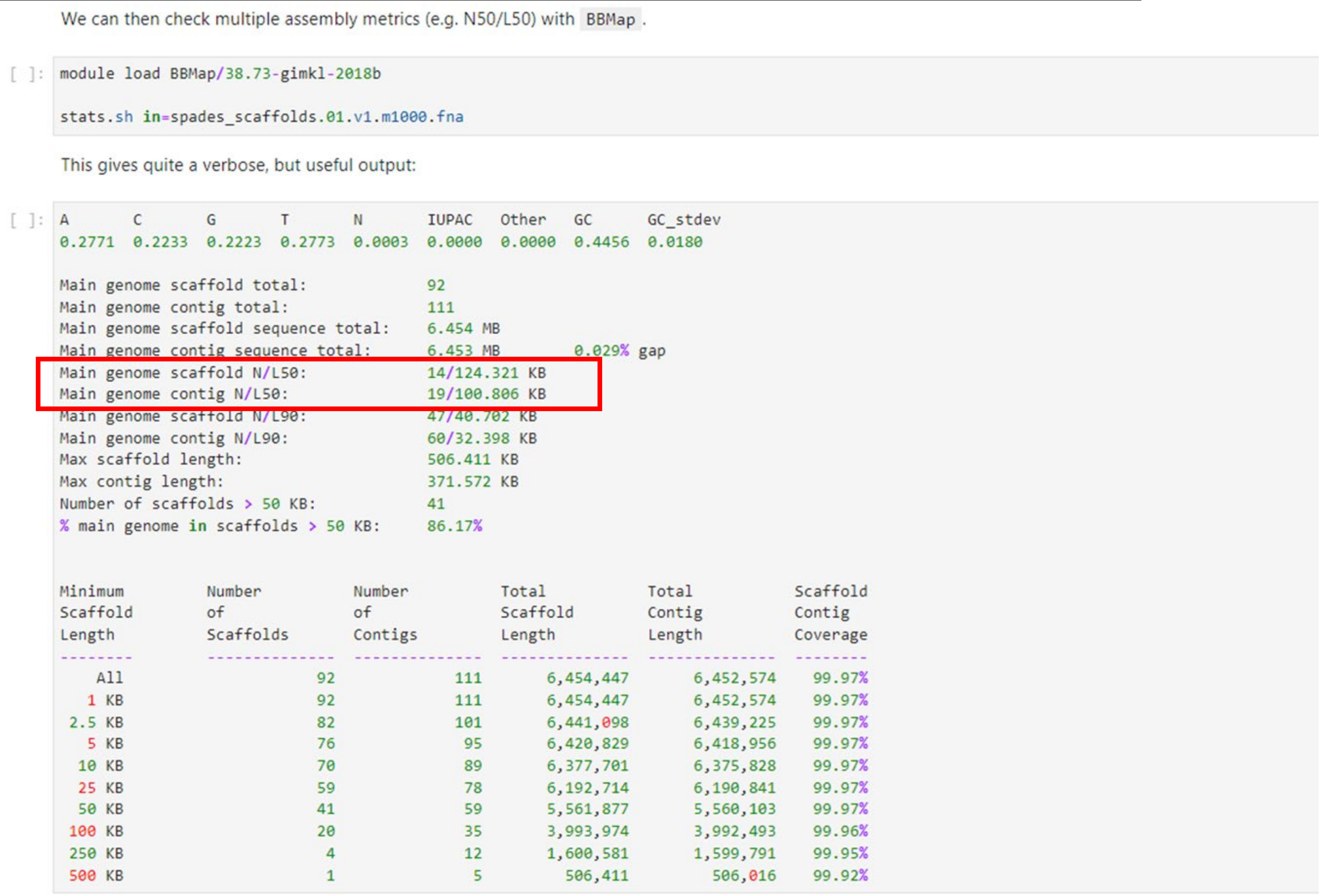
This gives quite a verbose, but useful output:

```
[ ]: A       C       G       T       N       IUPAC   Other   GC      GC_stdev
     0.2771  0.2233  0.2223  0.2773  0.0003  0.0000  0.0000  0.4456  0.0180

     Main genome scaffold total:             92
     Main genome contig total:               111
     Main genome scaffold sequence total:    6.454 MB
     Main genome contig sequence total:      6.453 MB       0.029% gap
     Main genome scaffold N/L50:             14/124.321 KB
     Main genome contig N/L50:               19/100.806 KB
     Main genome scaffold N/L90:             47/40.702 KB
     Main genome contig N/L90:               60/32.398 KB
     Max scaffold length:                    506.411 KB
     Max contig length:                      371.572 KB
     Number of scaffolds > 50 KB:            41
     % main genome in scaffolds > 50 KB:     86.17%


     Minimum         Number          Number          Total           Total           Scaffold
     Scaffold        of              of              Scaffold        Contig          Contig
     Length          Scaffolds       Contigs         Length          Length          Coverage
     --------        --------------  --------------  --------------  --------------  --------
         All                     92             111       6,454,447       6,452,574    99.97%
        1 KB                     92             111       6,454,447       6,452,574    99.97%
      2.5 KB                     82             101       6,441,098       6,439,225    99.97%
        5 KB                     76              95       6,420,829       6,418,956    99.97%
       10 KB                     70              89       6,377,701       6,375,828    99.97%
       25 KB                     59              78       6,192,714       6,190,841    99.97%
       50 KB                     41              59       5,561,877       5,560,103    99.97%
      100 KB                     20              35       3,993,974       3,992,493    99.96%
      250 KB                      4              12       1,600,581       1,599,791    99.95%
      500 KB                      1               5         506,411         506,016    99.92%
```

Assembly evaluation



We can then check multiple assembly metrics (e.g. N50/L50) with `BBMap`.

[ ]:
```
module load BBMap/38.73-gimkl-2018b

stats.sh in=spades_scaffolds.01.v1.m1000.fna
```

This gives quite a verbose, but useful output:

[ ]:
```
A       C       G       T       N       IUPAC   Other   GC      GC_stdev
0.2771  0.2233  0.2223  0.2773  0.0003  0.0000  0.0000  0.4456  0.0180

Main genome scaffold total:             92
Main genome contig total:               111
Main genome scaffold sequence total:    6.454 MB
Main genome contig sequence total:      6.453 MB        0.029% gap
Main genome scaffold N/L50:             14/124.321 KB
Main genome contig N/L50:               19/100.806 KB
Main genome scaffold N/L90:             47/40.702 KB
Main genome contig N/L90:               60/32.398 KB
Max scaffold length:                    506.411 KB
Max contig length:                      371.572 KB
Number of scaffolds > 50 KB:            41
% main genome in scaffolds > 50 KB:     86.17%


Minimum         Number          Number          Total           Total           Scaffold
Scaffold        of              of              Scaffold        Contig          Contig
Length          Scaffolds       Contigs         Length          Length          Coverage
--------        --------------  --------------  --------------  --------------  --------
    All                     92             111       6,454,447       6,452,574    99.97%
   1 KB                     92             111       6,454,447       6,452,574    99.97%
 2.5 KB                     82             101       6,441,098       6,439,225    99.97%
   5 KB                     76              95       6,420,829       6,418,956    99.97%
  10 KB                     70              89       6,377,701       6,375,828    99.97%
  25 KB                     59              78       6,192,714       6,190,841    99.97%
  50 KB                     41              59       5,561,877       5,560,103    99.97%
 100 KB                     20              35       3,993,974       3,992,493    99.96%
 250 KB                      4              12       1,600,581       1,599,791    99.95%
 500 KB                      1               5         506,411         506,016    99.92%
```

Task: Assembly evaluation

Go to Github MGSS webpage

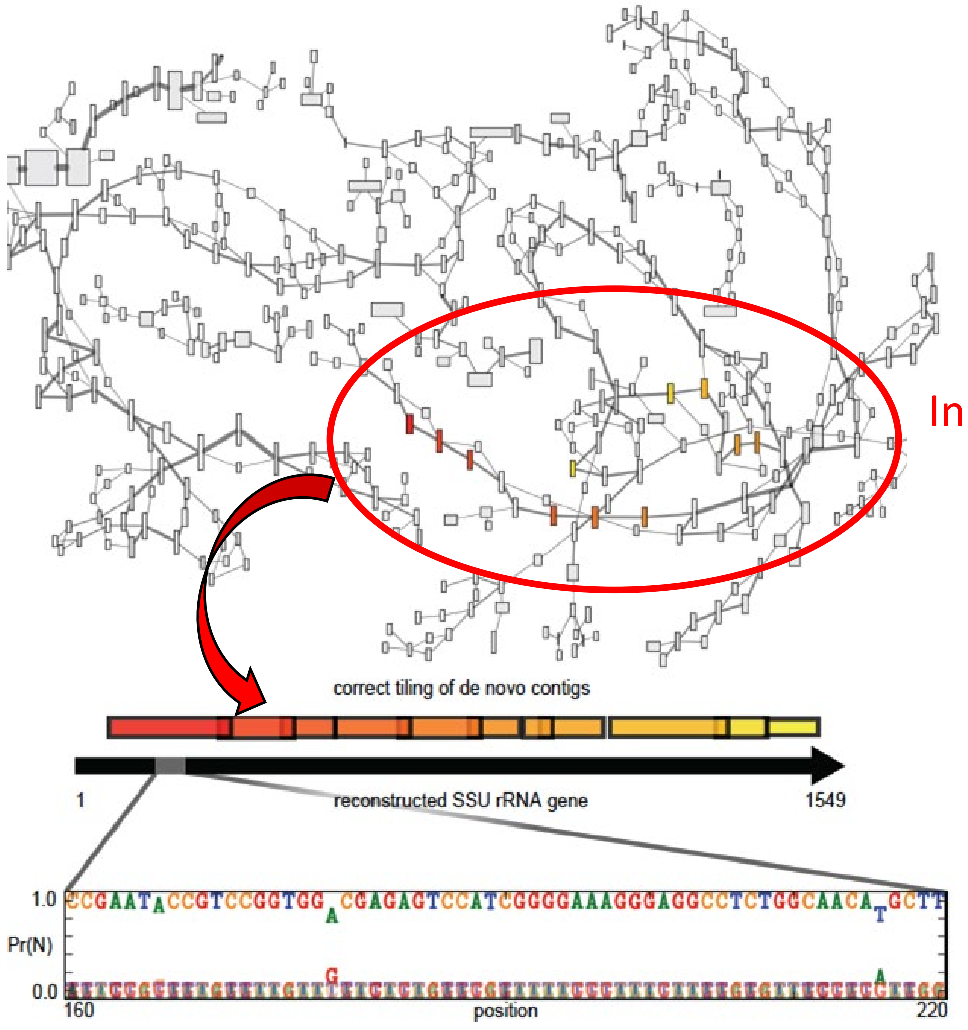
Tasks:

- Assembly evaluation
- Short contig removal

Other considerations: rRNA reconstruction

Other considerations: rRNA genes

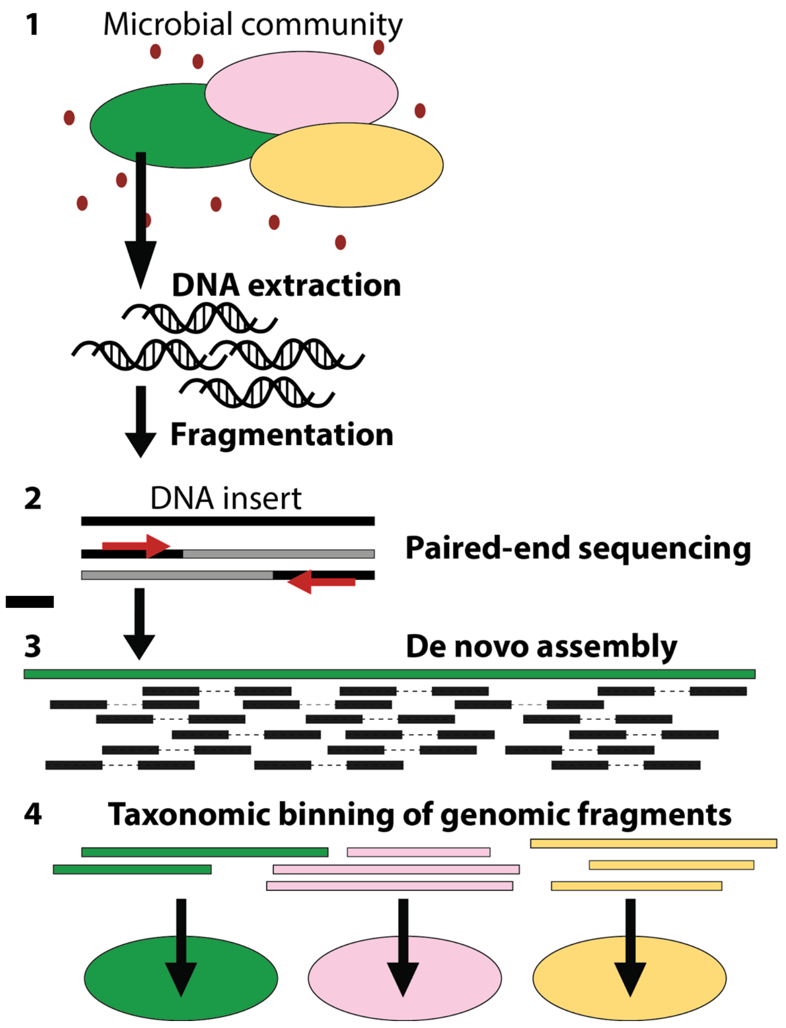
SSU rRNA reference guided and iterative assembly



Incorrect in *de novo* assembly
correct tiling of de novo contigs
1
reconstructed SSU rRNA gene
1549
1.0
Pr(N)
0.0
160
position
220

(Miller et al., 2011, Genome Biology)

Other considerations: rRNA genes



Reference-guided SSU rRNA reconstruction from WGS data (EMIRGE or phyloFlash)
https://github.com/csmiller/EMIRGE
https://hrgv.github.io/phyloFlash/

EMIRGE

Reconstructs full-length small subunit (SSU) gene sequences

SSU genes are a common phylogenetic marker used to differentiate taxa and assign taxonomy to sequence data (e.g. 16S rRNA)

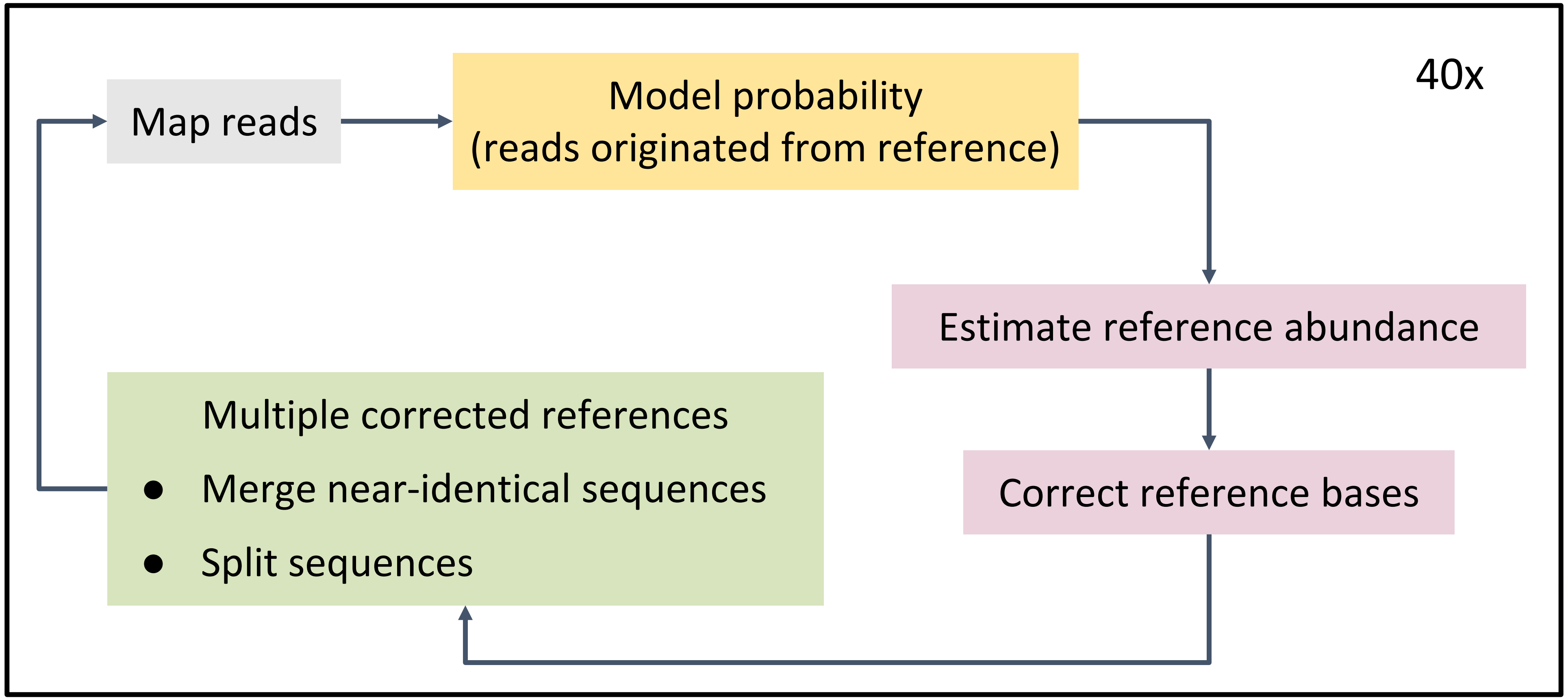
Performs iterations of an expectation maximization algorithm to calculate the probability scores for the reconstructed genes to obtain a consensus sequence

Also generates abundance data based on consensus sequences

Database guided reconstruction

EMIRGE

Sequence convergence after iterative nucleotide correction using expectation maximization (EM) algorithm


40x
Map reads
Model probability (reads originated from reference)
Estimate reference abundance
Correct reference bases
Multiple corrected references
● Merge near-identical sequences
● Split sequences

Database guided reconstruction

EMIRGE

Sequence convergence after iterative nucleotide correction using expectation maximization (EM) algorithm

- Database as initial guide (usually SILVA)
  - Need to remove chimeric sequences!
- Expectation maximisation (EM) algorithm handles uncertainty from
  - Sequencing errors
  - Mapping ambiguity between closely related strains
  - Uncertain reference assignments
  - Non-representation of reads in database
- Can reconstruct novel rRNA sequences

PhyloFlash

Wrapper pipeline for the reconstruction of full-length small subunit (SSU) gene sequences

Uses EMIRGE and/or metaSPAdes for SSU gene reconstruction

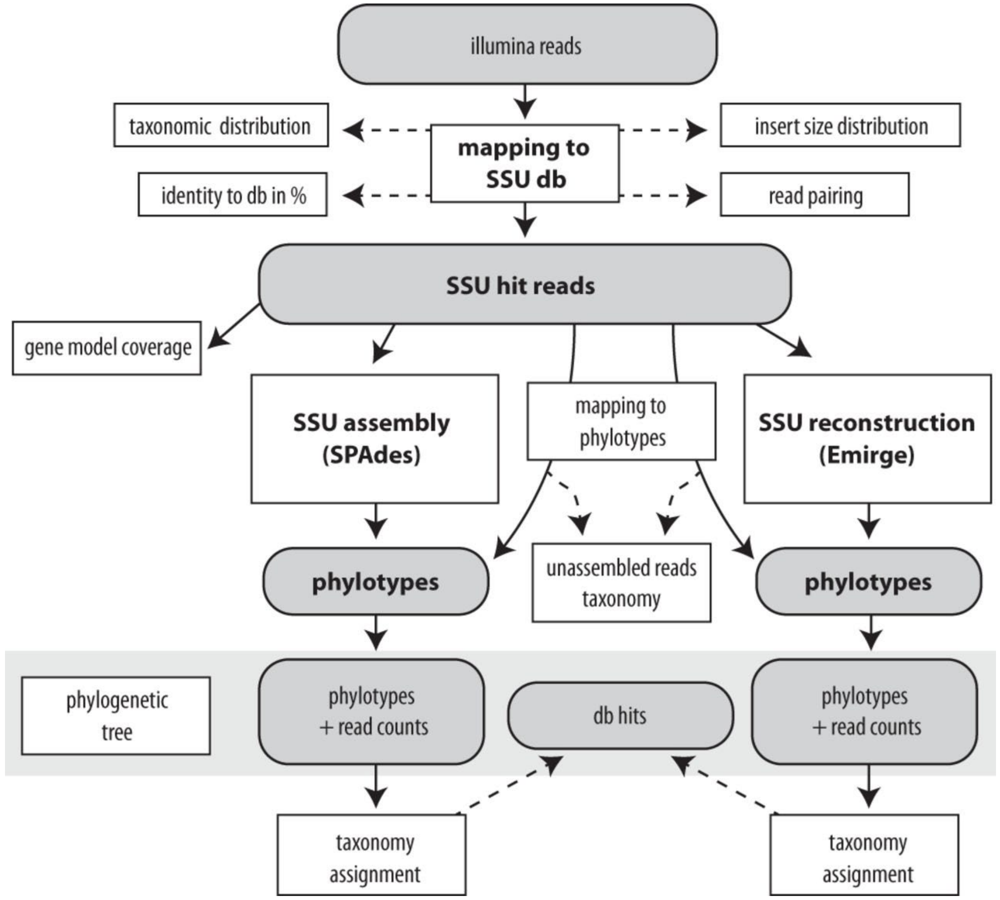
Output: NTU (number of taxonomic units) table and an HTML file full of summary data and figures

“We designed phyloFlash for the rapid screening of SSU rRNA sequences in metagenomic libraries”

Database guided reconstruction

PhyloFlash

Wrapper pipeline for the reconstruction of full-length small subunit (SSU) gene sequences



- Database mapping for potential rRNA gene reads
- SPAdes to assemble selected reads OR
- EMIRGE to reconstruct full sequence
- Outputs NTU (number of taxonomic units) with ready to use HTML summaries and NTU tables

Gruber-Vodicka, Seah & Pruesse (2020) *mSystems*

Task: PhyloFlash

Go to Github MGSS webpage

Tasks:

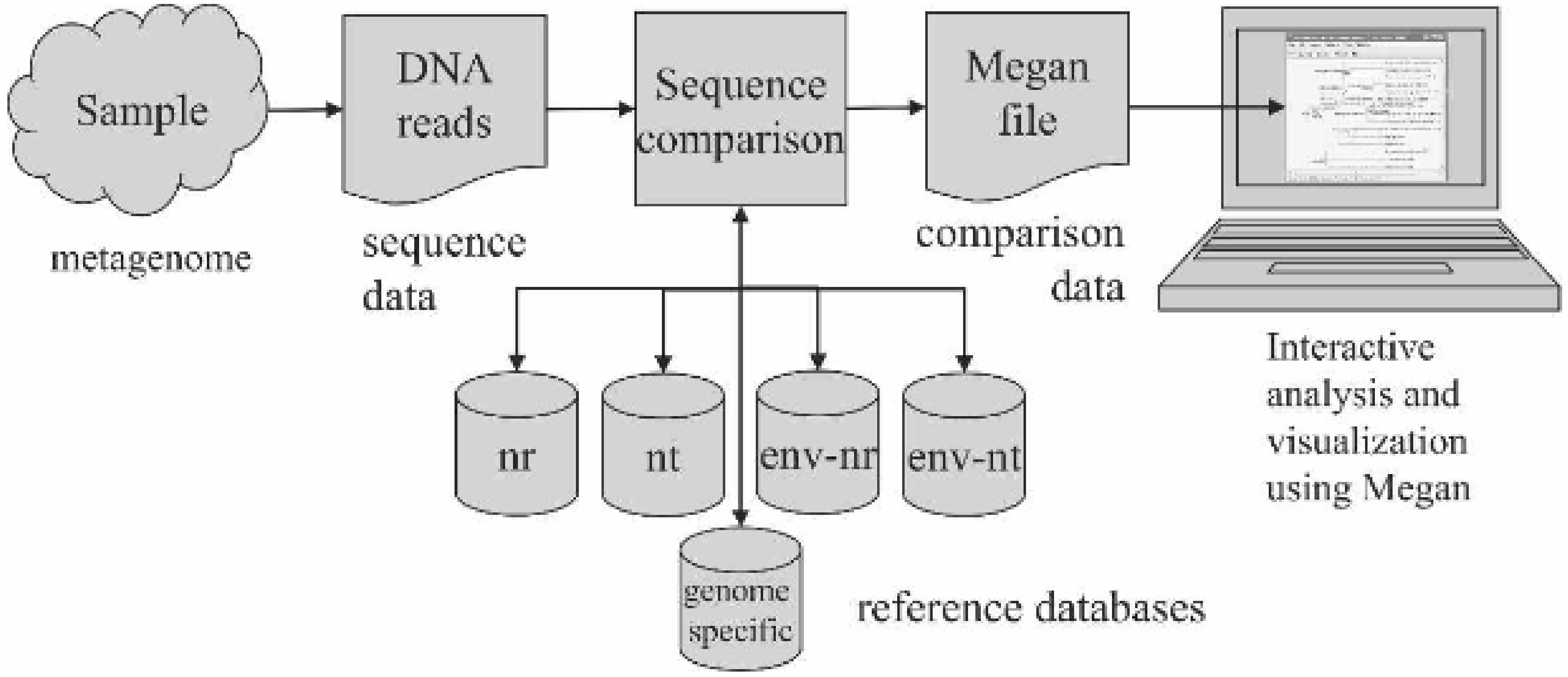
- **Explore taxonomy with PhyloFlash**

“Read-based” taxonomy assignment

Sequence alignment (Protein)

MEGAN

✅ BLASTx comparison to find best scoring LCA

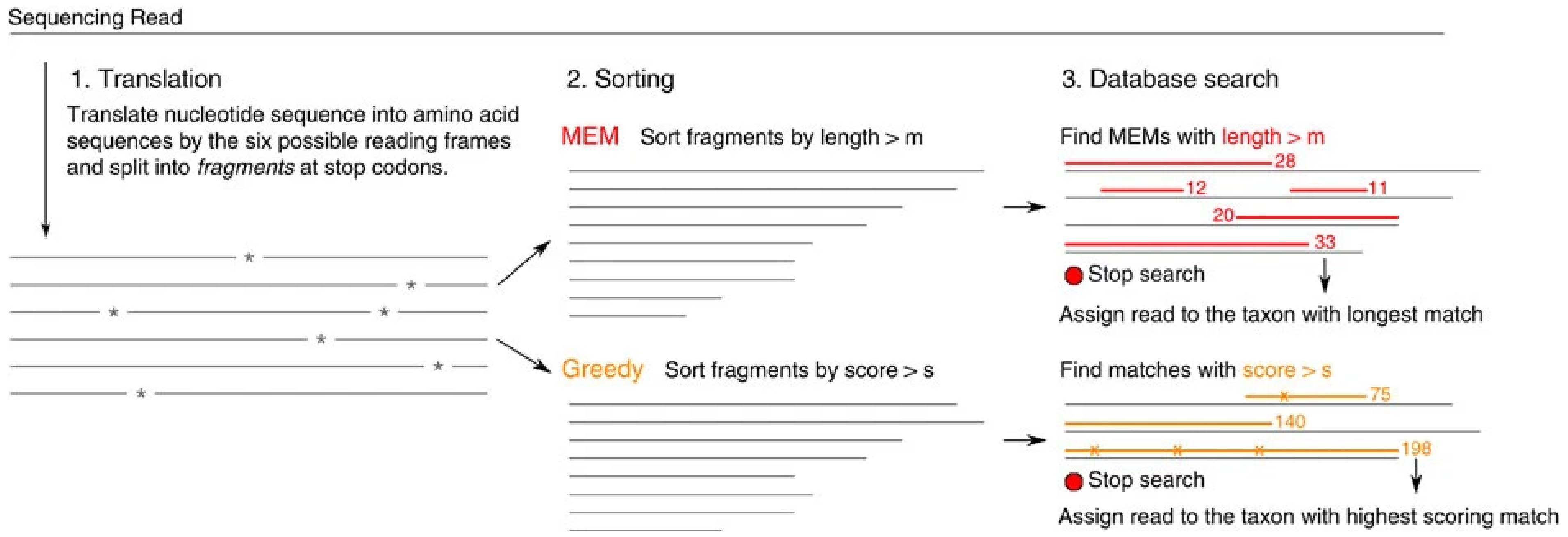


Huson et al. (2006) *Genome Res.*

Sequence alignment (Protein)

Kaiju

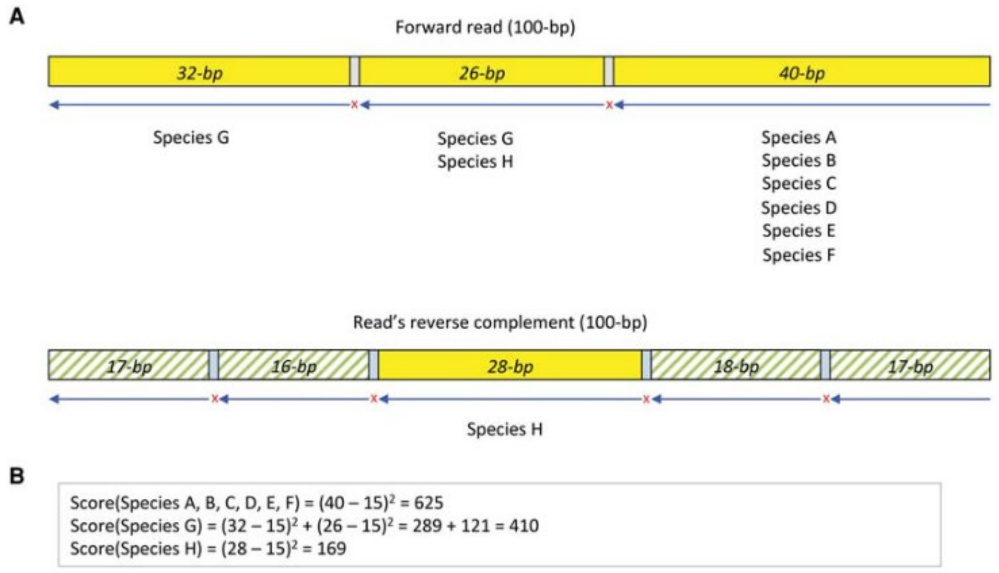
✅ Best scoring single match or LCA if multiple best matches



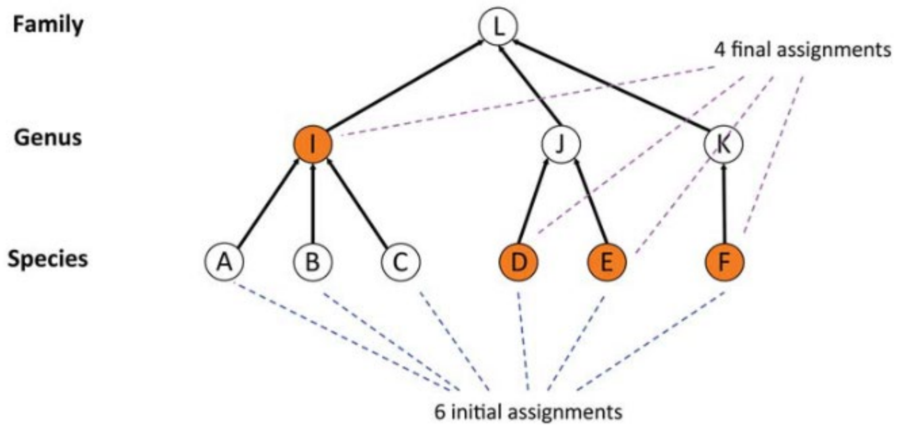
Sequencing Read
1. Translation
Translate nucleotide sequence into amino acid sequences by the six possible reading frames and split into *fragments* at stop codons.
2. Sorting
MEM Sort fragments by length > m
Greedy Sort fragments by score > s
3. Database search
Find MEMs with length > m
28
12
11
20
33
Stop search
Assign read to the taxon with longest match
Find matches with score > s
75
140
198
Stop search
Assign read to the taxon with highest scoring match

Menzel, Ng & Krogh (2016) *Nat. Communs.*

Sequence alignment (DNA)



Score(Species A, B, C, D, E, F) = $(40 - 15)^2 = 625$
Score(Species G) = $(32 - 15)^2 + (26 - 15)^2 = 289 + 121 = 410$
Score(Species H) = $(28 - 15)^2 = 169$



Centrifuge

✅ Best scoring matches after traversing tree

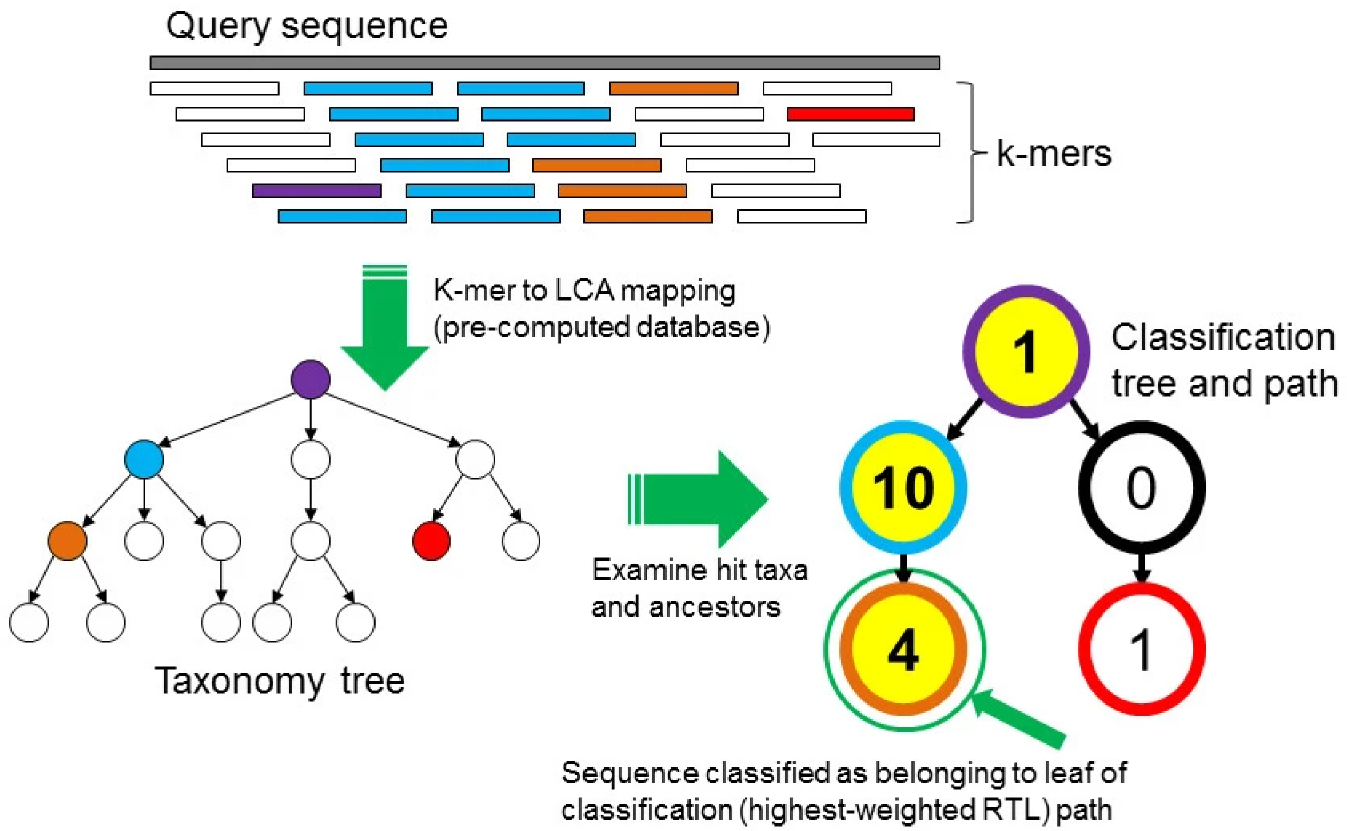
* Multiple matches possible

Kim et al. (2016) *Genome Res.*

k-mer based classification

Kraken2

✅ Best scoring leaf/LCA of pruned tree



Wood & Salzberg (2014) *Genome Biol.*

- Exact *k*-mer matching (default *k* = 35)
- Memory- and time-efficient database search
- Pre-built generic databases available
- Estimate abundance with Bracken
- **Trade-off**: *resource efficiency vs match accuracy*
    - Larger database ⇒ Better accuracy ⇒ Slower runtime

Task: Kraken

Go to Github MGSS webpage

Tasks:

- Explore taxonomy with Kraken
- Estimate taxonomic read abundance with Bracken

Mini-project

- Denitrification (Nitrate or nitrite to nitrogen)
- Ammonia oxidation (Ammonia to nitrite or nitrate)
- Anammox (Ammonia and nitrite to nitrogen)
- Sulfur oxidation (SOX pathway, thiosulfate to sulfate)
- Sulfur reduction (DSR pathway, sulfate to sulfide)
- Photosynthetic carbon fixation
- Non-photosynthetic carbon fixation (Reverse TCA or Wood-Ljungdahl)
- Non-polar flagella expression due to a chromosomal deletion
- Plasmid-encoded antibiotic resistance
- Aerobic (versus anaerobic) metabolism