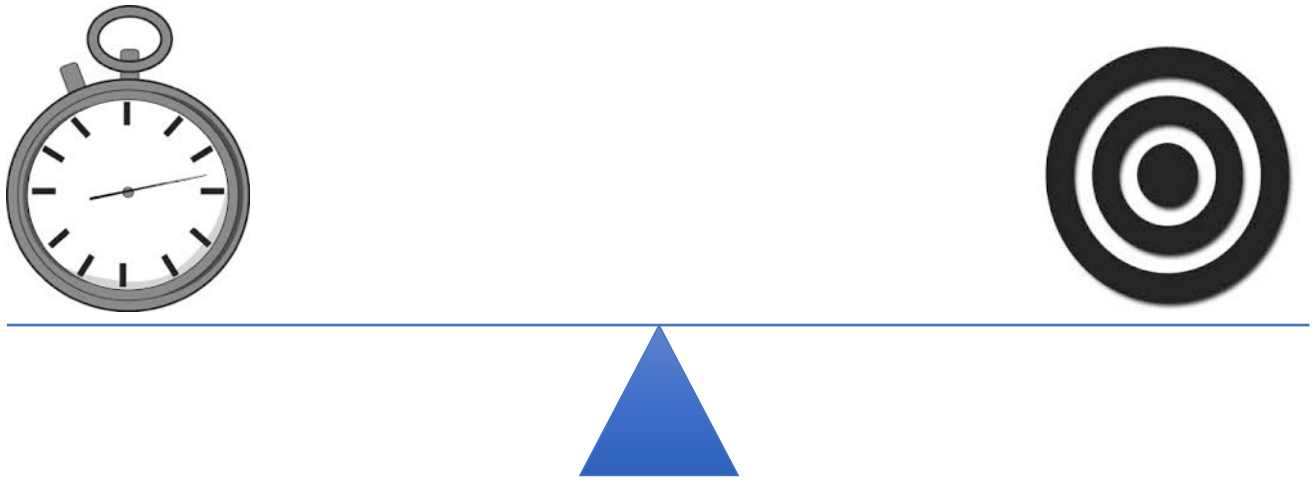
Short Read Mapping
and
GFF file format

BLAST vs short read alignment

- Alignment methods must have tradeoffs
- Depending on the application, may want to make different tradeoffs
- Different types of alignment objectives lead to different categories of aligners



BLAST

- Slow (in comparison to short read mappers)
- By default expects longer query sequences (>100bp)
- Can detect evolutionary relationships with significant loss of percent identity (i.e. sequences are only 50% similar)

Short Read Mappers

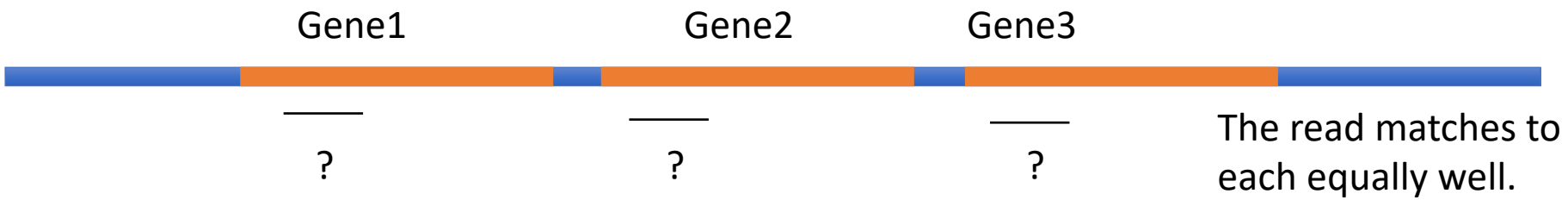
- Fast
- Can handle very short sequences (~25bp)
- Will only find matches of 90% identity or more

Short Read Mappers

- Orders of magnitude faster than BLAST
- several tens of millions of reads mapped per hour per CPU
- Only matches of 90% identity or greater are found
- Usually only output the best hit or the set of hits all equivalently good
  - The point is usually to find the origin in the reference genome
  - Other genomic regions of lower identity are not considered useful

Uniqueness

- Some reads can be mapped uniquely to the reference
- Some map to multiple locations
- Multiply mapped reads are difficult to apply to downstream applications
  - RNASeq – which gene do they represent?
  - SNP – which location carries the substitution?
- How to deal with multiply mapped reads?
  - Throw them away?



Clever Ways to find “Best” Alignments

- Use only the parts of genes that are unique
  - Discard multimapped reads
  - Can make the assumption that the multi-mapped reads would have mapped in the same ratio as unmapped reads
- Use the quality values
  - Penalize mismatches at high quality bases more than mismatches at low quality bases
- Paired End information
  - If one read does not map uniquely, but the other does, use that information to place the non-unique one

Decisions for the end user

- Slower and more sensitive or faster and less sensitive?
- How many mismatches are allowed for a read to be considered mapped?
  - Heterozygosity between sample and reference
  - Incomplete/low quality reference
- How many matches to report?
  - Does your downstream analysis need/want to include multiple matches?

Explore the documentation and parameters for your software of choice

Is it doing what you think its doing?

Software Options

- Most common, highly accurate:
  - HISAT2 (Use this instead of TopHat)
  - STAR
- Many others… how to choose?
  - Common in the literature
  - Good documentation
  - Memory efficient
  - Responsive mailing list or help forum
  - Maintained and updated when bugs are found

What mappers have in common: Indexing Strategies

- Usually, the first step is to transform part of the data into a more suitable form for fast searching
- Indexing – creating a glossary or look up table
- Without indexing you would have to scan everything each time you did a search
- Consider web search engines

Indexing

- You have a reference genome (but imagine its a billion bases)

  CTCCTAGAATGCTGGGAAGTGGAAGTCCAACTTCTTCCATGGGTTCACCT

- You have a read (but imagine you have 100 million)

  CTCCTATAATGCTGGGAA

Indexing

Start by creating ”words” from the reference:

```
CTCCTAGAATGCTGGGAAGTGGAAGTCCAACTTCTTCCATGGGTTCACCT
CTCCTA
 TCCTAG
  CCTAGA
   CTAGAA
    TAGAAT
     AGAATG
      GAATGC
       AATGCT
        ATGCTG
Etc.
```

Indexing

Put the words in order

```
AATGCT
AGAATG
ATGCTG
CCTAGA
CTAGAA
CTCCTA
GAATGC
TAGAAT
TCCTAG
```

Indexing and Read Mapping

- Now create “words” from the read…. And put in order

```
CTCCTATAATGCTGGGAA
CTCCTA
 TCCTAT
  CCTATA
   CTATAA
    TATAAT
     ATAATG
      TAATGC
       AATGCT
        ATGCTG
```

Indexing and Read Mapping

- Compare the two lists. Do we have overlapping words?

Reference word list:

```
AATGCT
AGAATG
ATGCTG
CCTAGA
CTAGAA
CTCCTA
GAATGC
TAGAAT
TCCTAG
```

```
Read word list:
AATGCT
ATAATG
ATGCTG
CCTATA
CTATAA
CTCCTA
TAATGC
TATAAT
TCCTAT
```

Indexing and Read Mapping

- Compare the two lists. Do we have overlapping words?

Reference word list:

AATGCT
AGAATG
ATGCTG
CCTAGA
CTAGAA
CTCCTA
GAATGC
TAGAAT
TCCTAG

Read word list:

AATGCT
ATAATG
ATGCTG
CCTATA
CTATAA
CTCCTA
TAATGC
TATAAT
TCCTAT

Indexing and Read Mapping

- Compare the two lists. Do we have overlapping words?

Reference word list:

AATGCT
AGAATG
ATGCTG
CCTAGA
CTAGAA
CTCCTA
GAATGC
TAGAAT
TCCTAG

Read word list:

AATGCT
ATAATG
ATGCTG
CCTATA
CTATAA
CTCCTA
TAATGC
TATAAT
TCCTAT

In this example the genome and the read align.

Why don't all the words match?

Indexing and Read Mapping

- The perfectly matching words are referred to as seeds

Reference word list:

AATGCT
AGAATG
ATGCTG
CCTAGA
CTAGAA
CTCCTA
GAATGC
TAGAAT
TCCTAG

Read word list:

AATGCT
ATAATG
ATGCTG
CCTATA
CTATAA
CTCCTA
TAATGC
TATAAT
TCCTAT

Next step:

Use those seeds to start the alignment, then extend

Reference:

CTCCTAGAATGCTGGGAAGT

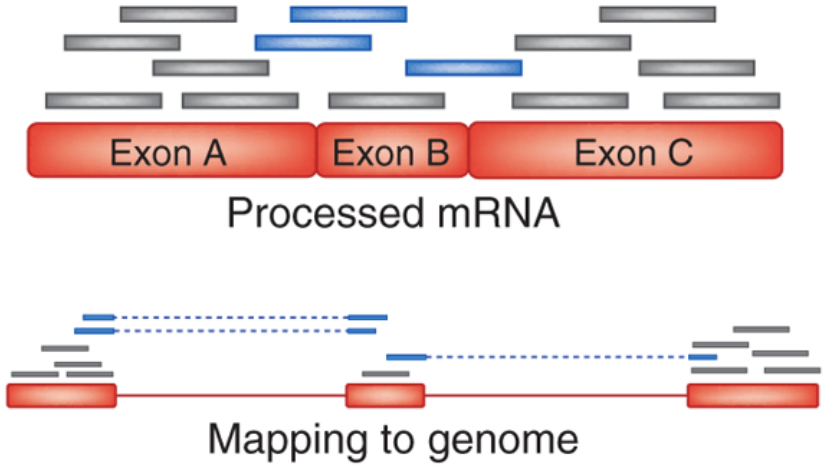
CTCCTATAATGCTGGGAA

Mapping to Genes

- Mapping RNA to a eukaryotic genome is more complicated than mapping DNA
  - Introns
  - Alternative splicing

- Usually, you want to use a mapping software designed for RNASeq

  - The software will use a file (gff3) to know where the genes are located and automatically splice out introns



GFF- Generic Feature Format

- GFF was the original file format
- Represent genomic features on a sequence
  - gene on a chromosome
- But it did not cover all the use cases needed. Eventually different groups chose to extend it in their own custom ways, and multiple new formats then became common, confusing everyone.

http://www.sequenceontology.org/gff3.shtml

GFF3
Generic Feature Format Version 3

- Gff3 format is an attempt to:
  - add and standardize the most common extensions to gff
  - preserve backward compatibility to gff
- Basics:
  - 9 columns
  - Tab delimited
  - Plain text

Backward compatibility - Maintaining compatibility with earlier models or versions of the same product. A new version of a program is said to be backward compatible if it can use files and data created with an older version of the same program.

Example line from a gff3 file:

```
Chr1   .   gene 301 2169   .   +       .   ID=SPAC1F7.08;Name=iron%20transport
```

Example line from a gff3 file:

Chr1	.	gene	301	2169	.	+	.	ID=SPAC1F7.08;Name=iron%20transport

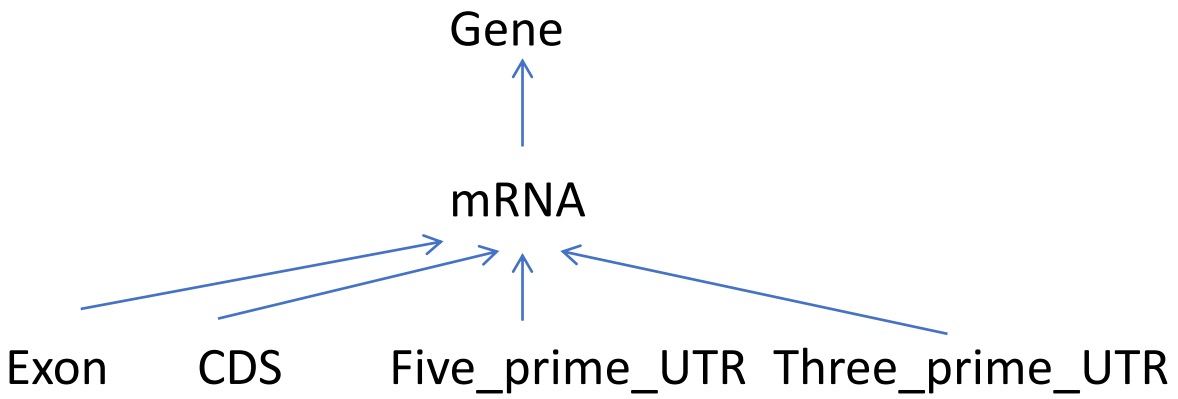
Column 1: Chromosome or source sequence – i.e. What is the background on which the feature is annotated?
Column 2: Source of the annotation – the software that did the annotation or a database
Column 3: Feature type – Must be a term or accession from the sequence ontology
Column 4: Start position of the feature, with sequence numbering starting at 1.
Column 5: End position of the feature, with sequence numbering starting at 1.
Column 6: Score – This is assigned by whatever software performs the annotation and varies by software
Column 7: Strand – defined as + (forward) or - (reverse).
Column 8: Phase - One of '0', '1' or '2'. '0' indicates that the first base of the feature is the first base of a codon, '1' that the second base is the first base of a codon, and so on.. (ONLY FOR CDS features)
Column 9: Attributes:
A list of feature attributes in the format tag=value. Multiple tag=value pairs are separated by semicolons

ID= must be unique

genome	.	mRNA	3012169	.	+	.	ID=m.SPAC1F7.08;Parent=SPAC1F7.08;Name=iron…

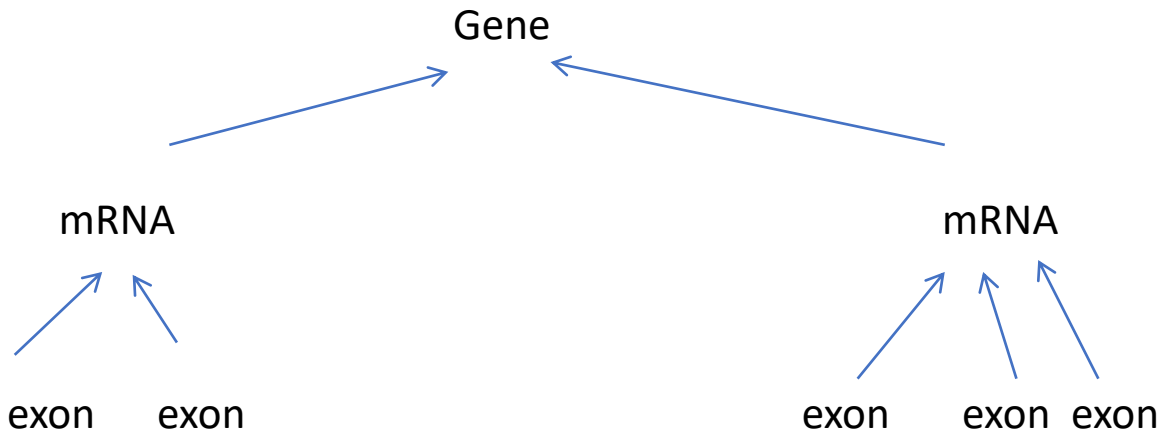
Parent=

Hierarchy of gene pieces



GFF3
Generic Feature Format Version 3

A feature can have many “children”, allowing for isoforms to be represented as well.



GFF3 – Alternative Isoforms

```
ctg123 example gene             1050 9000 . + . ID=EDEN;Name=EDEN;Note=protein kinase

ctg123 example mRNA             1050 9000 . + . ID=EDEN.1;Parent=EDEN;Name=EDEN.1;Index=1
ctg123 example five_prime_UTR   1050 1200 . + . Parent=EDEN.1
ctg123 example CDS              1201 1500 . + 0 Parent=EDEN.1
ctg123 example CDS              3000 3902 . + 0 Parent=EDEN.1
ctg123 example CDS              5000 5500 . + 0 Parent=EDEN.1
ctg123 example CDS              7000 7608 . + 0 Parent=EDEN.1
ctg123 example three_prime_UTR  7609 9000 . + . Parent=EDEN.1

ctg123 example mRNA             1050 9000 . + . ID=EDEN.2;Parent=EDEN;Name=EDEN.2;Index=1
ctg123 example five_prime_UTR   1050 1200 . + . Parent=EDEN.2
ctg123 example CDS              1201 1500 . + 0 Parent=EDEN.2
ctg123 example CDS              5000 5500 . + 0 Parent=EDEN.2
ctg123 example CDS              7000 7608 . + 0 Parent=EDEN.2
ctg123 example three_prime_UTR  7609 9000 . + . Parent=EDEN.2

ctg123 example mRNA             1300 9000 . + . ID=EDEN.3;Parent=EDEN;Name=EDEN.3;Index=1
ctg123 example five_prime_UTR   1300 1500 . + . Parent=EDEN.3
ctg123 example five_prime_UTR   3000 3300 . + . Parent=EDEN.3
ctg123 example CDS              3301 3902 . + 0 Parent=EDEN.3
ctg123 example CDS              5000 5500 . + 1 Parent=EDEN.3
ctg123 example CDS              7000 7600 . + 1 Parent=EDEN.3
ctg123 example three_prime_UTR  7601 9000 . + . Parent=EDEN.3
```