Error correction, consensus sequences and polishing

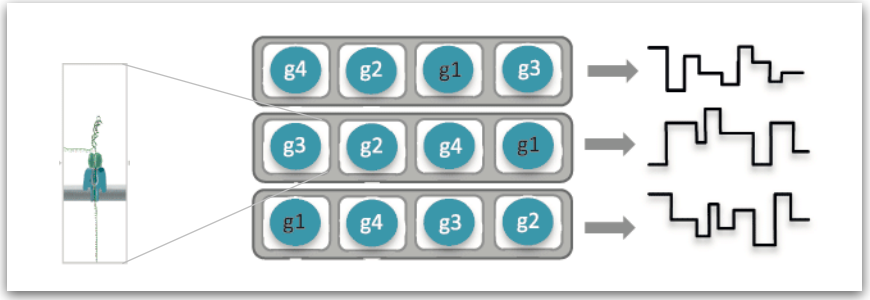
Tim Kahlke
tim.kahlke@uts.edu.au
https://github.com/timkahlke
Twitter: @AdvancedTwigTec

MinION Error profile

MinION Raw data

- Raw data detected as current
- *Segmented* into 5-mer/6-mer *events* by statistical models
- Events have a mean current, standard deviation & duration
- 2014 5-mers range from 40-70pA

  => overlaps



Ip *et al.* 2015

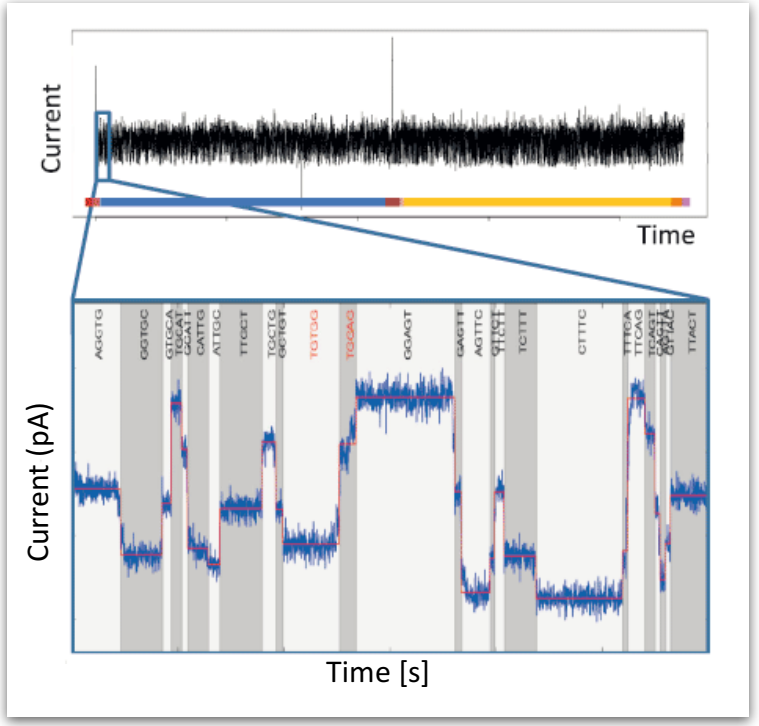
event	kmer	mean	range (1σ)
e1	AAAAA	70.2 pA	69.3 - 71.2 pA
e2	AAAAC	66.1 pA	65.4 - 66.9 pA
e3	AAACG	62.8 pA	62.0 - 63.6 pA
e4	AACGT	67.6 pA	66.3 - 68.9 pA
e5	ACGTC	56.6 pA	54.5 - 58.7 pA
e6	CGTCC	49.9 pA	49.1 - 50.7 pA

http://simpsonlab.github.io

MinION Raw data

Challenges

- Segmentation: Different event lengths
- Signal variation between pores
- Similarities between models
- Systematic errors!



Ip *et al.* 2015

Consensus
Cons -nsus
Consensus

Error correction

Polishing

Racon

- Consensus sequence module for long read alignments
- Can also perform error correction
- Similar to OLC consensus phase
- Very fast implementation
- Can be used with Minimap, HMAP and others

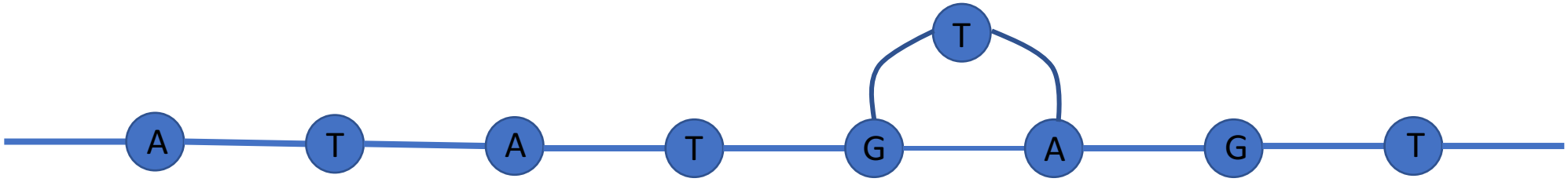
Error correction

- Use short-reads instead of long reads for mapping step
- Consensus sequence filtering of high-error rate reads is disabled

Racon – Partial Order Alignment Graph

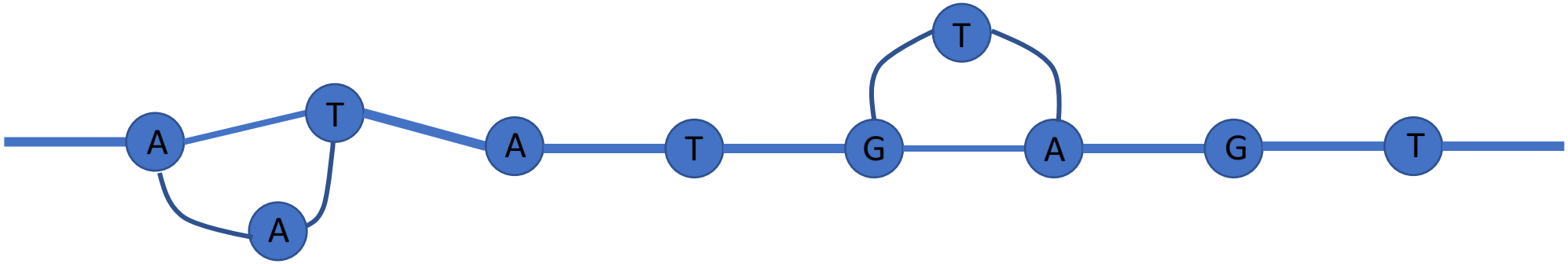
Source: Jared Simpson, ONT

Racon – Partial Order Alighment Graph



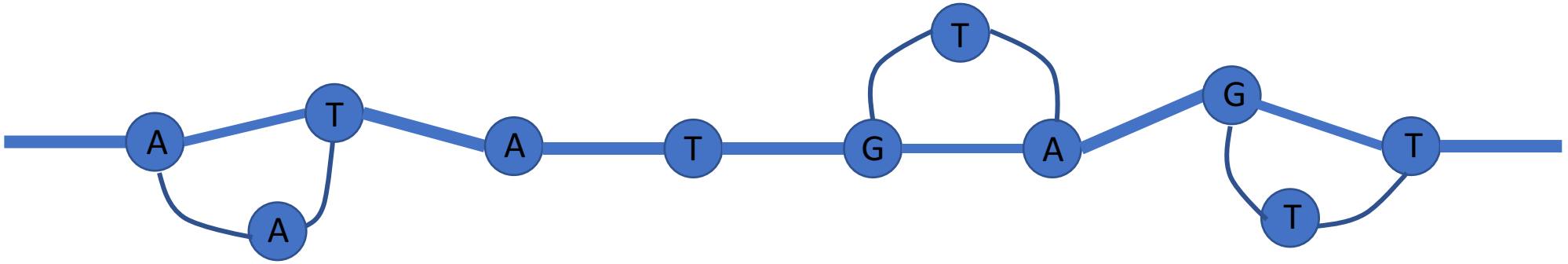
Source: Jared Simpson, ONT

Racon – Partial Order Alighment Graph



Source: Jared Simpson, ONT

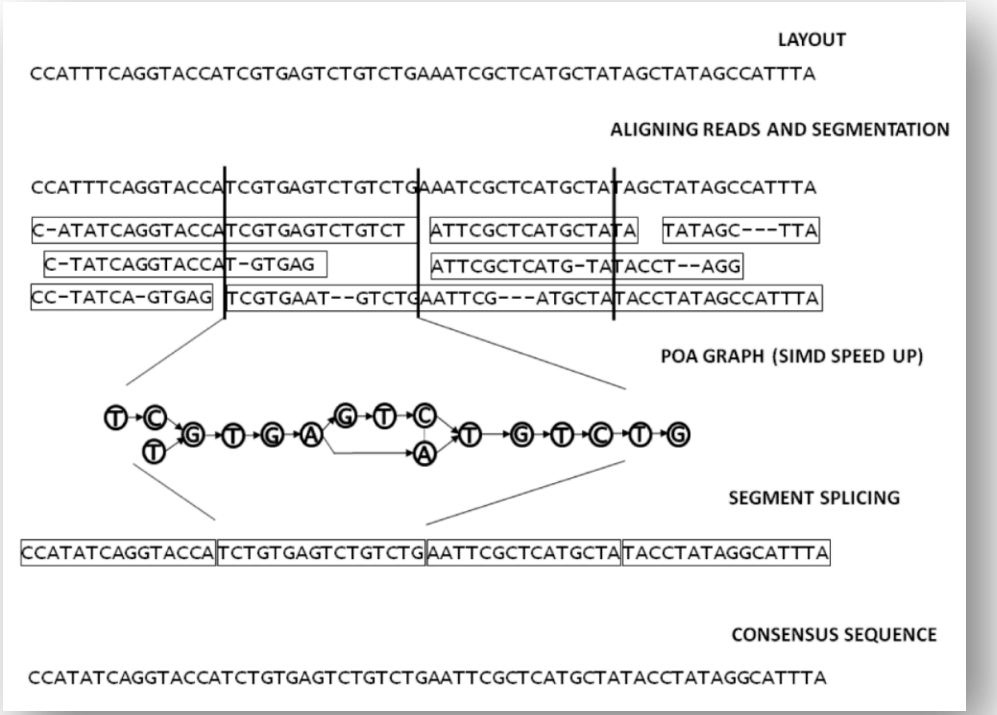
Racon – Partial Order Alighment Graph



Source: Jared Simpson, ONT

Racon

1. Map reads to (unitig) assembly
2. Align reads to assembly (e.g. minimap)
3. Create a Partial Order Alignment graph
4. Join Segments
5. Determine consensus sequence
6. Repeat step 1-5



Vaser *et al.* 2017

Canu

Canu

- Based on Celera assembler (OLC)
- Implements more RMA friendly sparse *Bets Overlap Graph* instead of full string graph (Celera)
- Splits contigs into different haplotypes (if divergense > given threshold)
- Consists of 3 stages
    1. Error correction
    2. Trimming
    3. Assembly

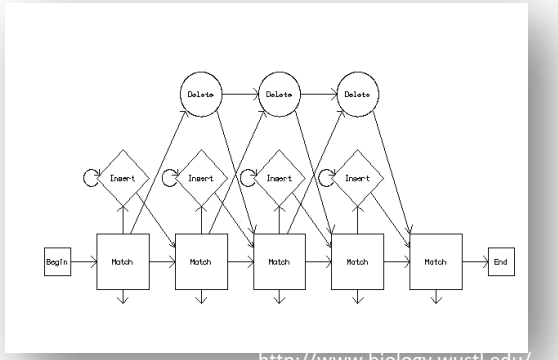
Nanopolish

Nanopolish

- Operates on the raw signal / fast5 files
- Hidden Markov Models approach
- Determine probability of a k-mer given a specific event or squiggle

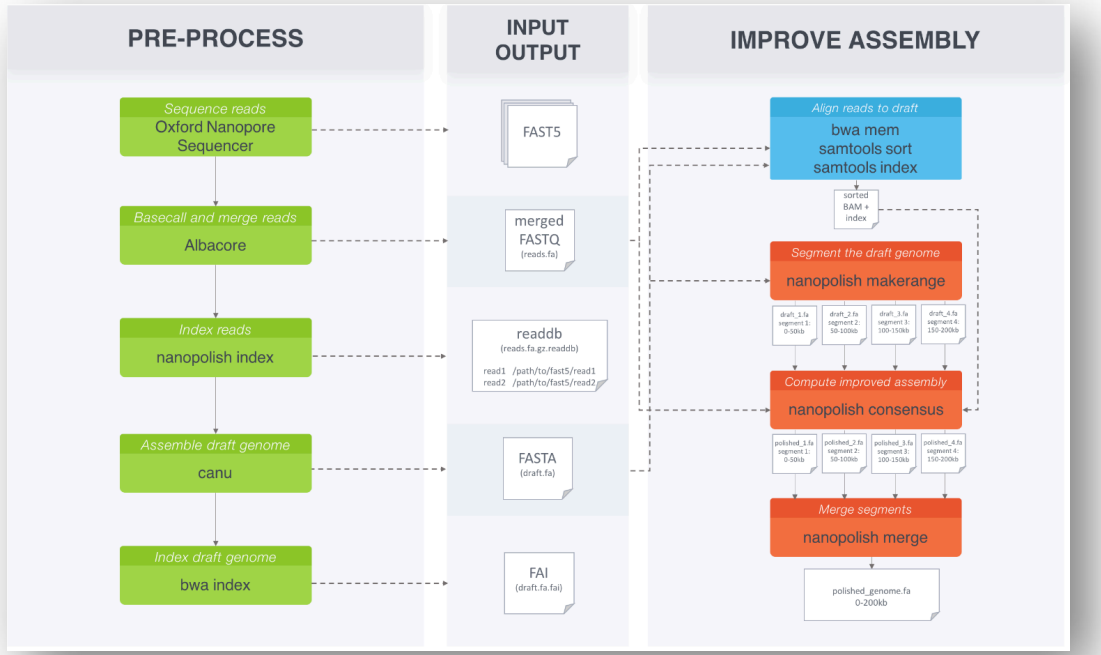
1. Mutate the sequence randomly and determine probability
2. Correct to most likely k-mer
3. Redo process until no improvements can be made

..ATGGC..
..ATGCC..
..ATGTC..



http://www.biology.wustl.edu/

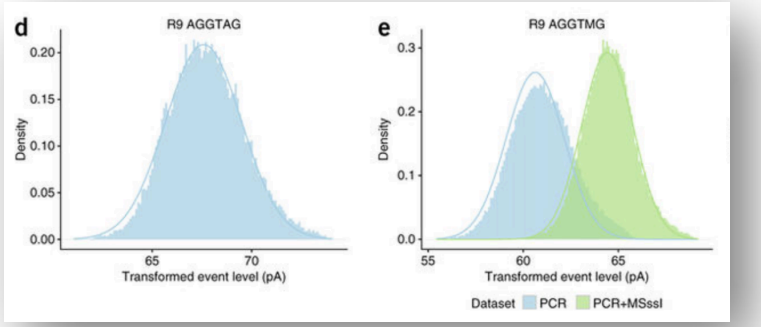
Nanopolish



http://nanopolish.readthedocs.io/en/latest/quickstart_consensus.html

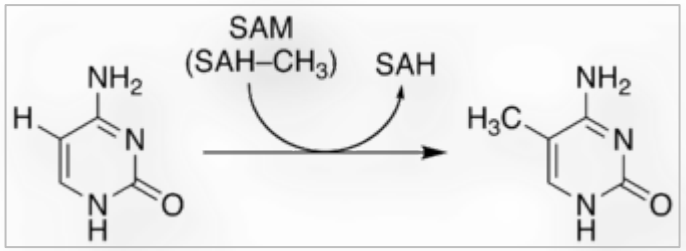
Nanopolish - methylation

- DNA modification changes event profile

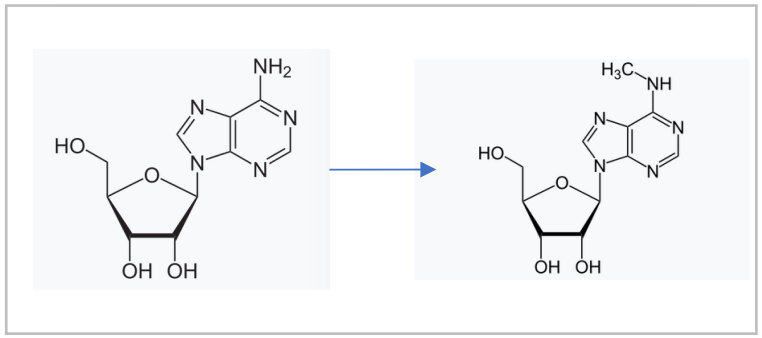


Simpson *et al* 2017

5-methylated cytosine



N5-adenosine



Nanopolish - methylation

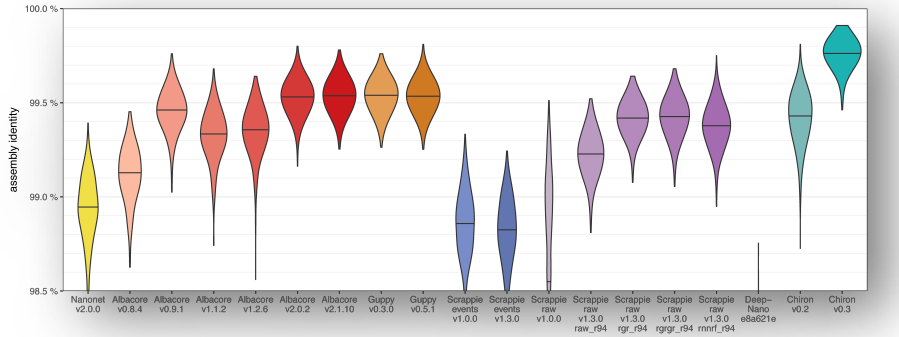
- So far implements models for two DNA modifications:
  1. Dam [G meATC]
  2. Dcm [C meC(A/T)GG])
- Can be used for error correction

Assembler comparison – Ryan Wick

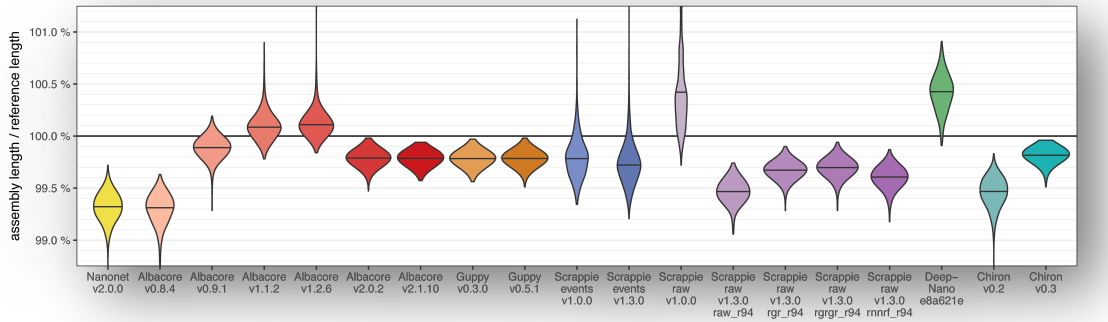
https://github.com/rrwick/Basecalling-comparison

Assembler performance

Percent identity

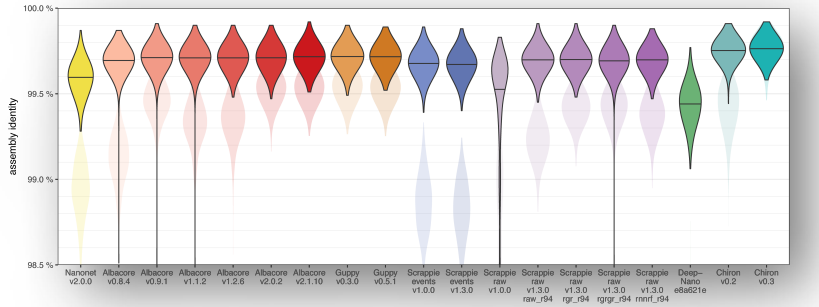


Relative length

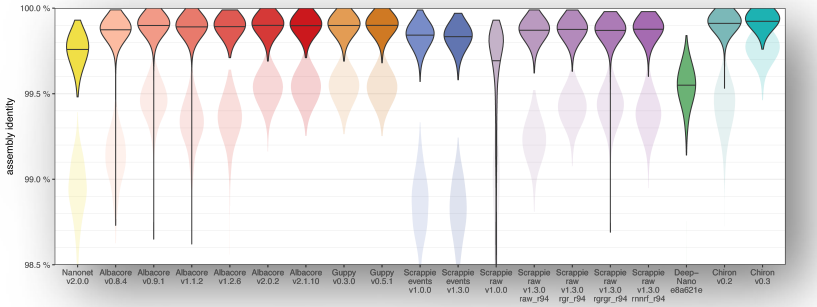


Assembler performance

Nanopolish - \wo methylation aware



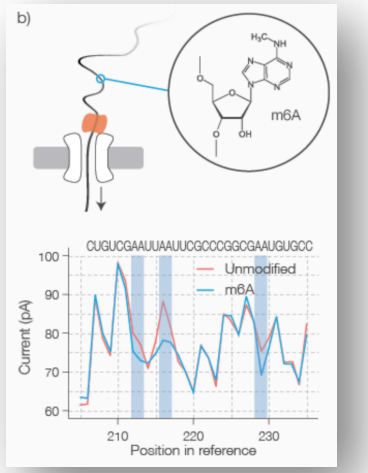
Nanopolish - \w methylation aware

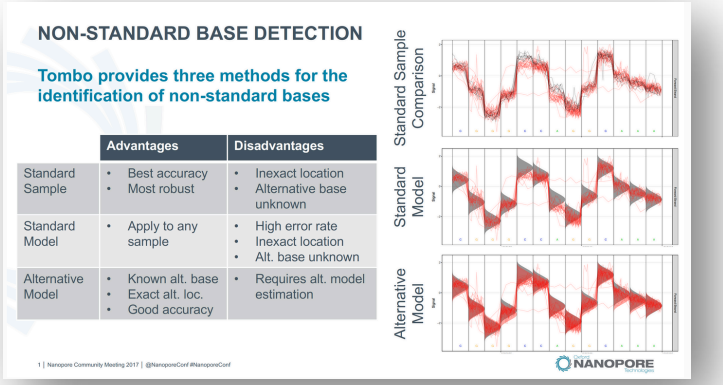


Methylation detection using Tombo

Tombo

- Similar to Nanopolish
- Tries to identify modifications in the raw signal (*re-squiggle*)
- Offers
    1. Detection of 5mA and 5mC
    2. Comparison of two samples
    3. Identificaiton of novel modifications





Questions ?