# De novo assembly of the 22Gb Loblolly Pine genome

## The biggest genome ever assembled

Aleksey Zimin

University of Maryland

on behalf of Pinerefseq consortium

## Data used for the assembly

- Exclusively Illumina short-read sequencing technology:

65x coverage by paired-end reads from a haploid single seed:

  – **1/3 of the coverage in GAIIx 160/156 overlapping pairs**

  – **2/3 of the coverage in HiSeq 100/100 pairs**

+ 13x coverage by jumping/DiTag reads from Diploid needles

- Over 16 Billion read data set!

## MaSuRCA Assembly run time/memory use

- Assembly by MaSuRCA assembler
- 64 core 1Tb memory computer
- Maximum memory usage 800gb
  - QuORUM error correction: 800Gb/10 days
  - Super reads transformation + Jumping filter: 400Gb/ 11 days
  - Contigging and Scaffolding by modified CABOG assembler: 450Gb/60+ days
  - Gap filling with super reads constructed from variable k-mer size: 300Gb/8 days

# Loblolly Pine assemblies

scaffolding needs work

MaSuRCA assembler UMD: Loblolly Pine 0.8

| | contig | scaffold |
|---|---|---|
| N50 size (Kbp) | 7 | 16 |
| Amount of sequence (Gbp) | 20.7 | 22.6 |

MaSuRCA+SOAPdenovo assembler JHU; QuORUM corrected reads/ Super reads filtered Jumping libraries:

| | contig | scaffold |
|---|---|---|
| N50 size (Kbp) | 0.687 | 55 |
| Amount of sequence (Gbp) | 19.7 | 19.7 |

Notes: N50 size was computed using genome size of 22Gb, scaffold numbers include gaps. SOAP has produced better scaffolds, but we know it is very aggressive at the cost of many errors. *The difference between the two assemblies is the post super-reads contigging/scaffolding.*

# Super-reads

Our key idea used in the assembly

- Based on the observation that most of the sequence in genomes is *locally* unique – branches are relatively rare
- We can efficiently count k-mers in the data set of all reads with Jellyfish e.g. consider 10-mers (we use much longer k of course):

```
AGCTGACTGACTGGTAACAA
AGCTGACTGA
 GCTGACTGAC
```

- Use all k-mers with counts > threshold T (e.g. T=1)
- **The idea is to make reads longer instead of breaking them into k-mers.**

## Super reads

Extending a read to become a super-read

- Consider a read – can its ends be extended uniquely?

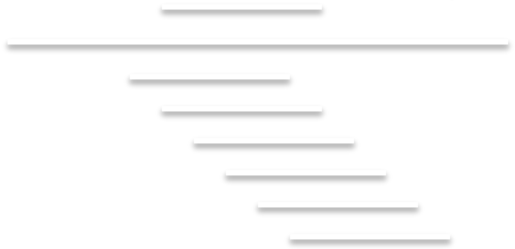
ACTGACCAGATGACCATGACAGATACATGGT

extend 5 GACTGACTGG CTGACTGGTA 10 stop

CTGACTGGTC 2

- Typically Illumina sequencing projects generate data with high coverage (>50x). With 100bp reads this implies that a new read starts on average at least every other base:

read R extended to super read S



super read S (red)

*the other reads extend to the S as well*

# Super reads

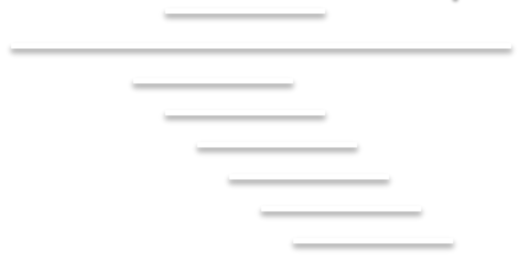
We can keep Extending on the left

- Consider a read

```
          CGACTGACCAGATGACCATGACAGATACATGGT  stop
extend  5 GACTGACTGG                    CTGACTGGTA 10 stop
extend 3 CGACTGACTG                     CTGACTGGTC  2
```

- Typically Illumina sequencing projects generate data with high coverage (>50x). With 100bp reads this implies that a new read starts on average at least every other base:

read R extended to super read S



super read S (red)

*the other reads extend to the S as well*

# Super reads

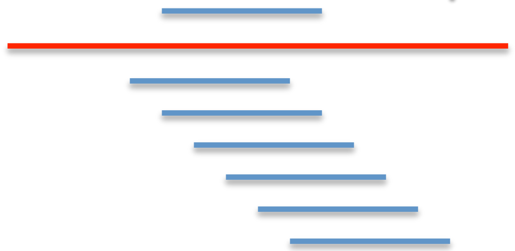
Extend, stopping at the next branch (or where there is no data)

- Consider a read

```
                CGACTGACCAGATGACCATGACAGATACATGGT  stop
extend   5  GACTGACTGG                       CTGACTGGTA  10  stop
extend  3  CGACTGACTG                        CTGACTGGTC    2
```

- Typically Illumina sequencing projects generate data with high coverage (>50x). With 100bp reads this implies that a new read starts on average at least every other base:

read R extended to super read S



super read S (red)

Many *other reads extend*

*to*

*the same S as well*

## 100 Times Fewer Super-Reads than Reads

Many read extensions stop at the same branch points

- We started with about 15 Billion paired end reads, 120bp average length
- We produced ~ 150 Million super reads – **100 times fewer reads!**
- The super reads contained 52Gb of sequence
- 50% of that sequence was in 500 bp or longer super reads

# Long mate pair libraries

- All long mate pairs were produced from diploid needles on GAIIx
- Only used pairs where both reads were covered by k-mers (used k=52) from haploid PE (megagametophyte) data
- We filtered the jumping libraries and DiTag libraries for non-junction and redundant pairs

| Library type | Mean/ stdev | Reads Sequenced | Reads after EC/mapping | Final both >63bp | Clone Coverage |
|---|---|---|---|---|---|
| **Jumping** | 3-5kb/ 200-300b | 1,666M | 70% | 32% | 37x |
| **Di-Tag** | 36kb/ 4000b | 93M | 42% | 9% | 7x |

## Mapping assembly to finished sequence

Finished sequence from a different tree

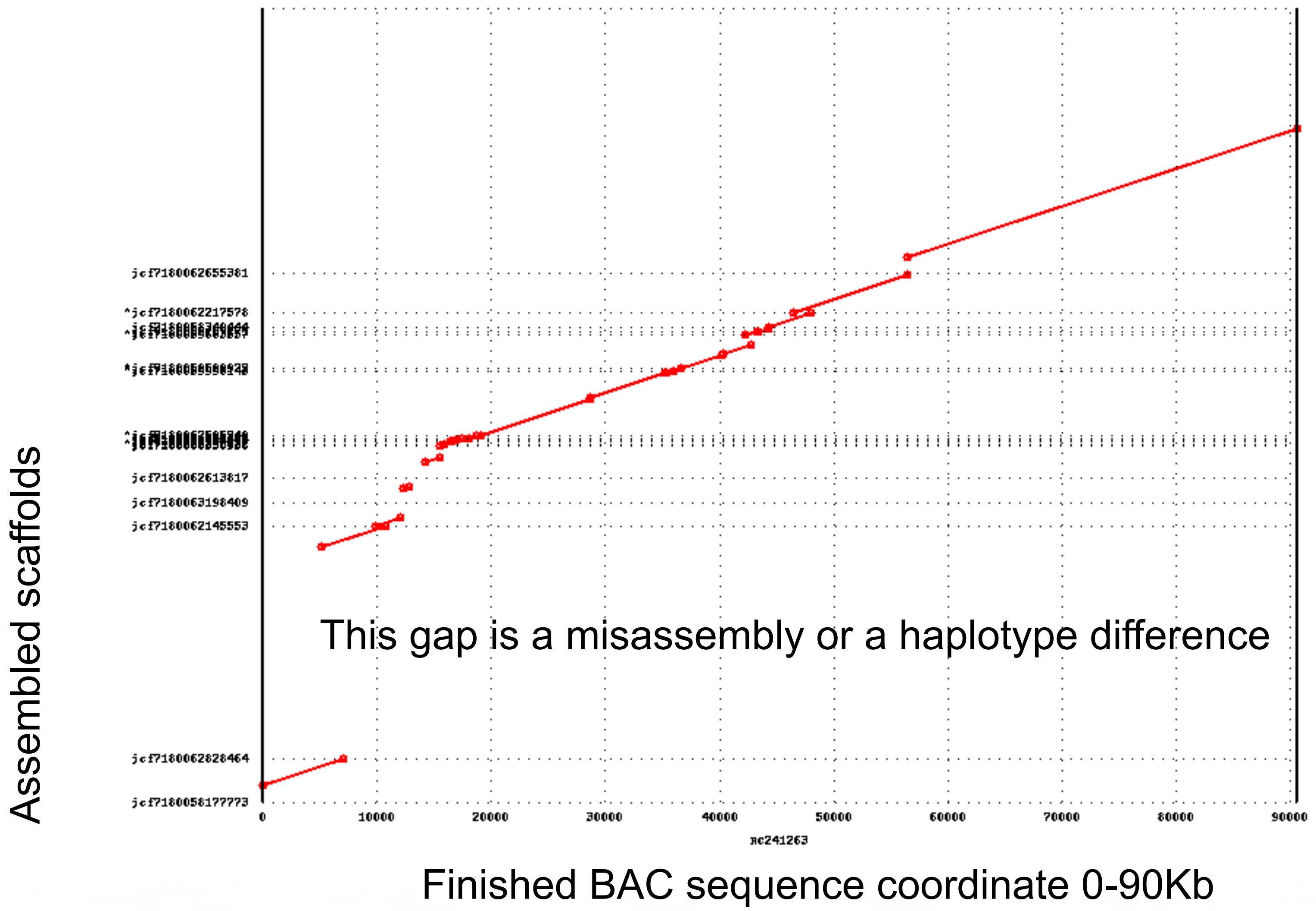
- Some ~12Mb of finished sequence available from a different tree.
- Using only the best matches considered, requiring a minimum 200bp match, 0.8 assembly covers 95% of the finished sequence with an average identity 98%.
- This suggests that there is a 2% difference between our tree and the tree used for finished sequence.

## Low resolution repeats.

- Curious fact – 40% of the sequencing reads map to this finished sequence with down to 85% identity. Such low quality repeats do not usually confuse the assembly process.

# Mapping assembly to finished sequence

An example BAC



## Further steps to improve assembly

Post-processing may help

- Additional scaffolding with unused mate pairs using a standalone scaffolder (SSPACE, Bambus2, etc)
- Reconcile scaffolds with SOAPdenovo assembly
- Use transcript data to help scaffolding – we think that introns are big
- Improve filtering – keep more valuable mate pairs
- Analyze the result and improve MaSuRCA scaffolding