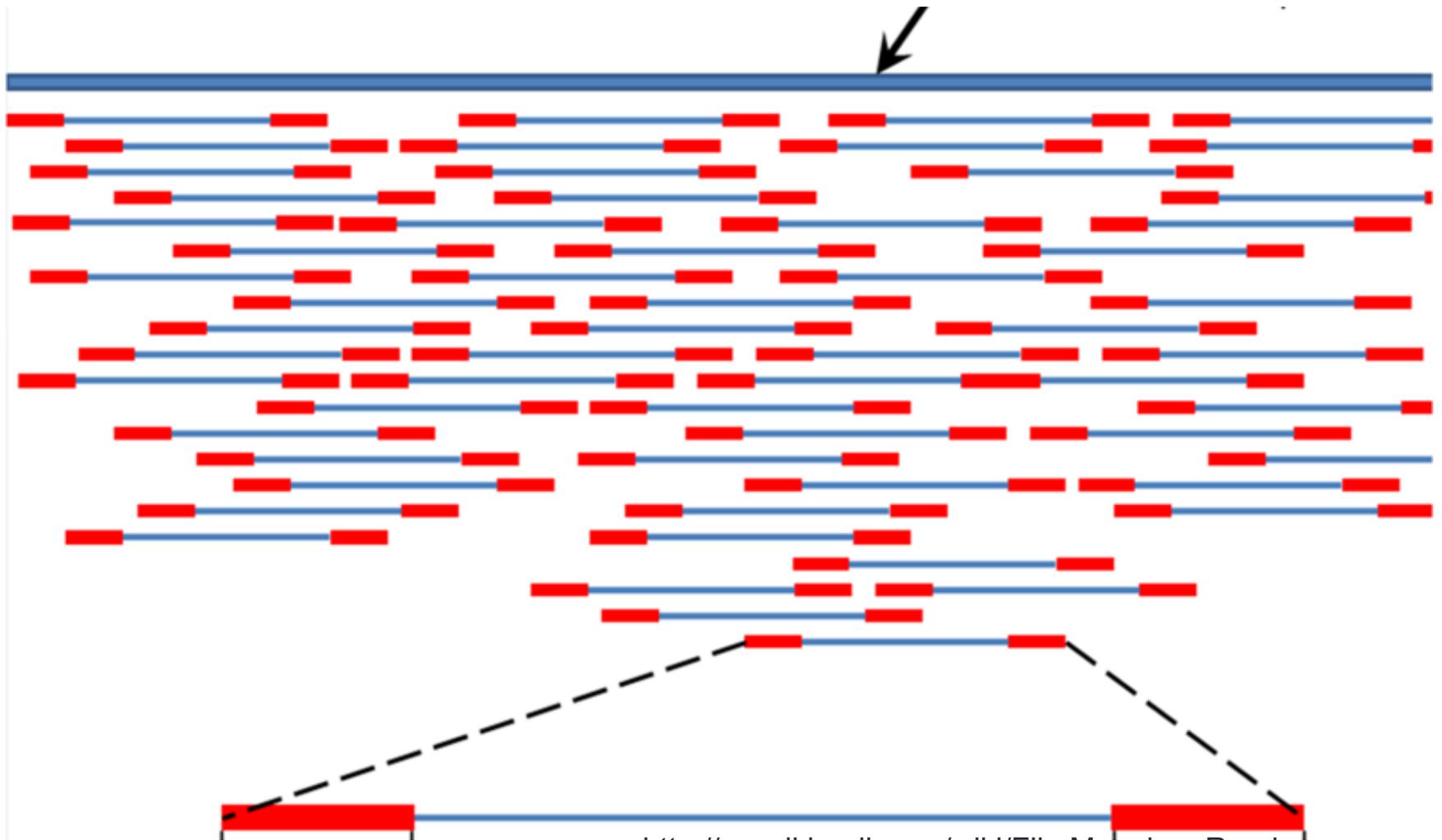


STREAMING VARIANT CALLING?

C. Titus Brown
Michigan State University

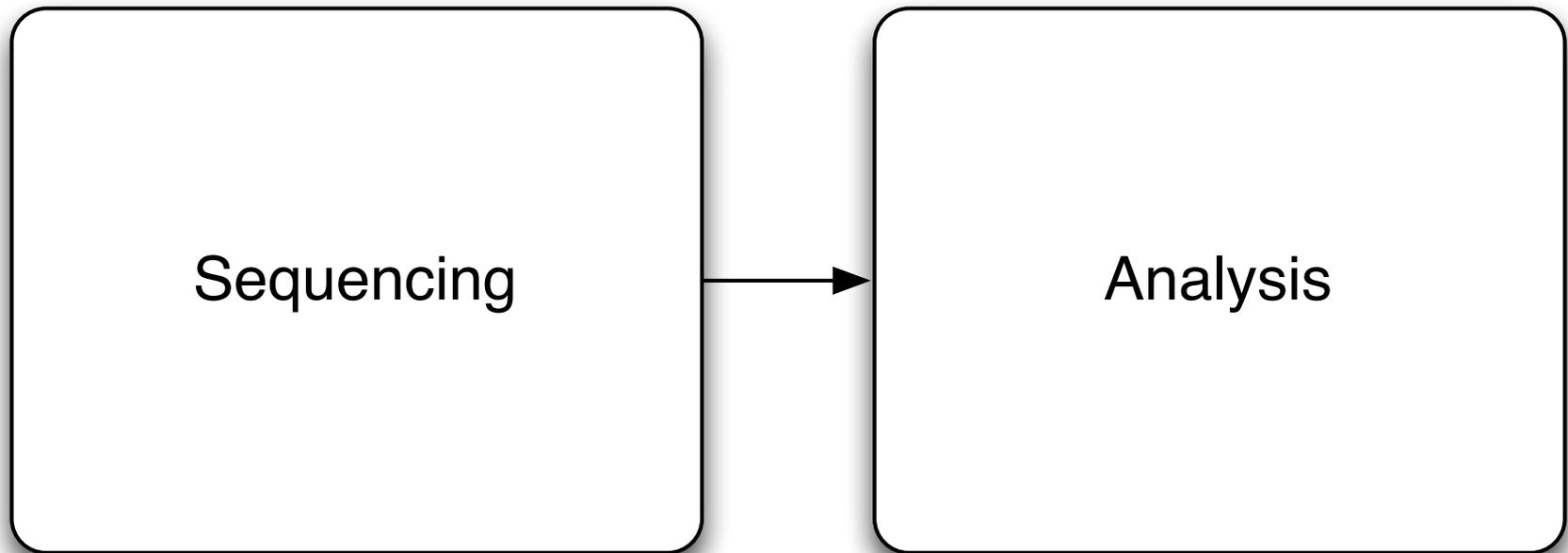
Mapping: locate reads in reference



Variant detection after mapping

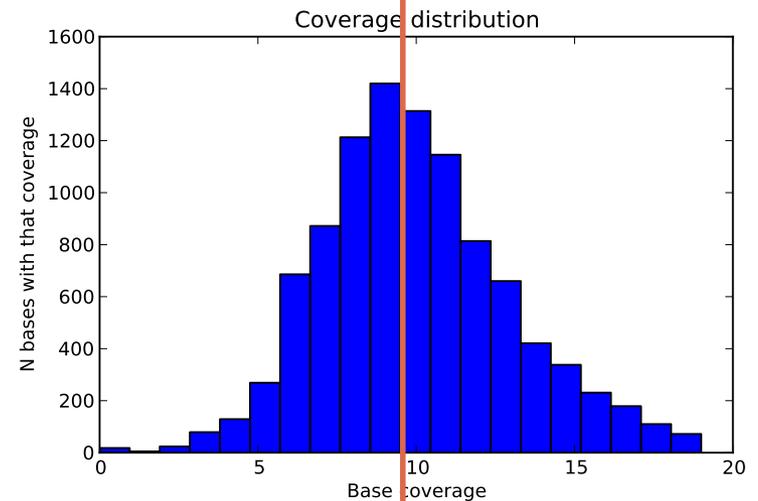
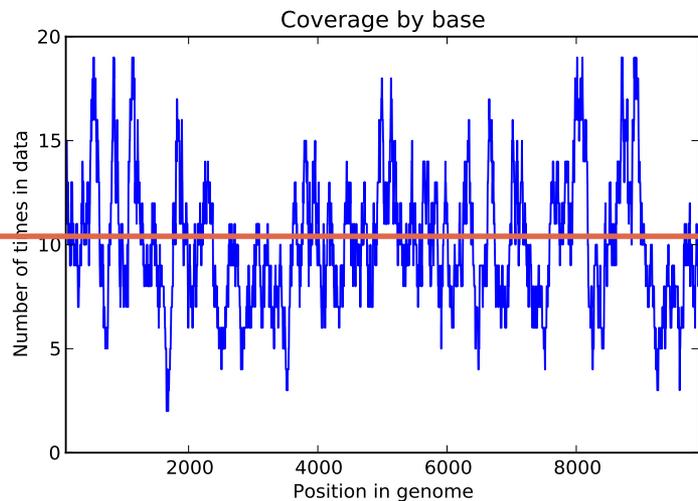


Problem 1:
Analysis is done *after* sequencing.



Problem 2:

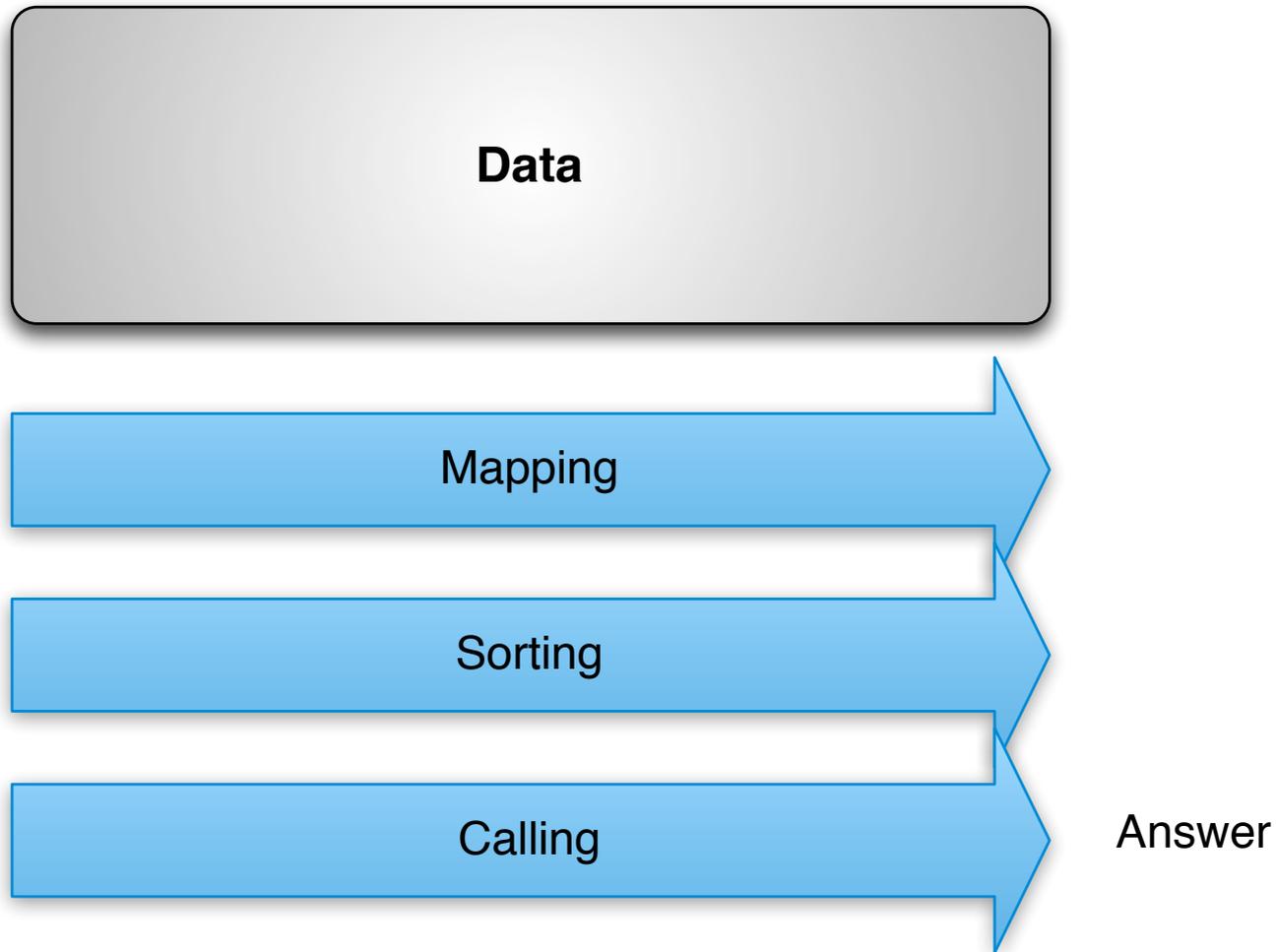
Much of your data is unnecessary.



Shotgun data is *randomly* sampled;
So, you need high coverage for high sensitivity.

Problem 3:

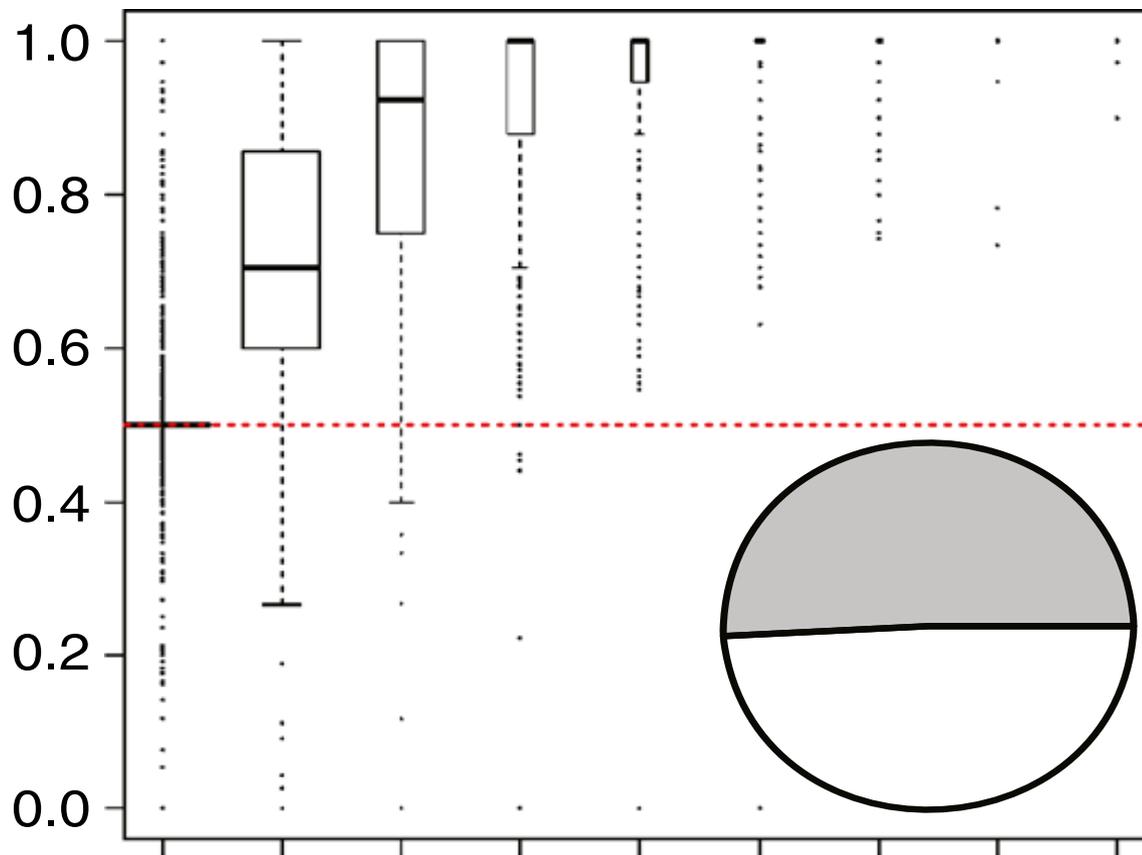
Current variant calling approaches are *multipass*



Problem 4: Allelic mapping bias favors reference genome.

Single reference genome
1 mismatch

Proportion of reference allele

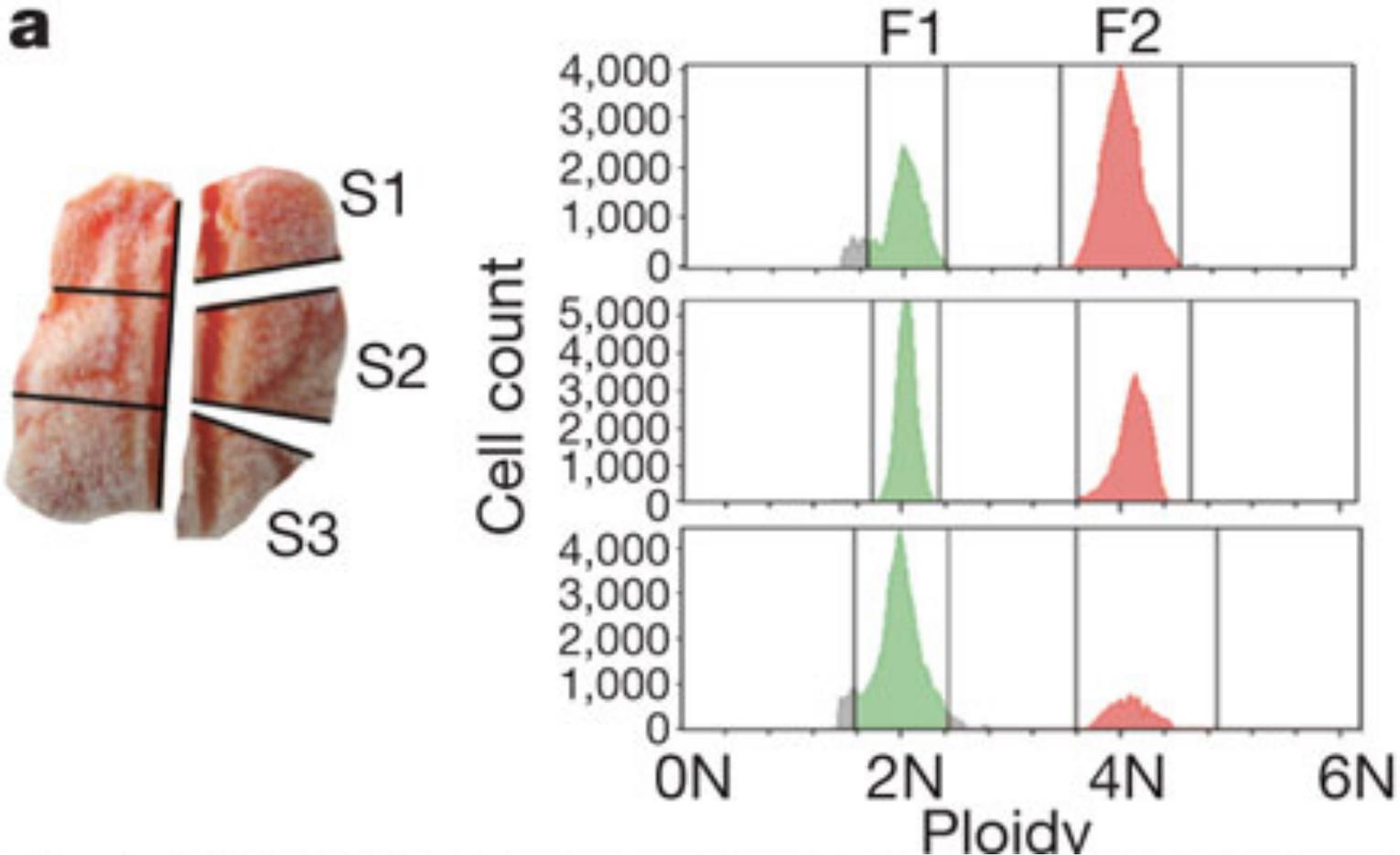


Number of nbh differentiating polymorphisms.

Stevenson et al., 2013 (BMC Genomics)

Why are we concerned at all!?

Looking forward 5 years...



Some basic math:

- 1000 single cells from a tumor...
- ...sequenced to 40x haploid coverage with Illumina...
- ...yields 120 Gbp each cell...
- ...or 120 Tbp of data.

- HiSeq X10 can do the sequencing in ~3 weeks.

- The variant calling will require 2,000 CPU weeks...

- ...so, given ~2,000 computers, can do this all in one month.

Similar math applies:

- Pathogen detection in blood;
- Environmental sequencing;
- Sequencing rare DNA from circulating blood.

- Two issues:
 - **Volume of data & compute infrastructure;**
 - ***Latency* for clinical applications.**

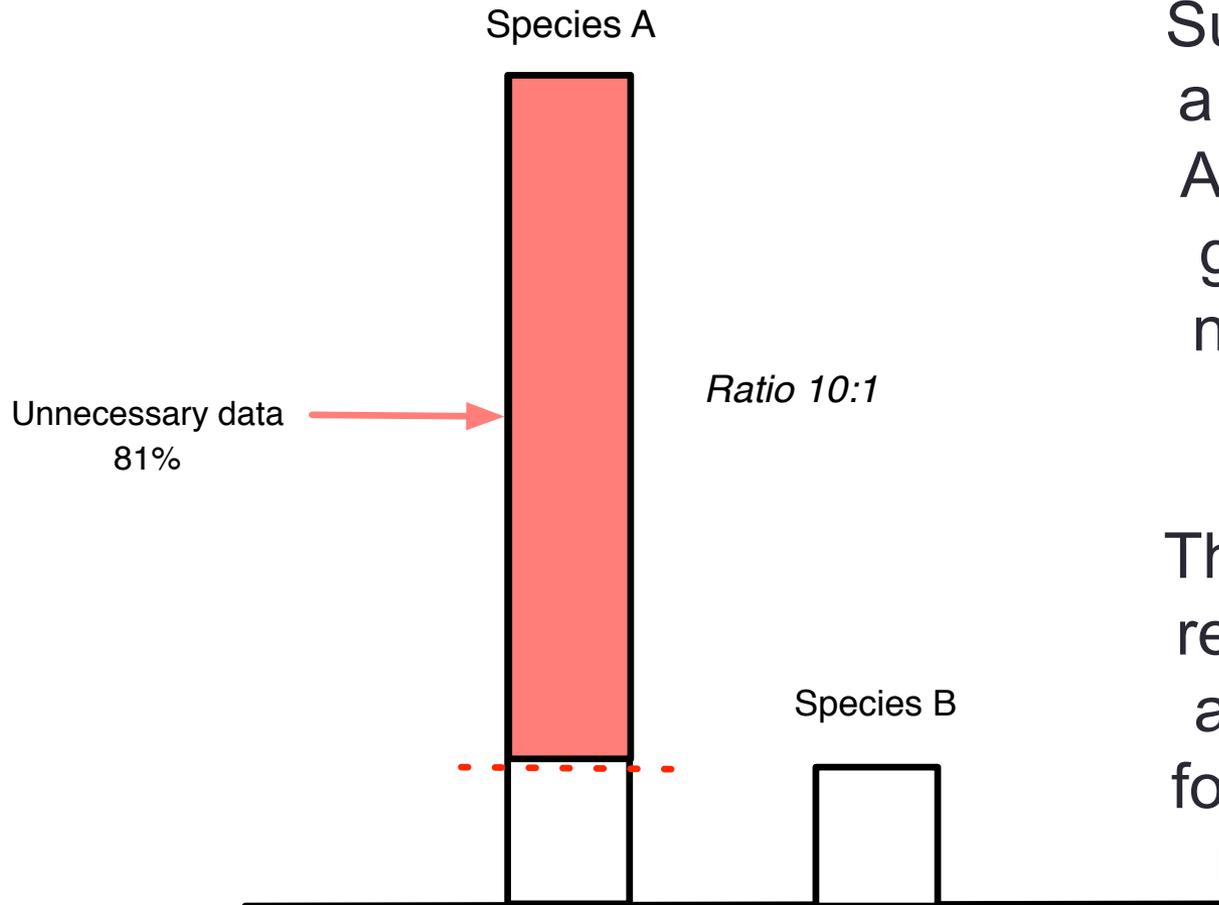
Can we improve this situation?

- Tie directly into machine as it generates sequence (Illumina, PacBio, and Nanopore can all do streaming, in theory)
- Analyze data as it comes off; for some (many?) applications, can stop run early if signal detected.
- Avoid using a reference genome for primary variant calling.
 - Easier indel detection, less allelic mapping bias
 - Can use reference for *interpretation*.

Does such a magical approach exist!?

~Digression: Digital normalization

(a computational version of library normalization)



Suppose you have a dilution factor of A (10) to B(1). To get 10x of B you need to get 100x of A! Overkill!!

The high-coverage reads in sample A are *unnecessary* for assembly, and, in fact, *distract*.

Digital normalization

----- True sequence (unknown)

Reads
(randomly sequenced)

Digital normalization

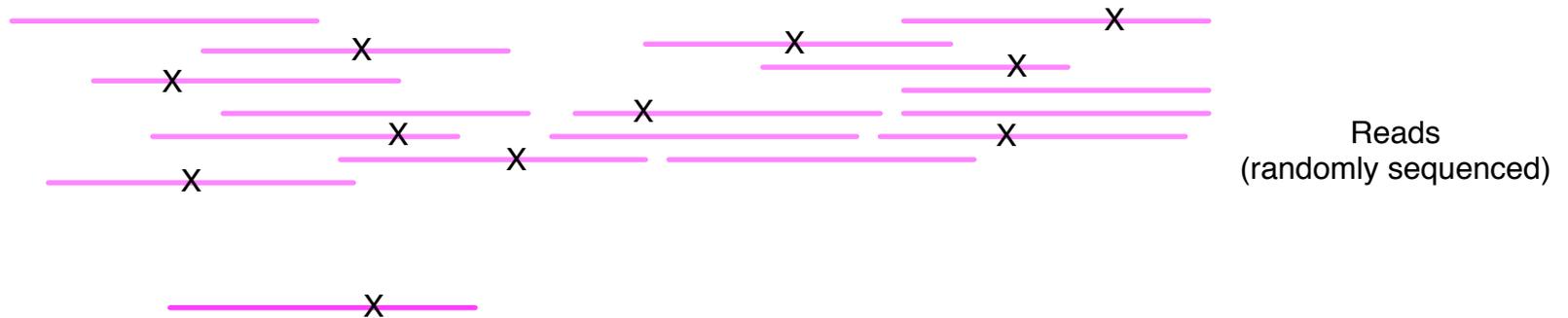
----- True sequence (unknown)



Reads
(randomly sequenced)

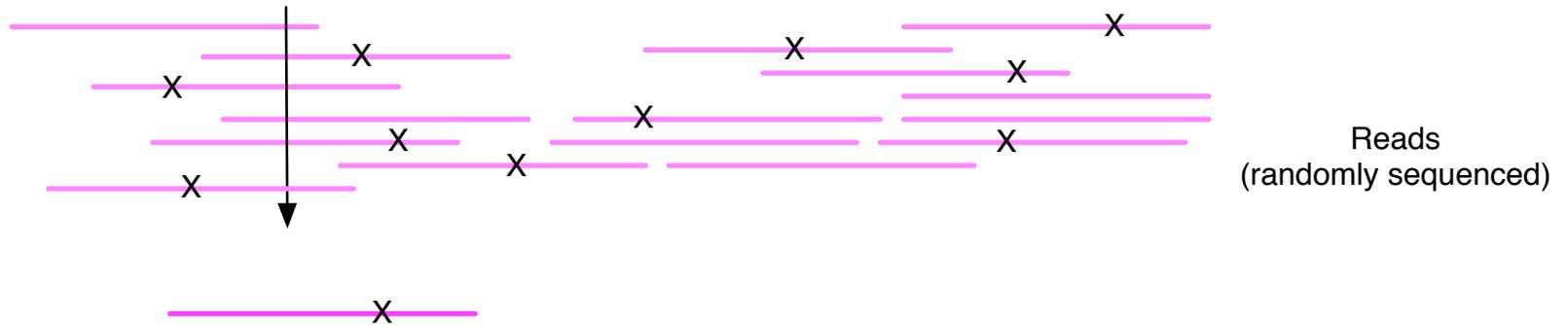
Digital normalization

----- True sequence (unknown)



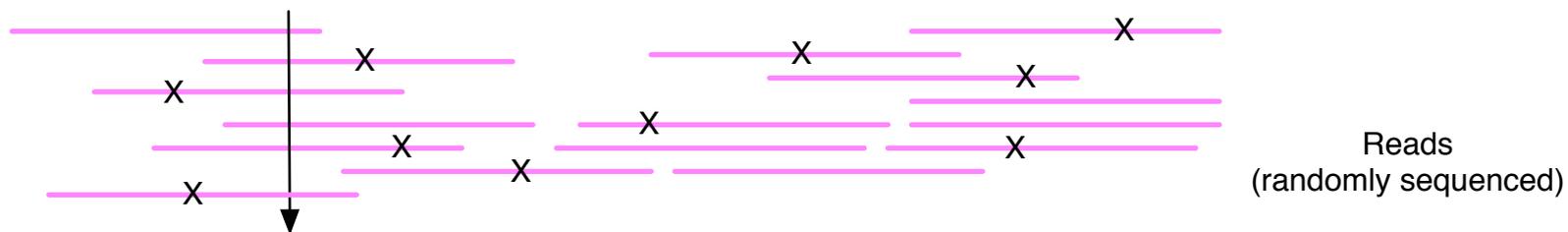
Digital normalization

----- True sequence (unknown)



Digital normalization is *streaming*

----- True sequence (unknown)

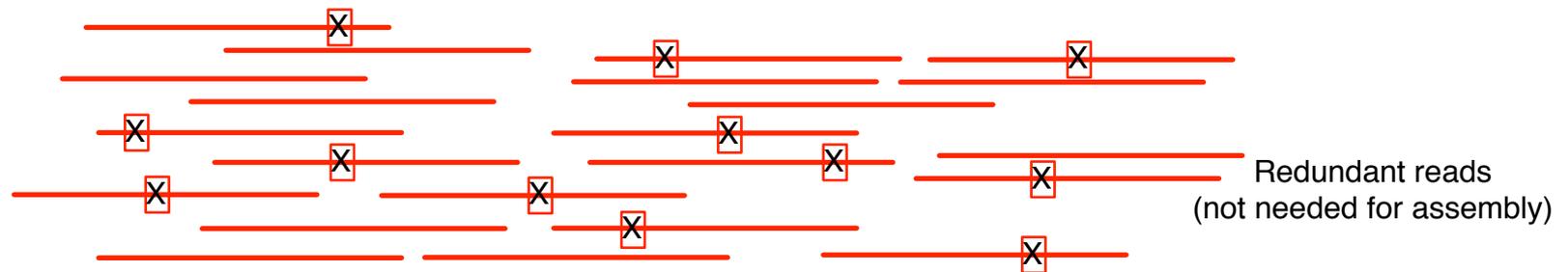
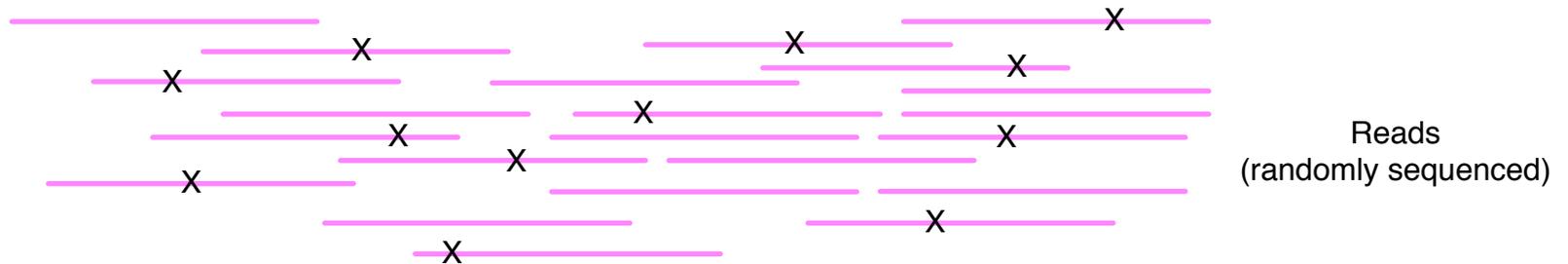


----- X -----

If next read is from a high coverage region - **discard**

Digital normalization

----- True sequence (unknown)



Some key points --

- Digital normalization is *streaming*.
- Digital normalizing is *computationally efficient* (lower memory than other approaches; parallelizable/multicore; single-pass)
- Currently, primarily used for prefiltering for *assembly*, but relies on underlying abstraction (De Bruijn graph) that is also used in *variant calling*.

Assembly now scales with richness, not diversity.

Table 3. Three-pass digital normalization reduces computational requirements for contig assembly of genomic data.

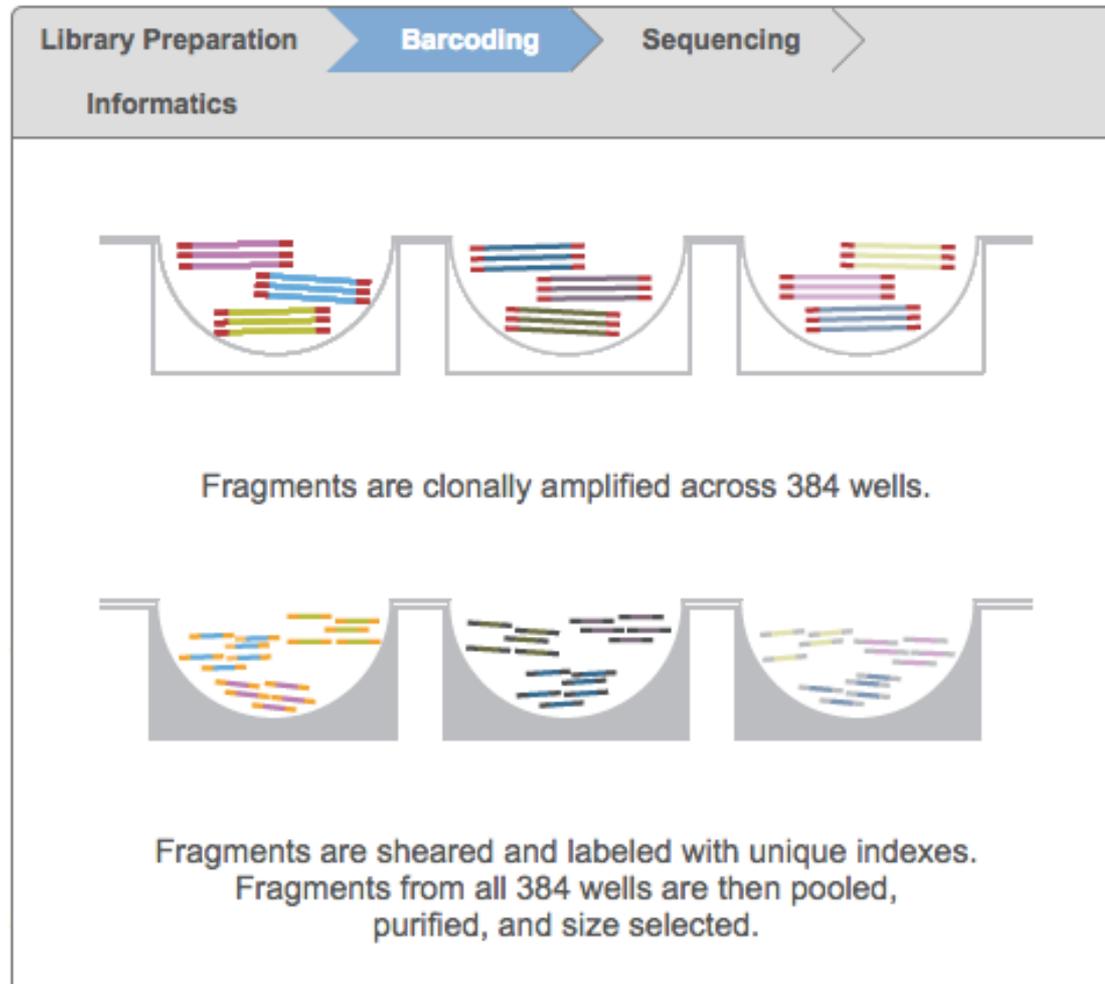
Data set	N reads pre/post	Assembly time pre/post	Assembly memory pre/post
<i>E. coli</i>	31m / 0.6m	1040s / 63s (16.5x)	11.2gb / 0.5 gb (22.4x)
<i>S. aureus</i> single-cell	58m / 0.3m	5352s / 35s (153x)	54.4gb / 0.4gb (136x)
<i>Deltaproteobacteria</i> single-cell	67m / 0.4m	4749s / 26s (182.7x)	52.7gb / 0.4gb (131.8x)

- 10-100 fold decrease in memory requirements
- 10-100 fold speed up in analysis

Diginorm is widely useful:

1. Assembly of the *H. contortus* parasitic nematode genome, a “high polymorphism/variable coverage” problem. (Schwarz et al., 2013; pmid 23985341)
2. Reference-free assembly of the lamprey (*P. marinus*) transcriptome, a “big assembly” problem. (in prep)
3. *Osedax* symbiont metagenome, a “contaminated metagenome” problem (Goffredi et al, 2013; pmid 24225886)

Anecdotal: diginorm is used in Illumina long-read sequencing (?)



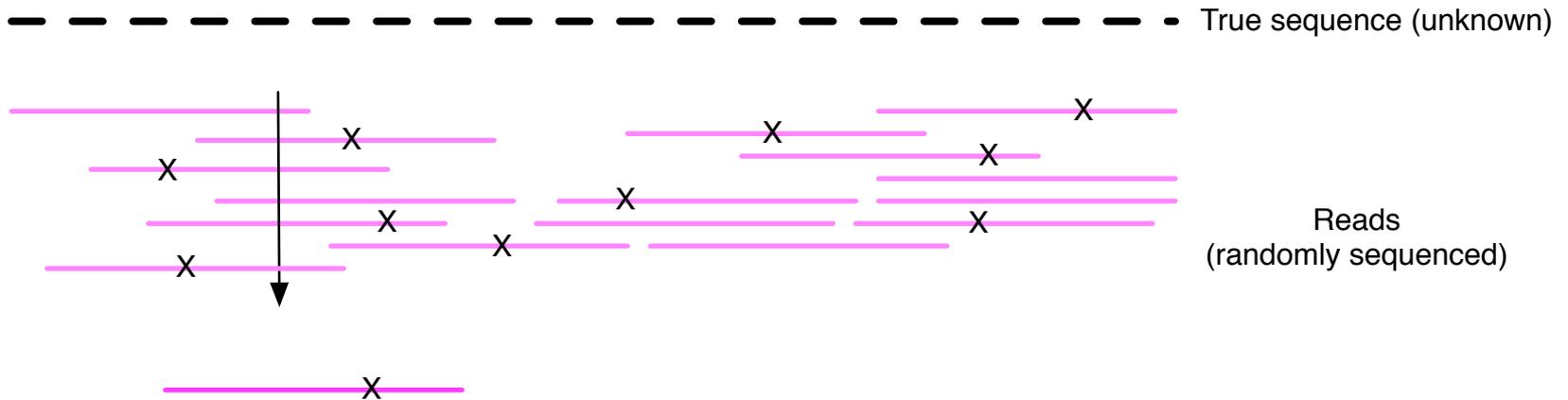
Diginorm is “lossy compression”

- Nearly perfect from an information theoretic perspective:
 - Discards 95% more of data for genomes.
 - Loses < 00.02% of information.

Table 1. Digital normalization to C=20 removes many erroneous k-mers from sequencing data sets. Numbers in parentheses indicate number of true k-mers lost at each step, based on reference.

Data set	True 20-mers	20-mers in reads	20-mers at C=20	% reads kept
Simulated genome	399,981	8,162,813	3,052,007 (-2)	19%
Simulated mRNAseq	48,100	2,466,638 (-88)	1,087,916 (-9)	4.1%
<i>E. coli</i> genome	4,542,150	175,627,381 (-152)	90,844,428 (-5)	11%
Yeast mRNAseq	10,631,882	224,847,659 (-683)	10,625,416 (-6,469)	9.3%
Mouse mRNAseq	43,830,642	709,662,624 (-23,196)	43,820,319 (-13,400)	26.4%

Digital normalization => graph alignment



What we are actually doing this stage is building a *graph* of all the reads, and aligning new reads to that graph.

Error correction via *graph alignment*

Original Sequence: AGCCGGAGGTCC**CGAATCTGAT**GGGGAGGCG
Read: AGCCGGAGGT**A**CGAATCTGATGGGGAGGCG

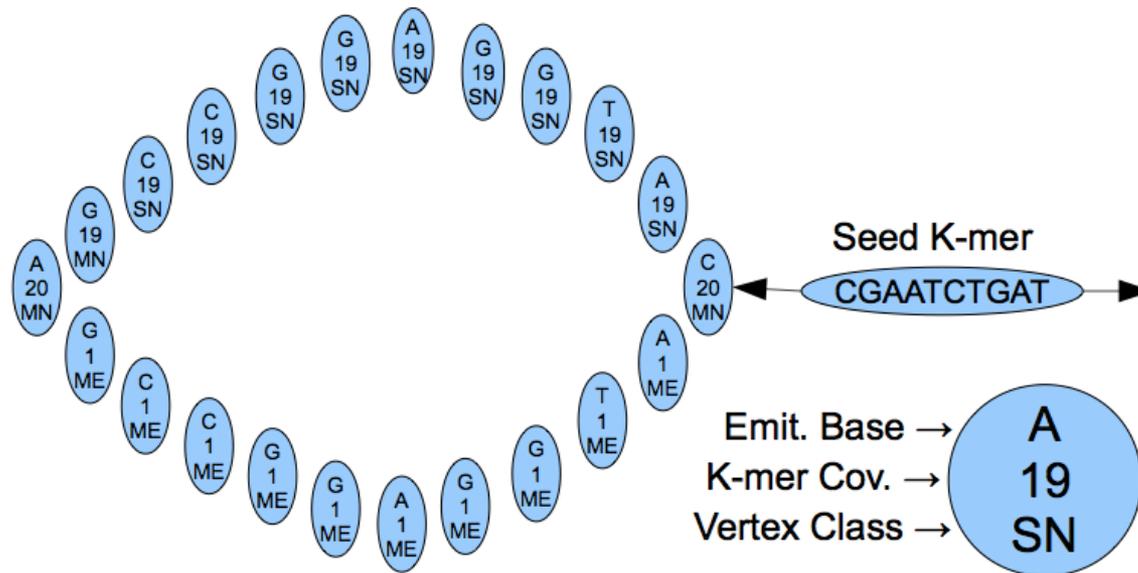


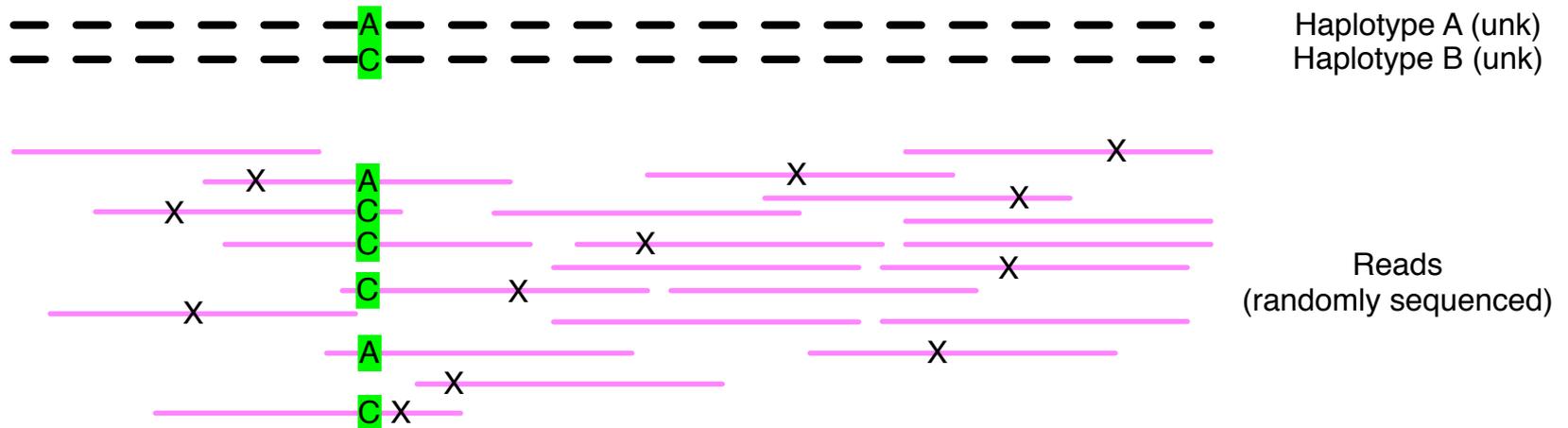
Figure 4.3: Simplified example of a graph alignment beginning at a seed k-mer. Two paths exist due to a single SNP, which forms a bubble structure. The goal is to correct the read by replacing the substituted SNP for the original base. The abbreviations for the vertex classes are as follows: MN = match and non-erroneous, ME = match and erroneous, SN = substitution and non-erroneous.

Error correction on simulated *E. coli* data

	<i>TP</i>		<i>FP</i>		<i>TN</i>		<i>FN</i>	
<i>ideal</i>	3,469,834	99.1%	8,186		460,655,449		31,731	0.9%
<i>1-pass</i>	2,827,839	80.8%	30,254		460,633,381		673,726	19.2%
<i>1.2-pass</i>	3,403,171	97.2%	8,764		460,654,871		98,394	2.8%
	(corrected)		(mistakes)		(OK)		(missed)	

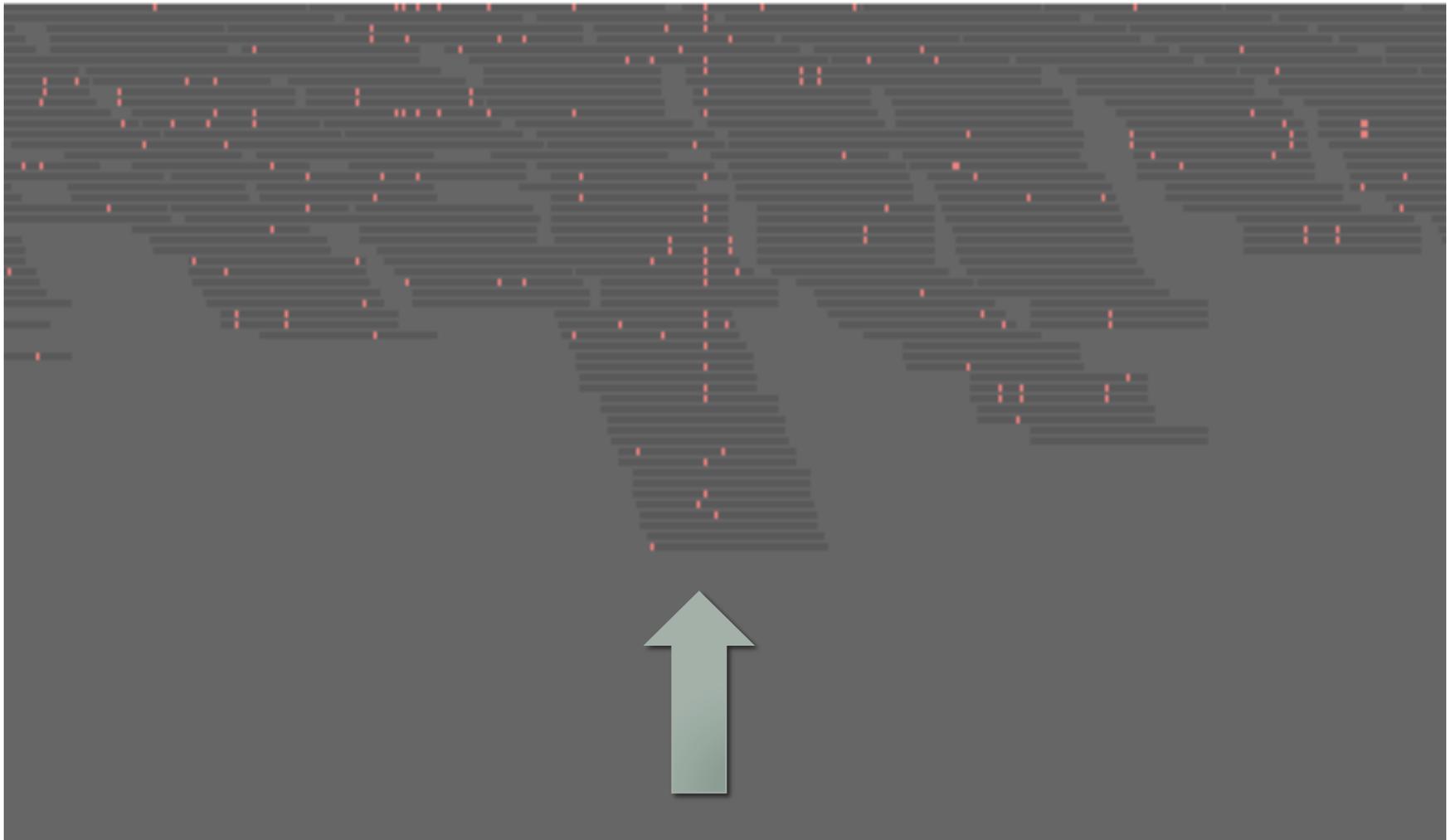
1% error rate, 100x coverage.

Error correction \Leftrightarrow variant calling

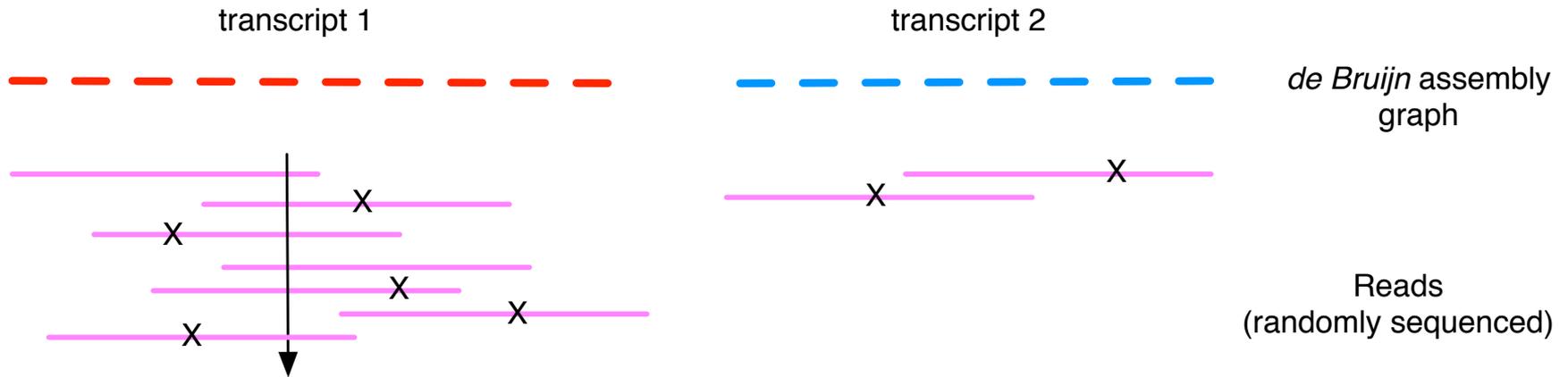


Single pass, *reference free*, tunable, streaming
online variant calling.

Coverage is adjusted to retain signal

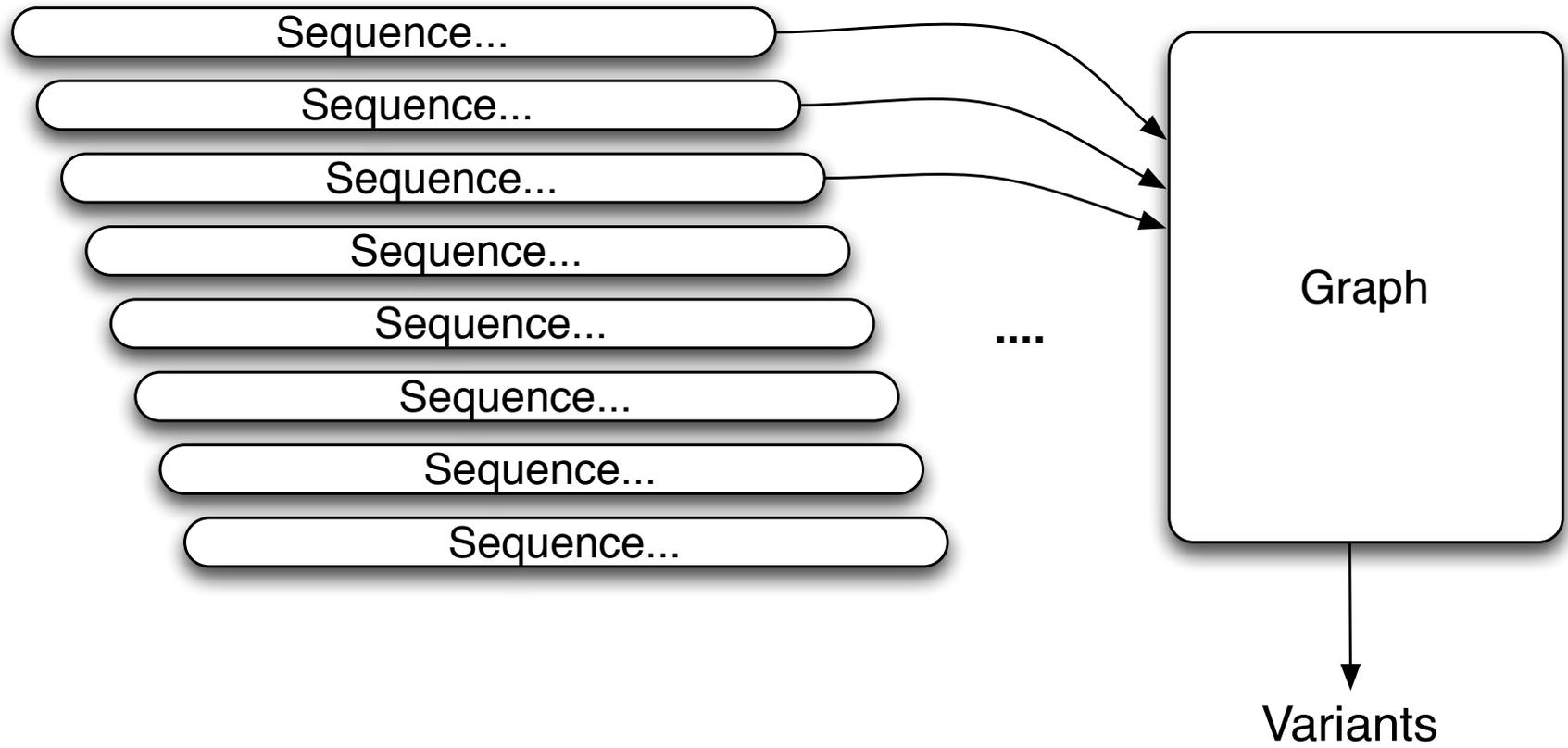


Graph alignment can detect read saturation

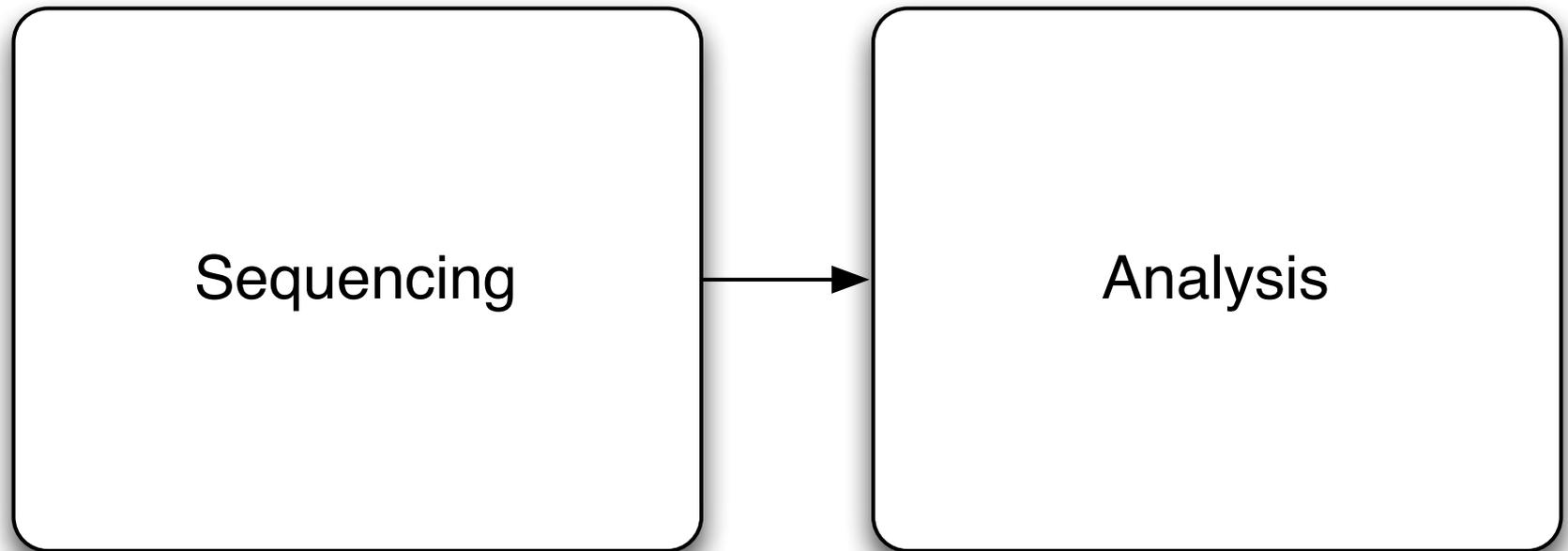


We can do *local* analysis of saturated graph components.

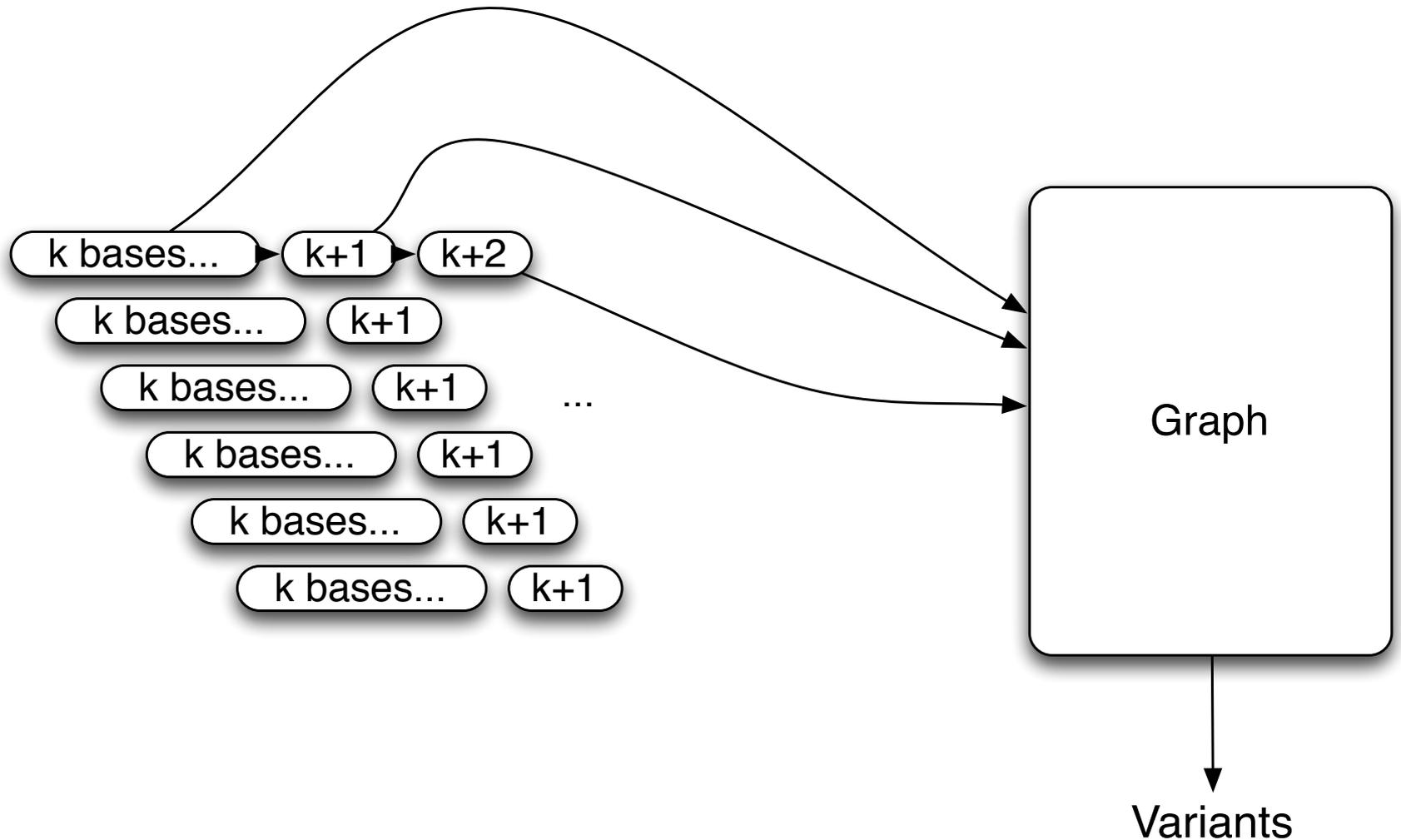
Streaming with *reads*...



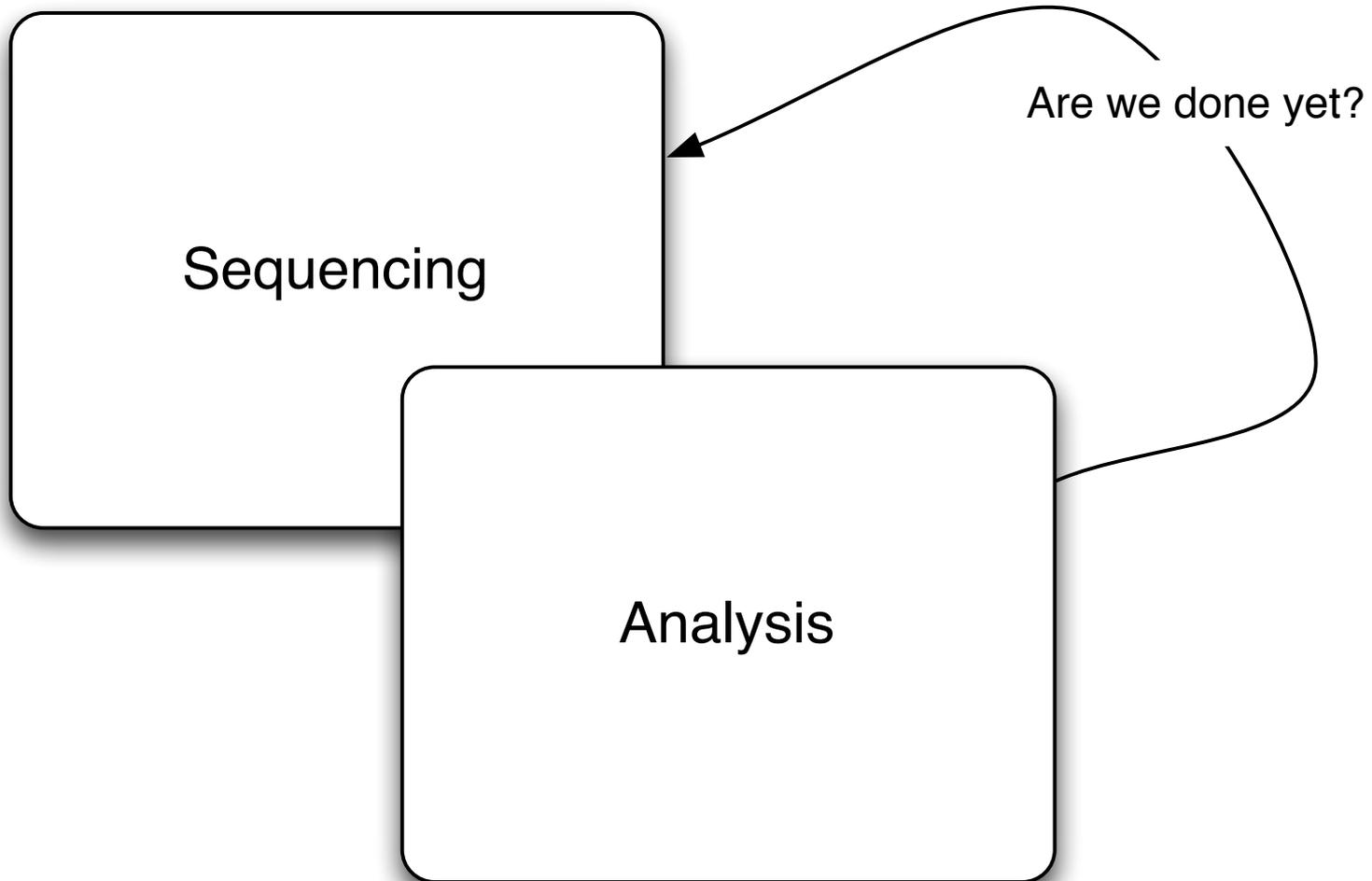
Analysis is done *after* sequencing.



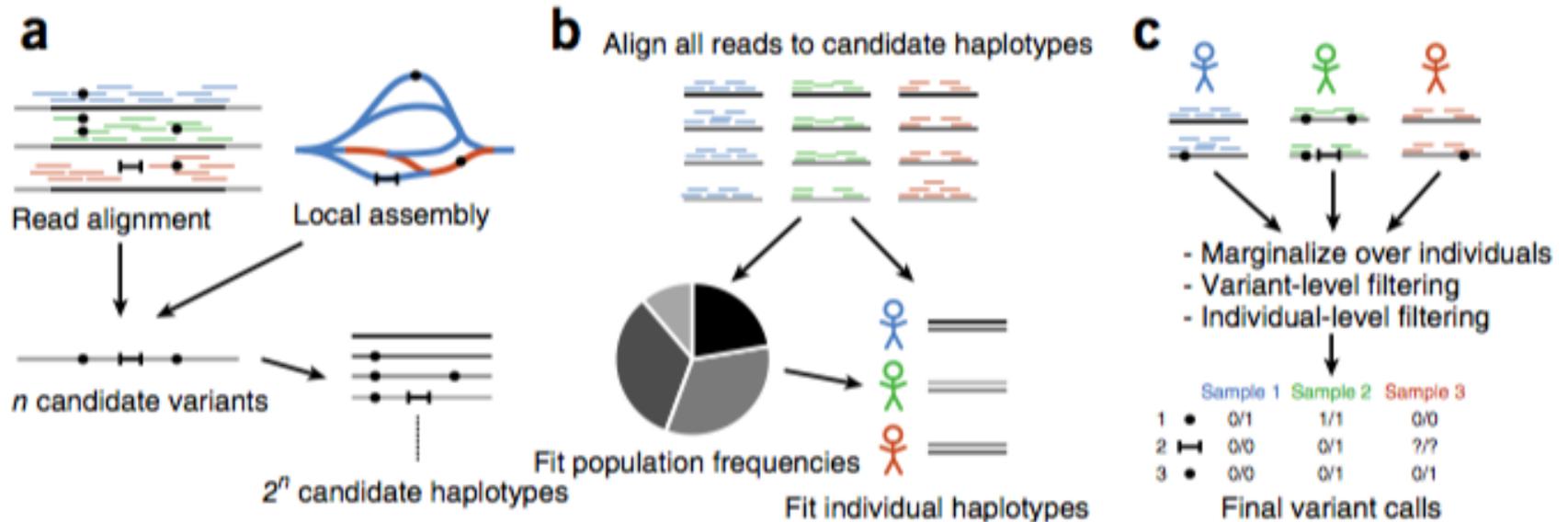
Streaming with *bases*



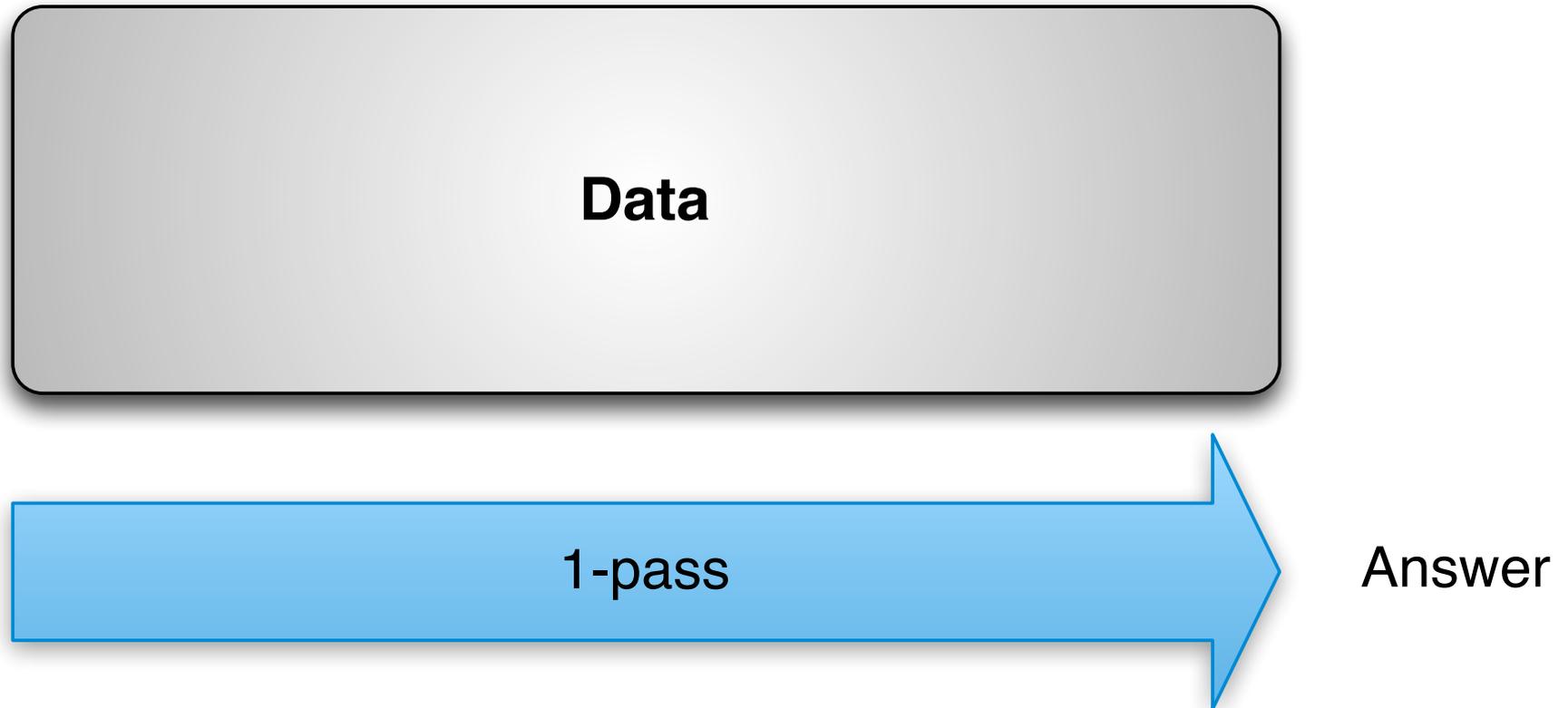
Integrate sequencing and analysis



Streaming approach also supports more compute-intensive interludes – remapping, etc.



Streaming algorithms can be very efficient



See also eXpress, Roberts et al., 2013.

So: reference-free variant calling

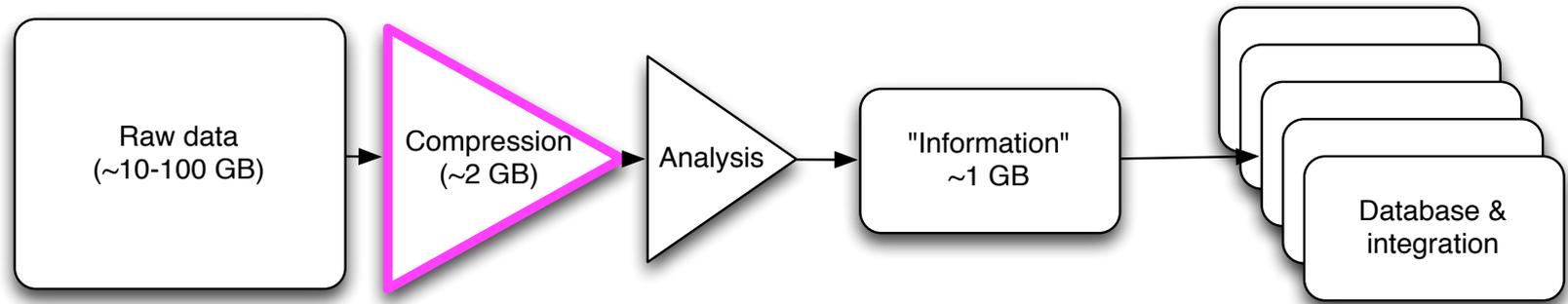
- Streaming & online algorithm; single pass.
 - For real-time diagnostics, can be applied *as bases are emitted* from sequencer.
- Reference free: independent of reference bias.
- Coverage of variants is *adaptively* adjusted to retain all signal.
- Parameters are easily tuned, although theory needs to be developed.
 - High sensitivity (e.g. $C=50$ in 100x coverage) => poor compression
 - Low sensitivity ($C=20$) => good compression.
- Can “subtract” reference => novel structural variants.
 - (See: Cortex, Zam Iqbal.)

Two other features --

- More single-computer scalable approach than current: low disk access, high parallelizability.
- Openness – our software is free to use, reuse, remix; no intellectual property restrictions. (Hence “We hear Illumina is using it...”)

Prospectus for streaming variant detection

- Underlying concept is sound and offers many advantages over current approaches;
- We have proofs of concept implemented;
- We know that underlying approach works well in amplification situations, as well;
- Tuning and math/theory needed!
- ...grad students keep on getting poached by Amazon and Google. (This is becoming a serious problem.)



Lossy compression can substantially reduce data size while retaining information needed for later (re)analysis.

Lossy compression



<http://en.wikipedia.org/wiki/JPEG>

Lossy compression



<http://en.wikipedia.org/wiki/JPEG>

Lossy compression



<http://en.wikipedia.org/wiki/JPEG>

Lossy compression

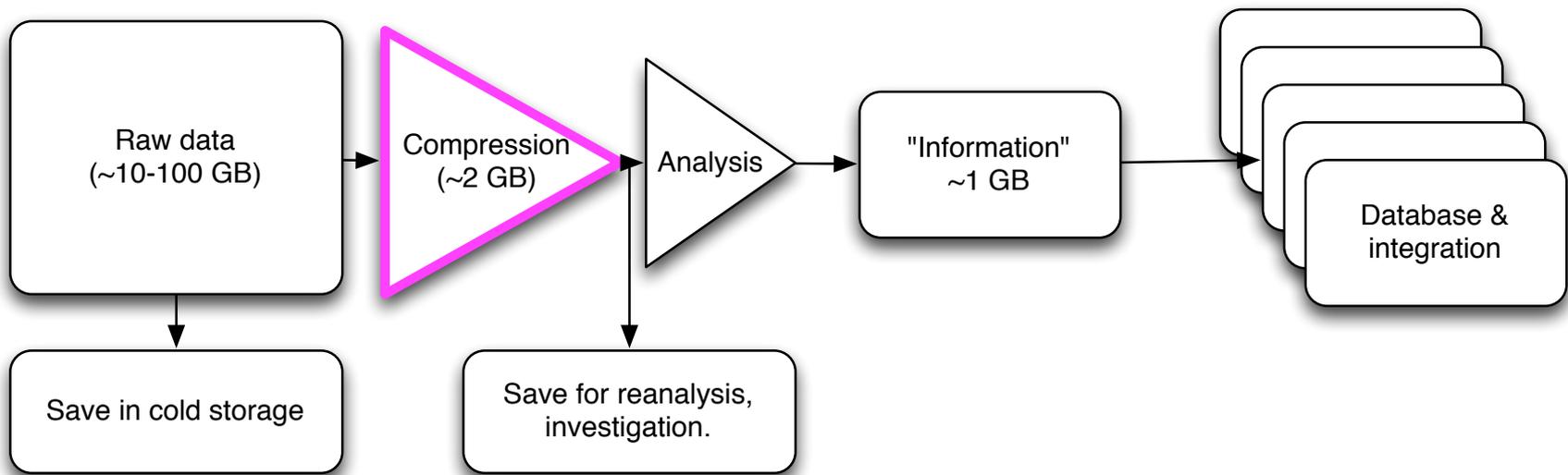


<http://en.wikipedia.org/wiki/JPEG>

Lossy compression



<http://en.wikipedia.org/wiki/JPEG>



Data integration?

Once you have all the data, what do you do?

"Business as usual simply cannot work."

Looking at *millions* to *billions* of genomes.

(David Haussler, 2014)

Data recipes

Standardized (versioned, open, remixable, cloud) pipelines and *protocols* for sequence data analysis.

See: [khmer-recipes](#), [khmer-protocols](#).

Increases buy-in :)

Training!

Lots of training planned at Davis –
open workshops.

ivory.idyll.org/blog/2014-davis-and-training.html

Increases buy-in x 2!

Acknowledgements

Lab members involved

- Adina Howe (w/Tiedje)
- **Jason Pell**
- Qingpeng Zhang
- Tim Brom
- **Jordan Fish**
- **Michael Crusoe**

Collaborators

- Jim Tiedje, MSU
- Billie Swalla, UW
- Janet Jansson, LBNL
- Susannah Tringe, JGI
- Eran Andrechek, MSU

Funding

USDA NIFA; NSF IOS;
NIH NHGRI; NSF
BEACON.