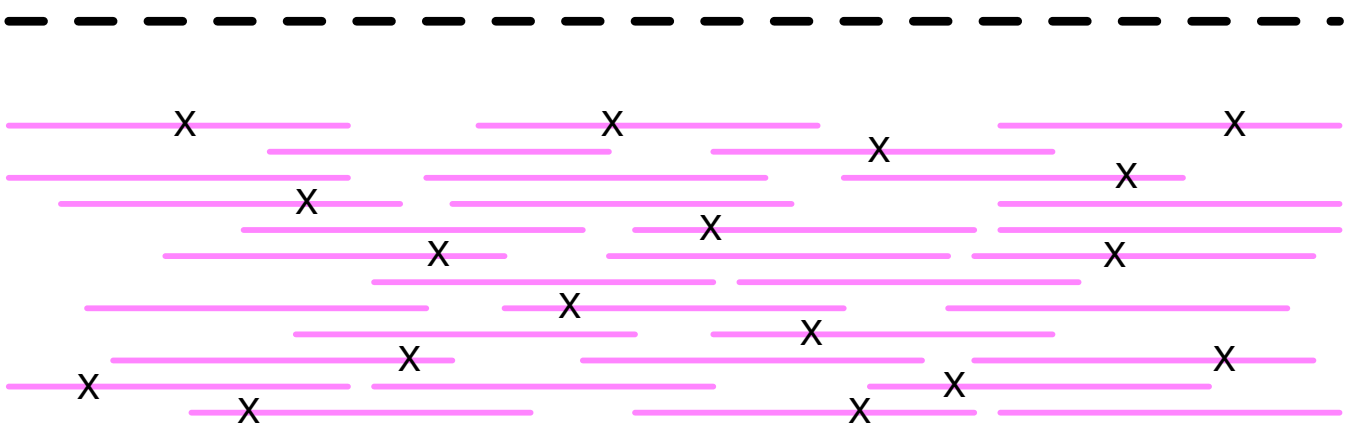
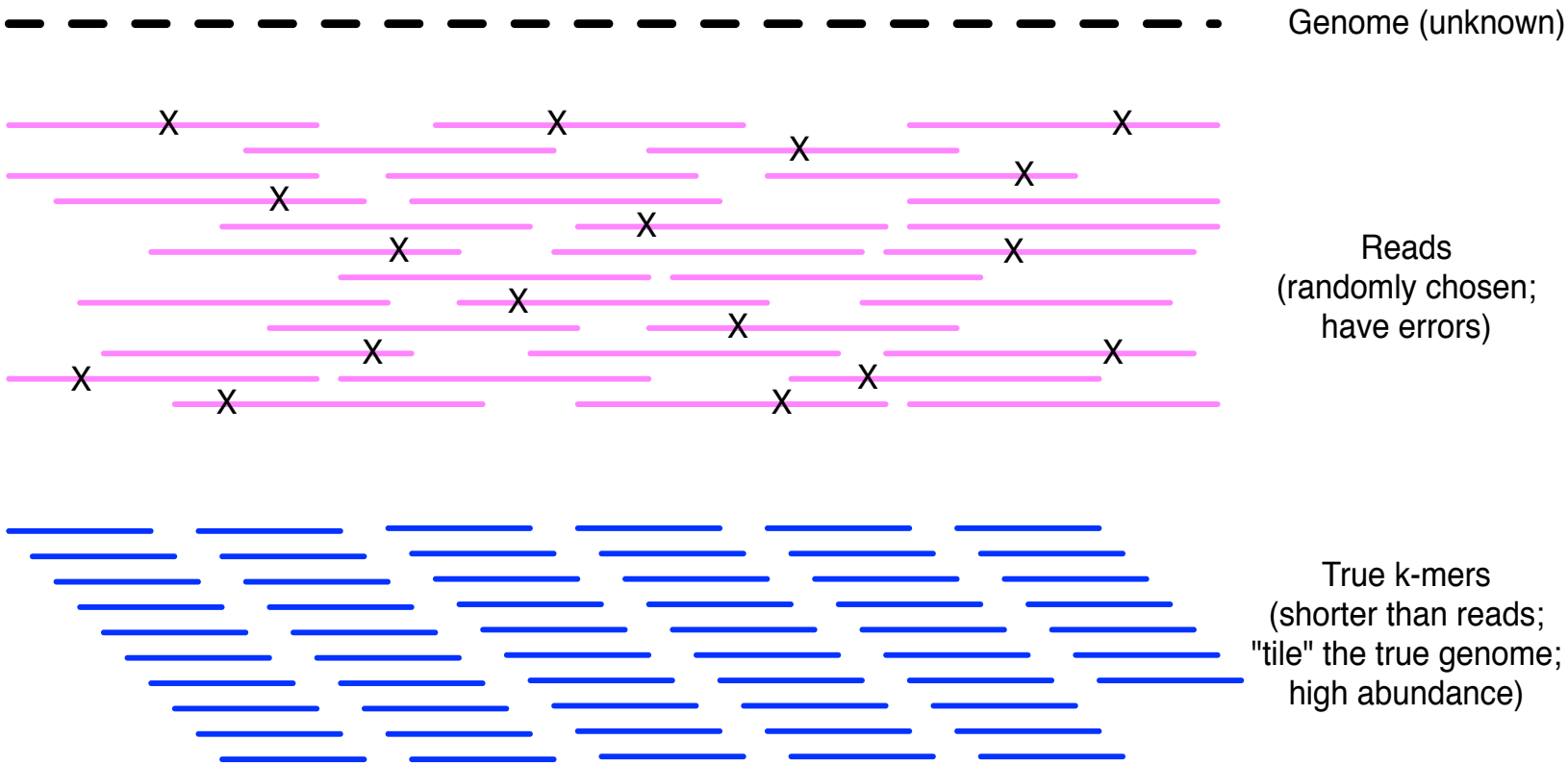
Shotgun sequencing



"Coverage" is simply the average number of reads that overlap each true base in genome.

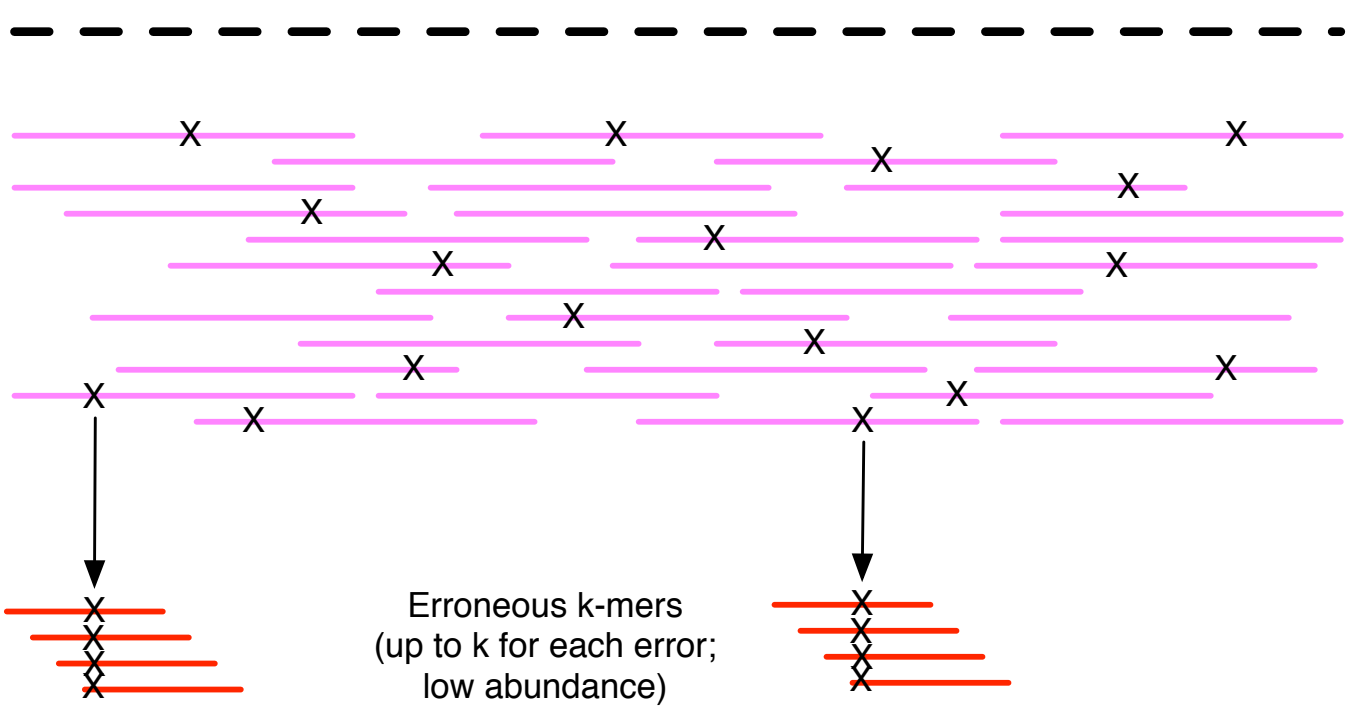
Here, the coverage is ~10 – just draw a line straight down from the top through all of the reads.

Reducing to k-mers ⇔ overlaps



Note that k-mer abundance is not properly represented here! Each blue k-mer will be present around 10 times.

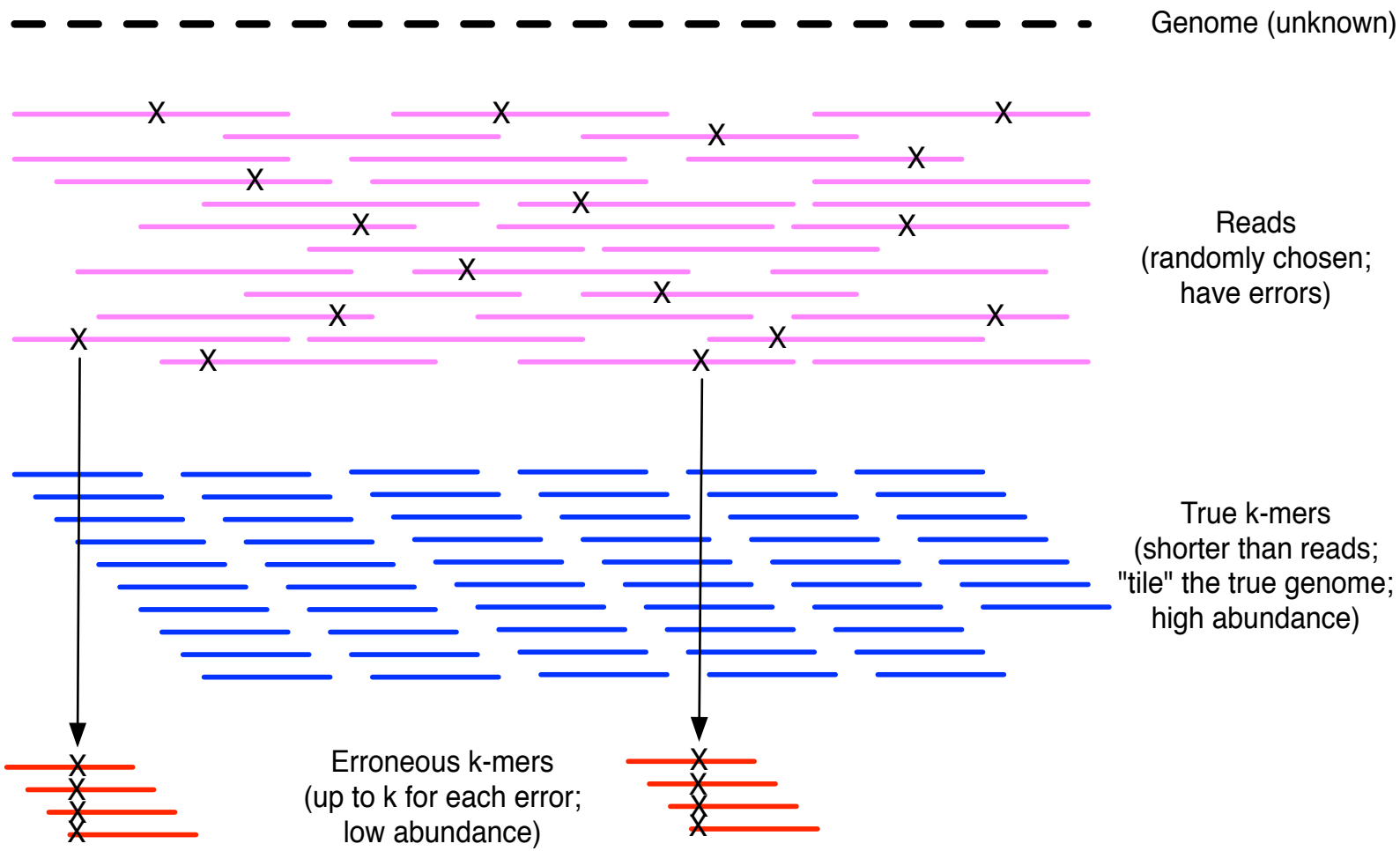
Errors create *new* k-mers



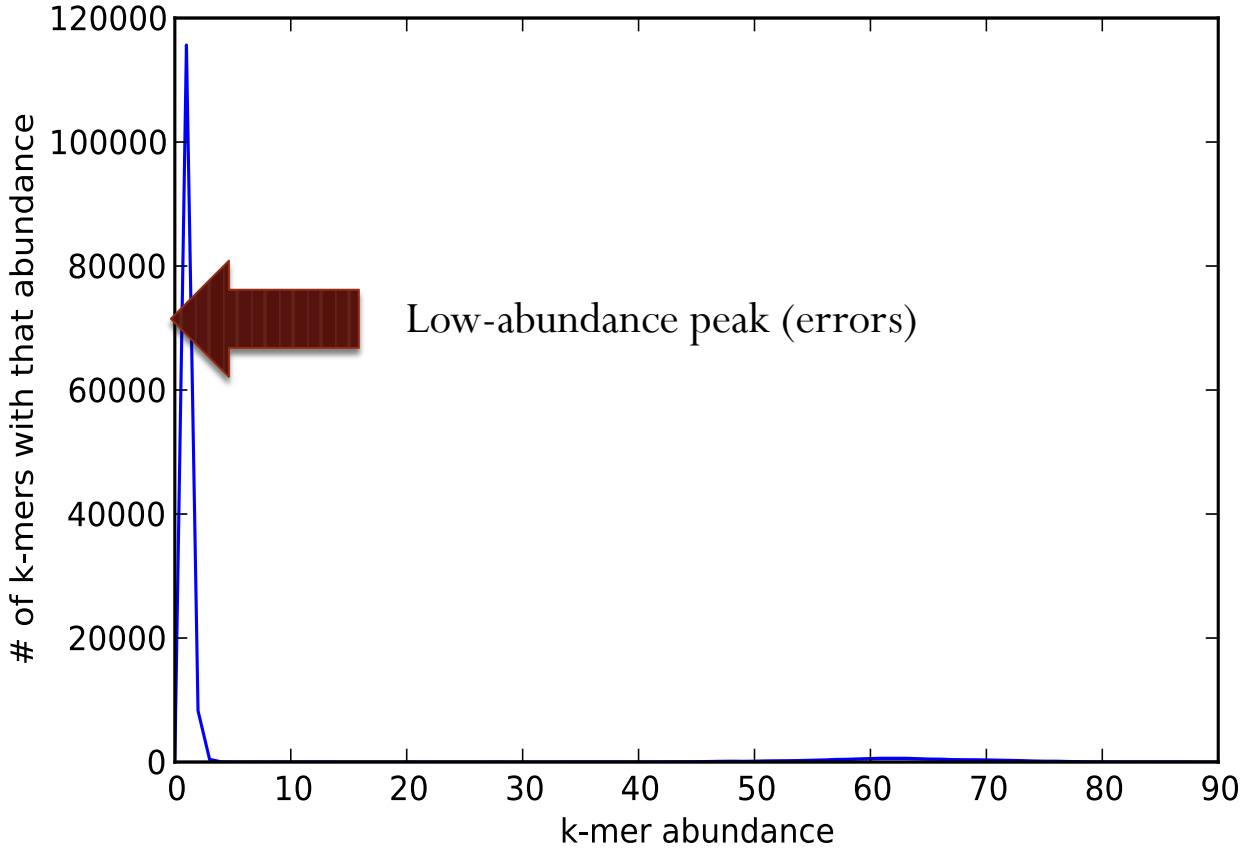
Each single base error generates ~k new k-mers.

Generally, erroneous k-mers show up only once – errors are *random*.

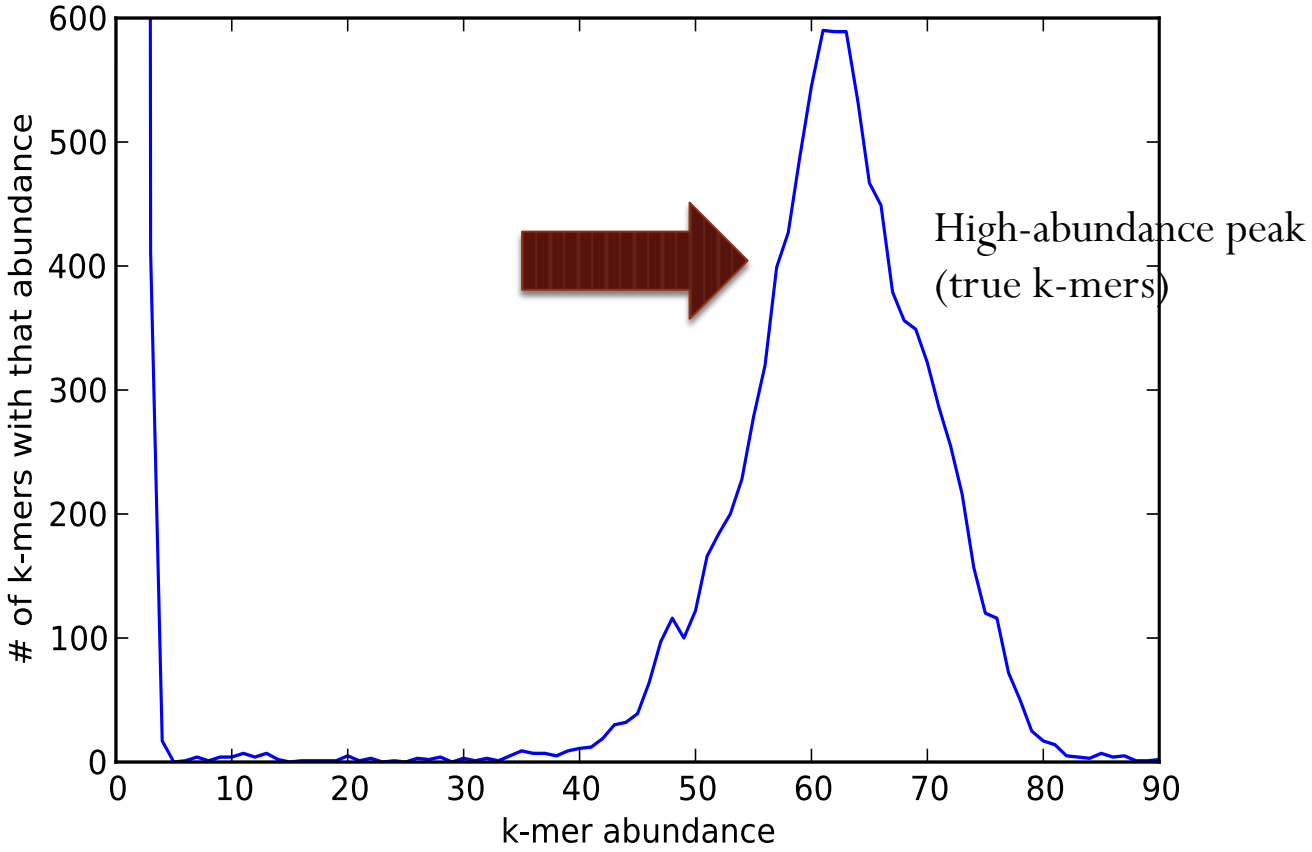
So, k-mer abundance plots are mixtures of true and false k-mers.



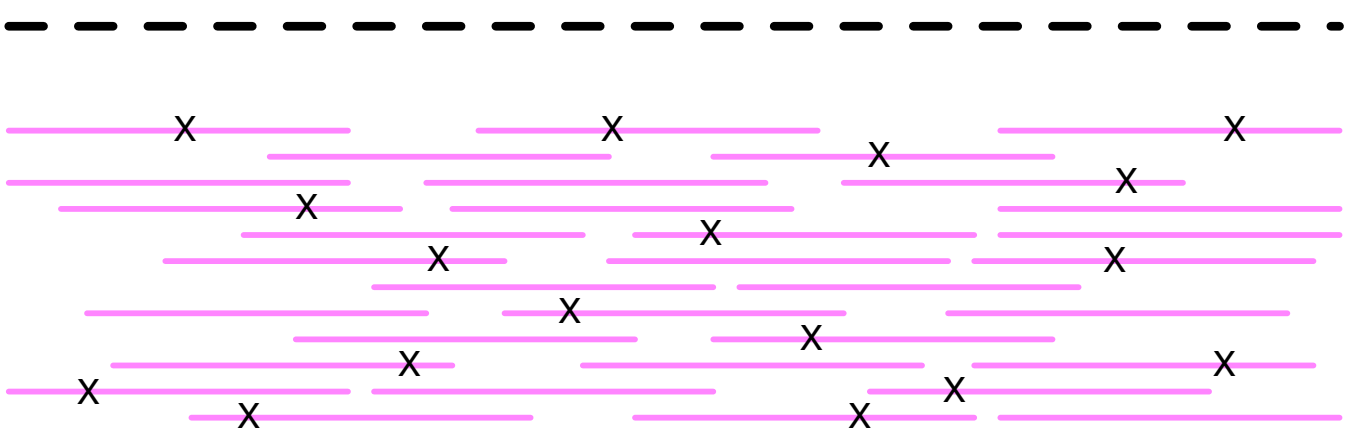
Counting k-mers - histograms



Counting k-mers - histograms



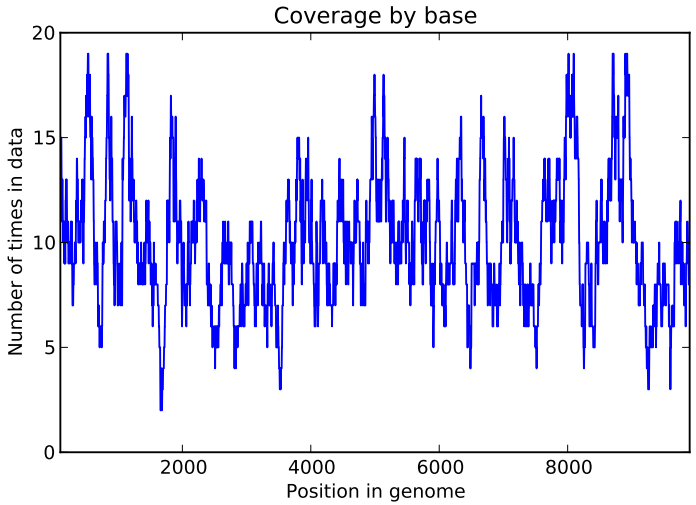
Shotgun sequencing and coverage

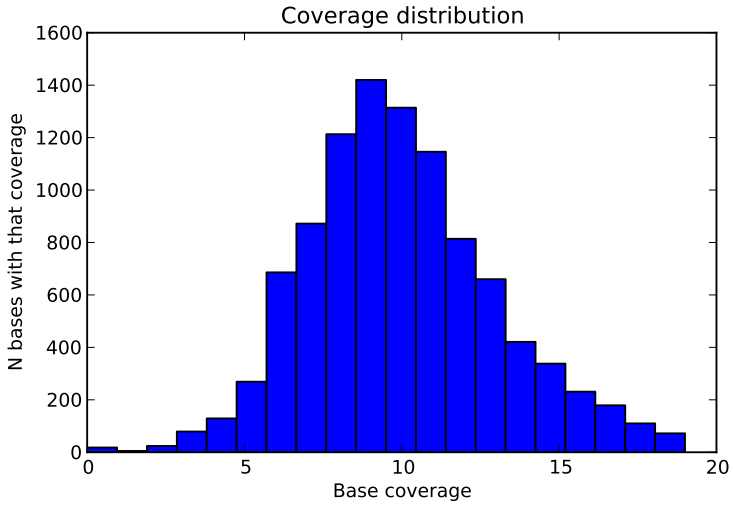


"Coverage" is simply the average number of reads that overlap each true base in genome.

Here, the coverage is ~10 – just draw a line straight down from the top through all of the reads.

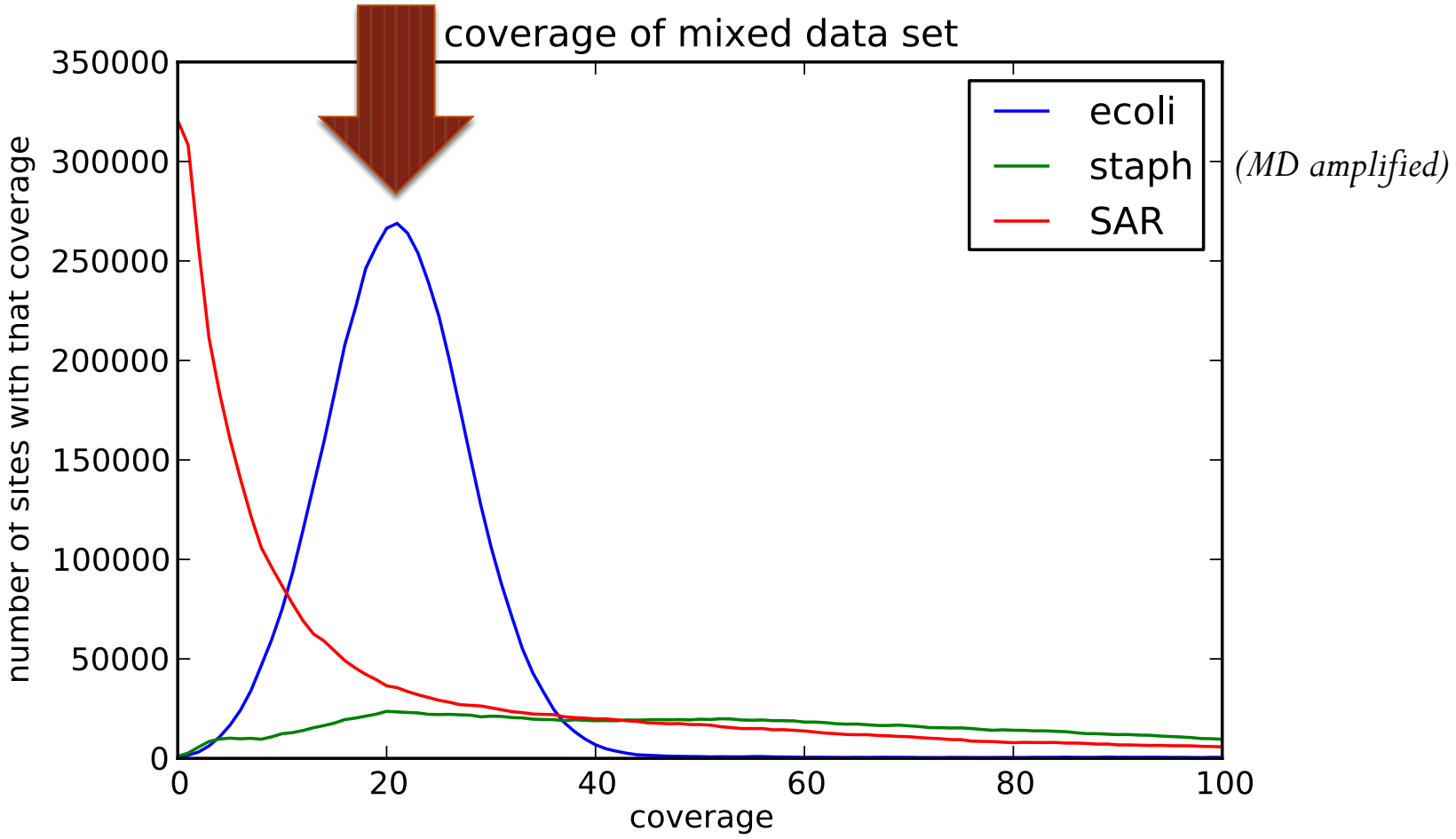
Random sampling => deep sampling needed





Typically 10-100x needed for robust recovery (300 Gbp for human)

Various experimental treatments can also modify coverage distribution.



Non-normal coverage distributions lead to decreased assembly sensitivity

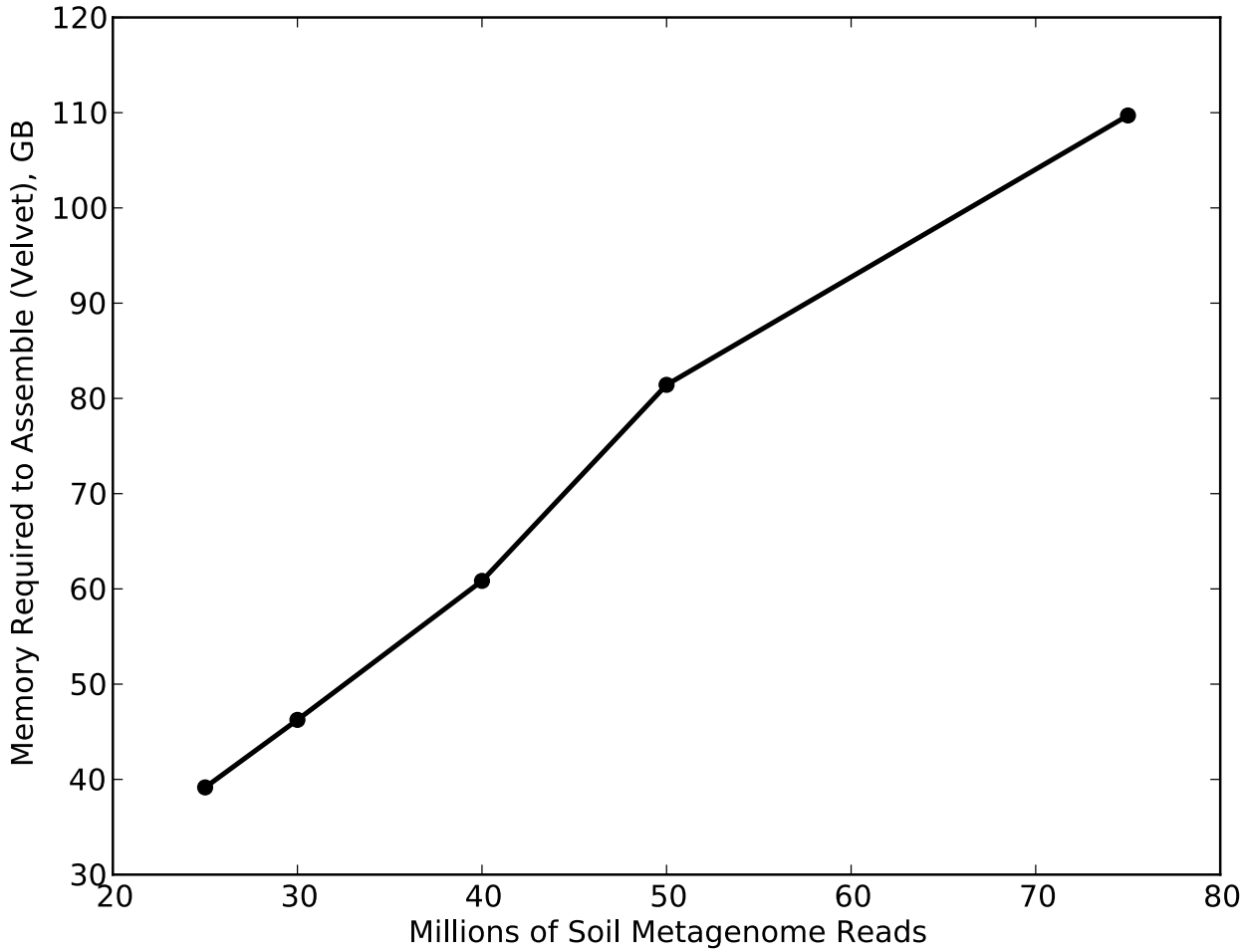
- Many assemblers embed a "coverage model" in their approach.
  - Genome assemblers: abnormally low coverage is erroneous; abnormally high coverage is repetitive sequence.
  - Transcriptome assemblers: isoforms should have same coverage across the entire isoform.
  - Metagenome assemblers: differing abundances indicate different strains.

- Is there a different way? (Yes.)

Memory requirements (Velvet/Oases – est)

• Bacterial genome (colony)	• 1-2 GB
• Human genome	• 500-1000 GB
• Vertebrate mRNA	• 100 GB +
• Low complexity metagenome	• 100 GB
• High complexity metagenome	• 1000 GB ++

Practical memory measurements



K-mer based assemblers scale poorly

Why do big data sets require big machines??

Memory usage ~ "real" variation + number of errors

Number of errors ~ size of data set

GCGTCAGGTAG**C**AGACCACCGCCATGGCGACGATG

GCGTCAGGTAGGAGACCACCG**T**CATGGCGACGATG

GCGT**T**AGGTAGGAGACCACCGCCATGGCGACGATG

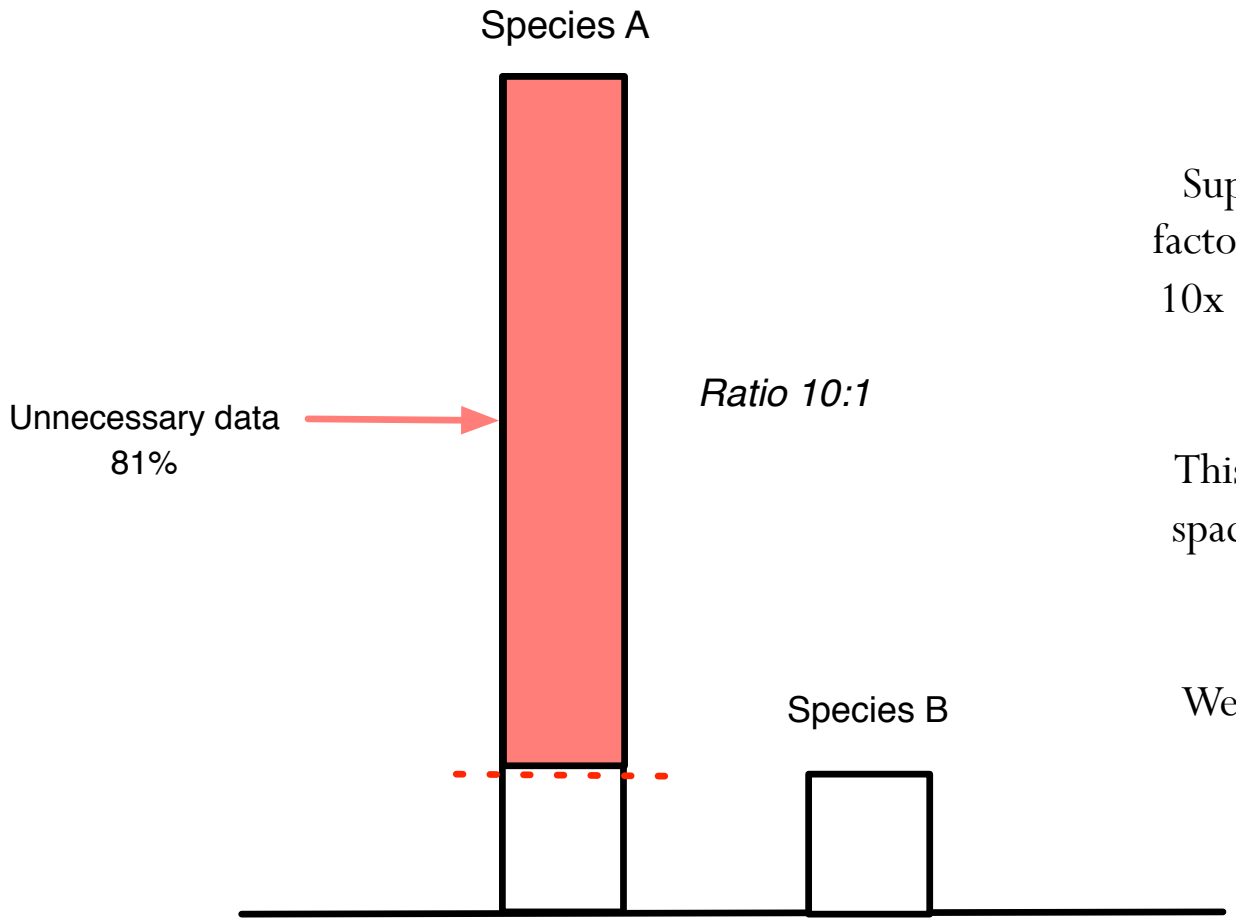
GCGTCAGGTAGGAGACC**G**CCGCCATGGCGACGATG

Why does efficiency matter?

- It is now cheaper to generate sequence than it is to analyze it computationally!
  - Machine time
  - (Wo)man power/time
- More efficient programs allow better exploration of analysis parameters for maximizing sensitivity.
- Better or more sensitive bioinformatic approaches can be developed on top of more efficient theory.

Approach: 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!!

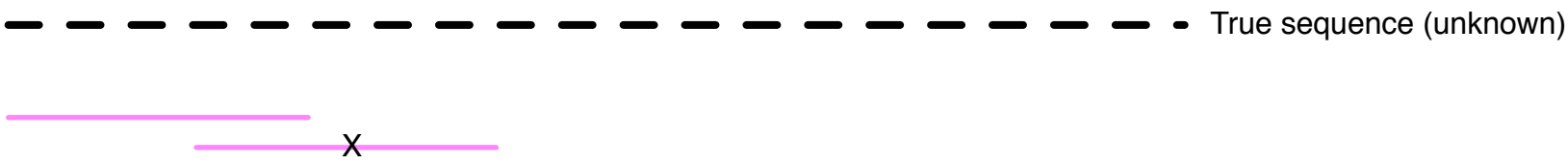
This 100x will consume disk space and, because of errors, **memory**.

We can discard it for you…

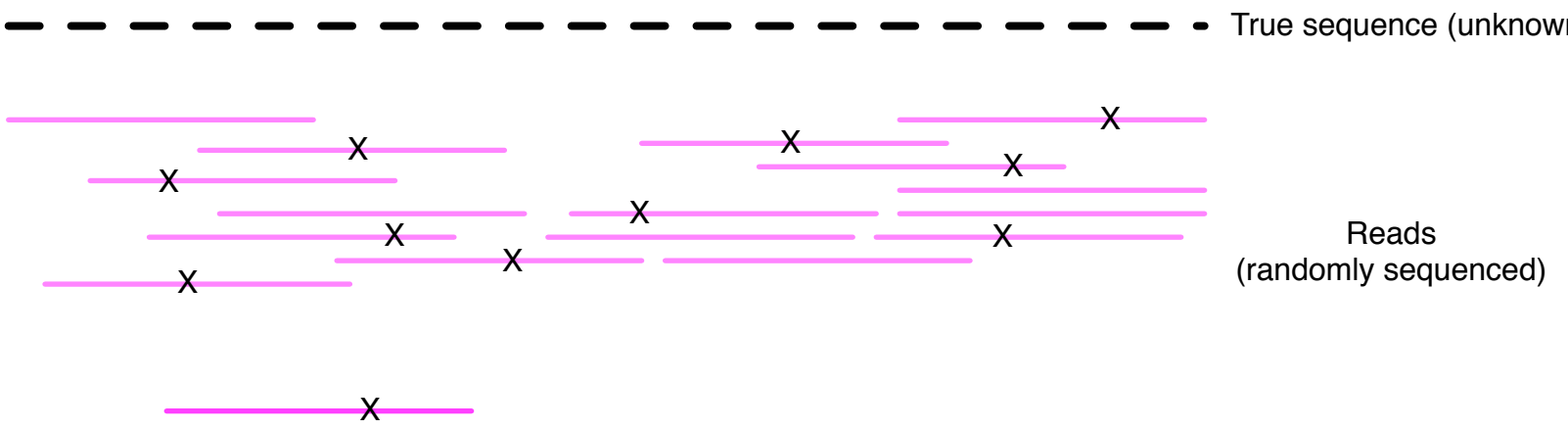
Digital normalization

Reads
(randomly sequenced)

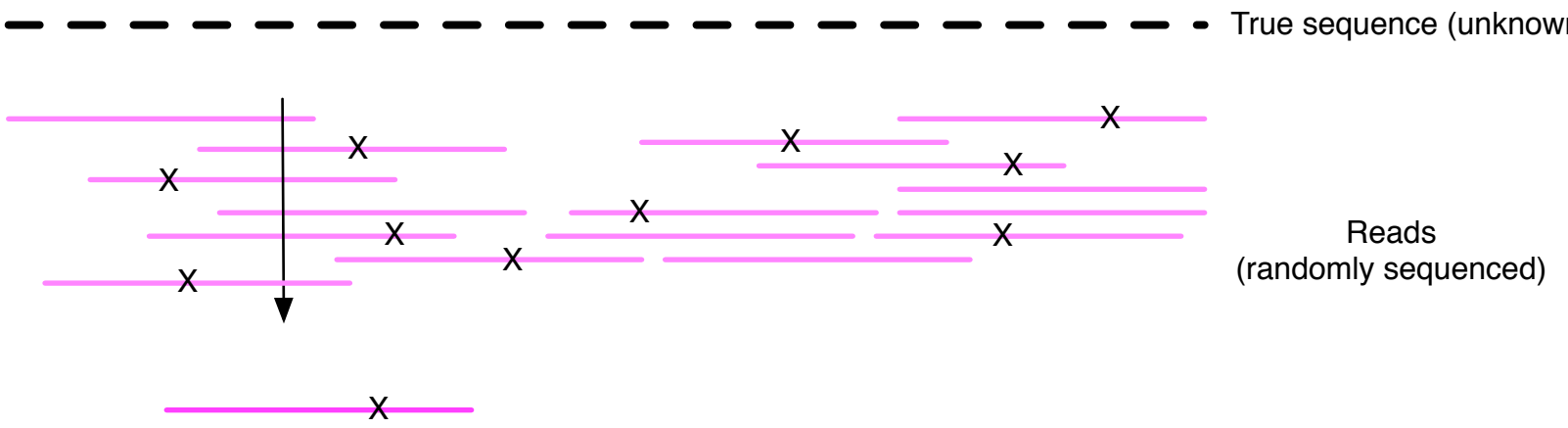
Digital normalization



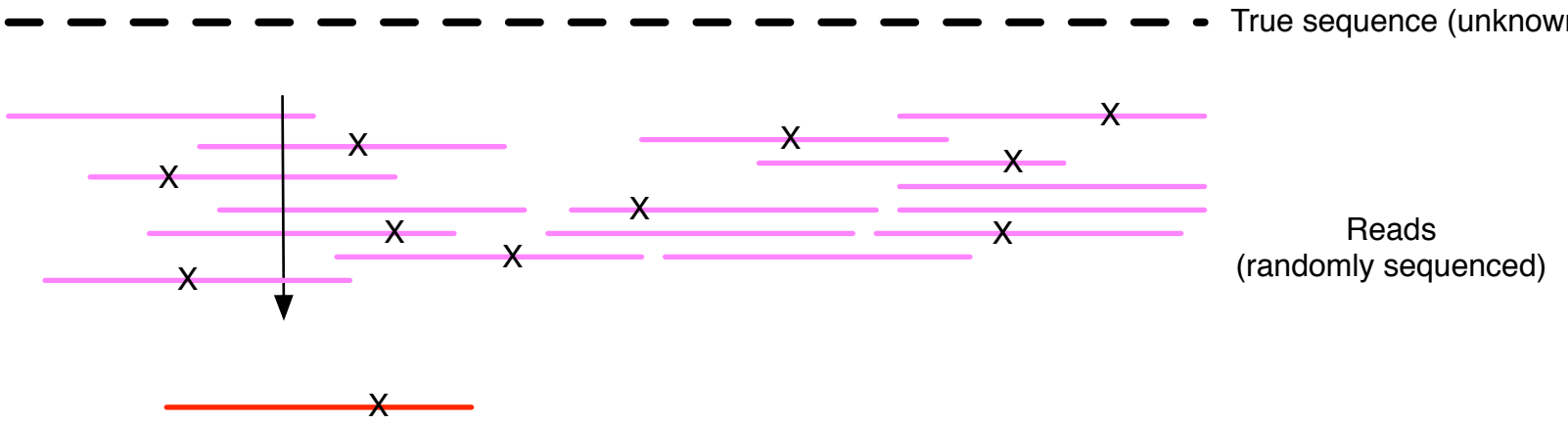
Digital normalization



Digital normalization

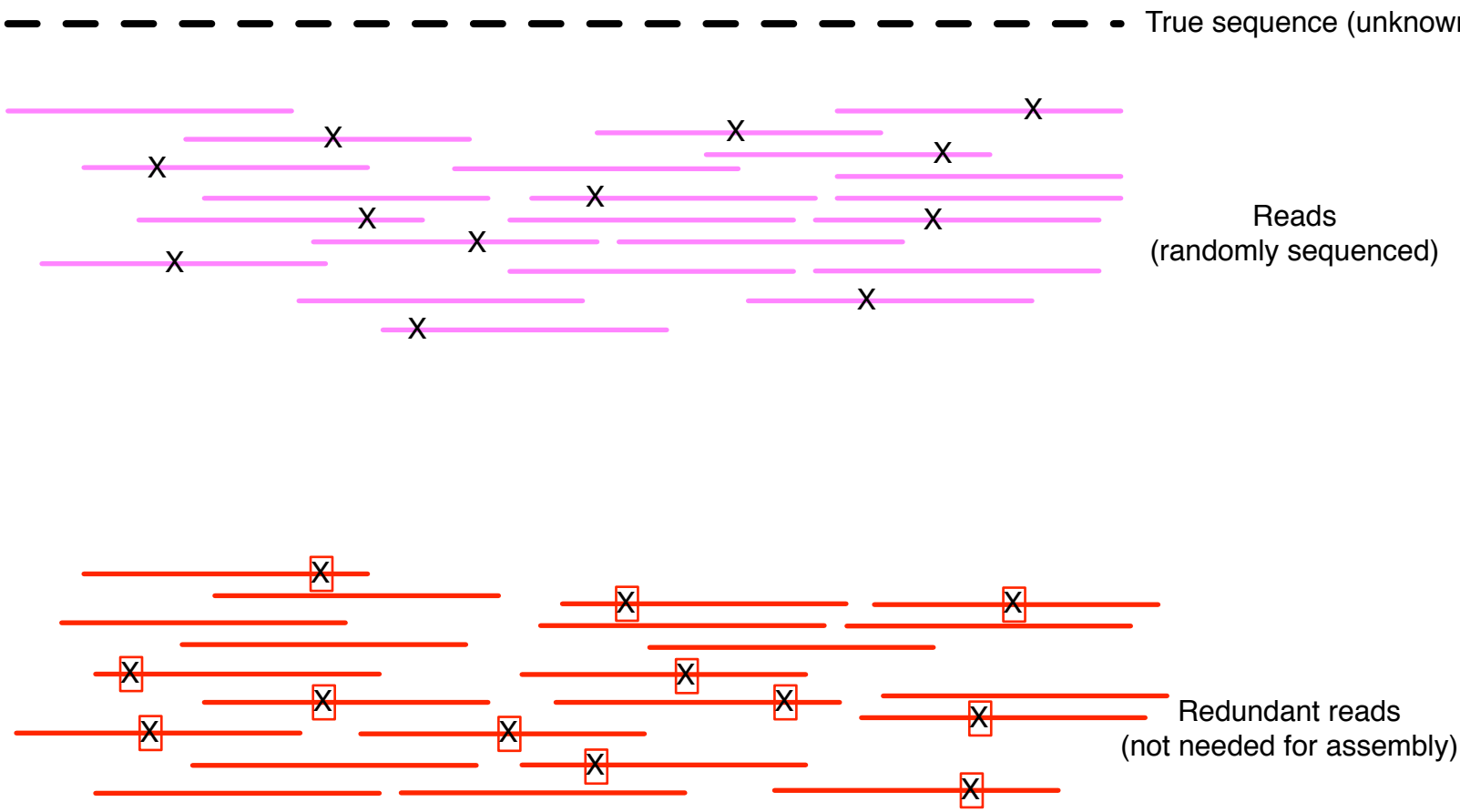


Digital normalization



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

Digital normalization

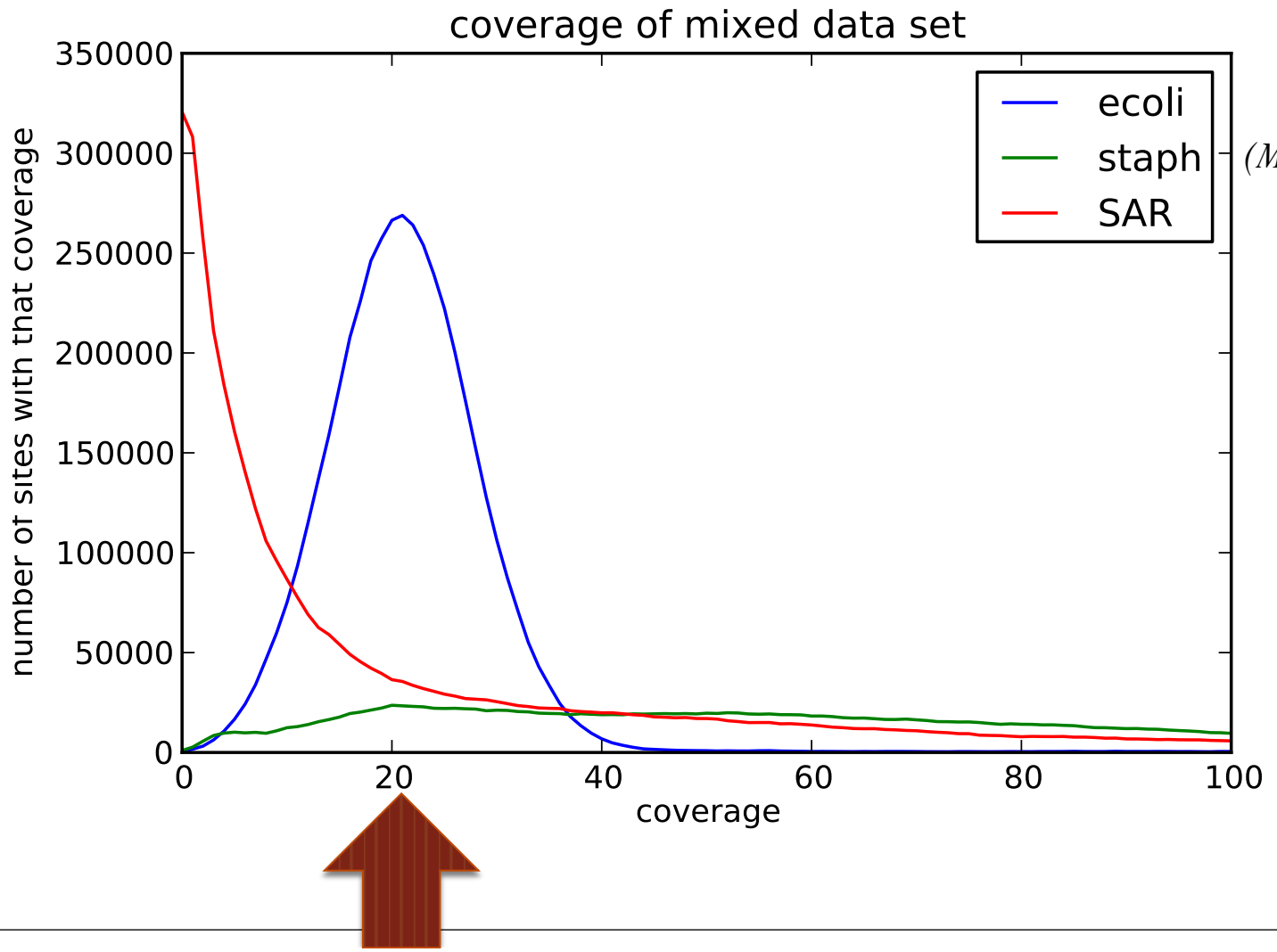


Digital normalization approach

A *digital* analog to cDNA library normalization, diginorm:

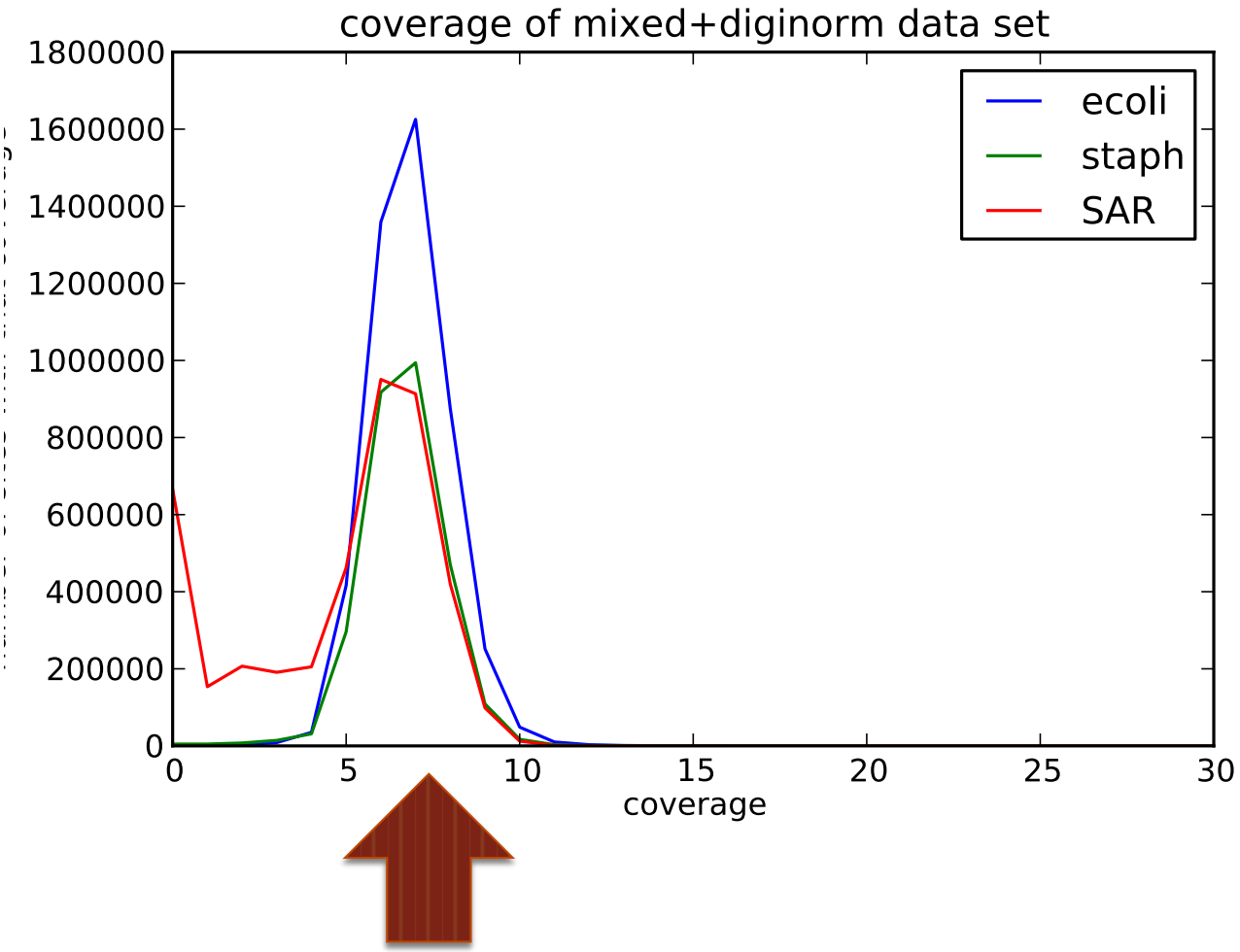
- Is single pass: looks at each read only once;
- Does not "collect" the majority of errors;
- Keeps all low-coverage reads;
- Smooths out coverage of regions.

Coverage before digital normalization:



(MD amplified)

Coverage after digital normalization:



Normalizes coverage

Discards redundancy

Eliminates majority of errors

Scales assembly dramatically.

Assembly is 98% identical.

Digital normalization approach

A *digital* analog to cDNA library normalization, diginorm is a read prefiltering approach that:

- Is single pass: looks at each read only once;
- Does not "collect" the majority of errors;
- Keeps all low-coverage reads;
- Smooths out coverage of regions.

Contig assembly is significantly more efficient and now scales with underlying genome size

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
E. coli	31m / 0.6m	1040s / 63s (16.5x)	11.2gb / 0.5 gb (22.4x)
S. aureus single-cell	58m / 0.3m	5352s / 35s (153x)	54.4gb / 0.4gb (136x)
Deltaproteobacteria single-cell	67m / 0.4m	4749s / 26s (182.7x)	52.7gb / 0.4gb (131.8x)

- Transcriptomes, microbial genomes incl MDA, and most metagenomes can be assembled in under 50 GB of RAM, with identical or *improved* results.

Digital normalization retains information, while discarding data and errors

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%
E. coli 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%

Table 2. Three-pass digital normalization removes most erroneous k-mers. Numbers in parentheses indicate number of true k-mers lost at each step, based on known reference.

Data set	True 20-mers	20-mers in reads	20-mers remaining	% reads kept
Simulated genome	399,981	8,162,813	453,588 (-4)	5%
Simulated mRNAseq	48,100	2,466,638 (-88)	182,855 (-351)	1.2%
E. coli genome	4,542,150	175,627,381 (-152)	7,638,175 (-23)	2.1%
Yeast mRNAseq	10,631,882	224,847,659 (-683)	10,532,451 (-99,436)	2.1%
Mouse mRNAseq	43,830,642	709,662,624 (-23,196)	42,350,127 (-1,488,380)	7.1%

Lossy compression

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

Lossy compression

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

Lossy compression

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

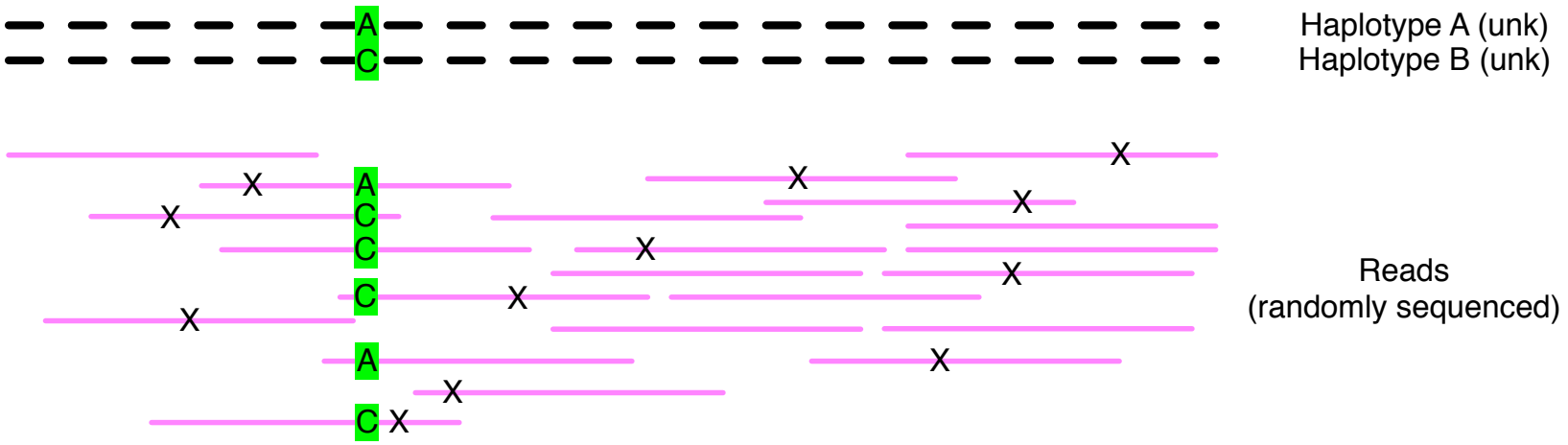
Lossy compression

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

Lossy compression

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

Can we apply this algorithmically efficient technique to variants? Yes.



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

Coverage is adjusted to retain signal