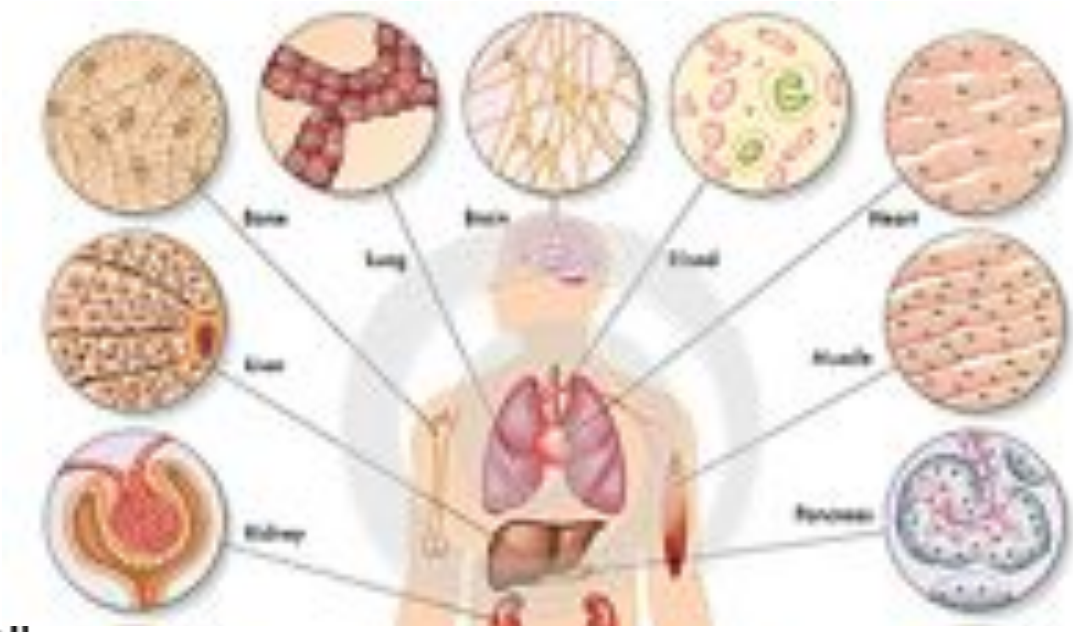
Searching for GATTACA

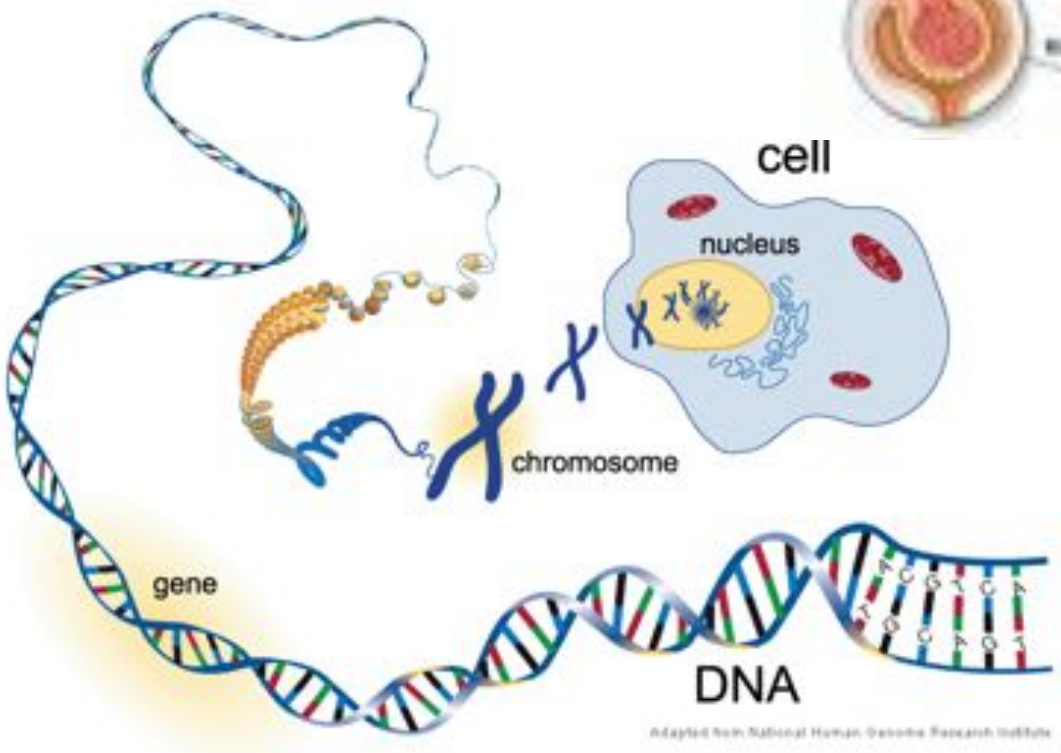
Michael Schatz

Bioinformatics Lecture 1
Undergraduate Research Program 2014

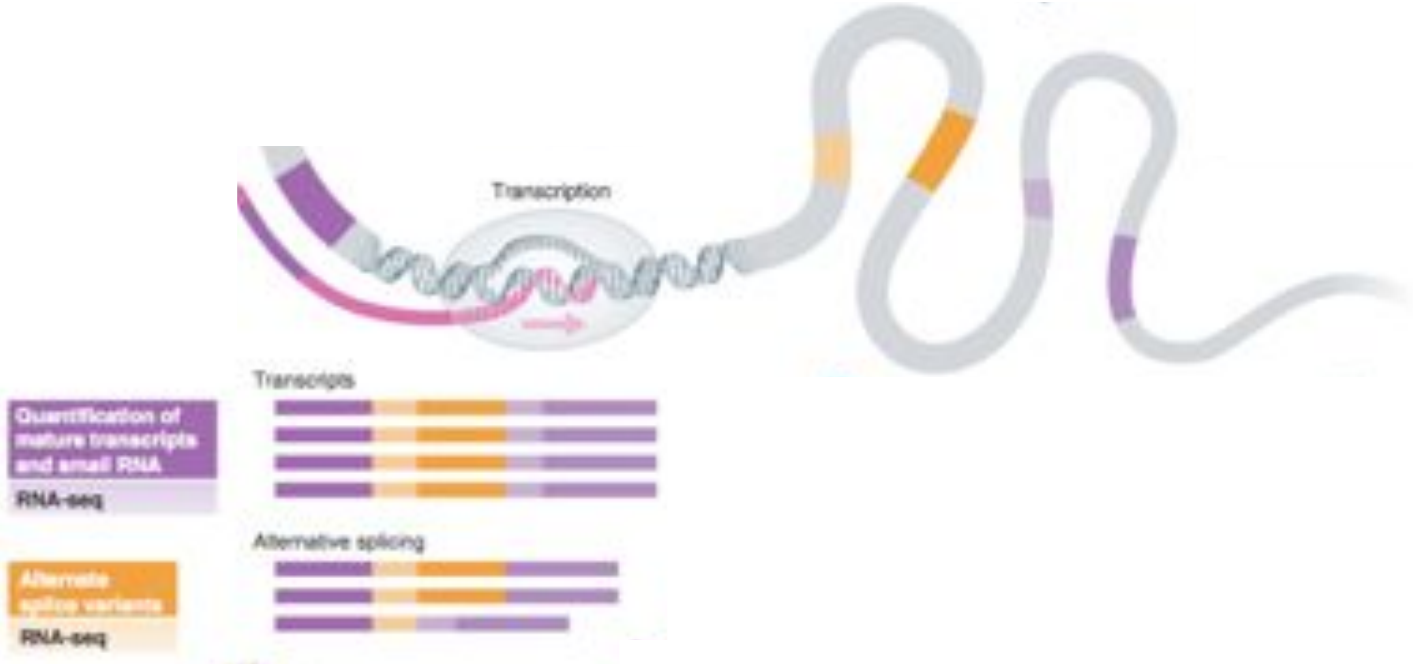
Cells & DNA

Each cell of your body contains an exact copy of your 3 billion base pair genome.





Your specific nucleotide sequence encodes the genetic program for your cells and ultimately your traits



Soon et al., Molecular Systems Biology, 2013

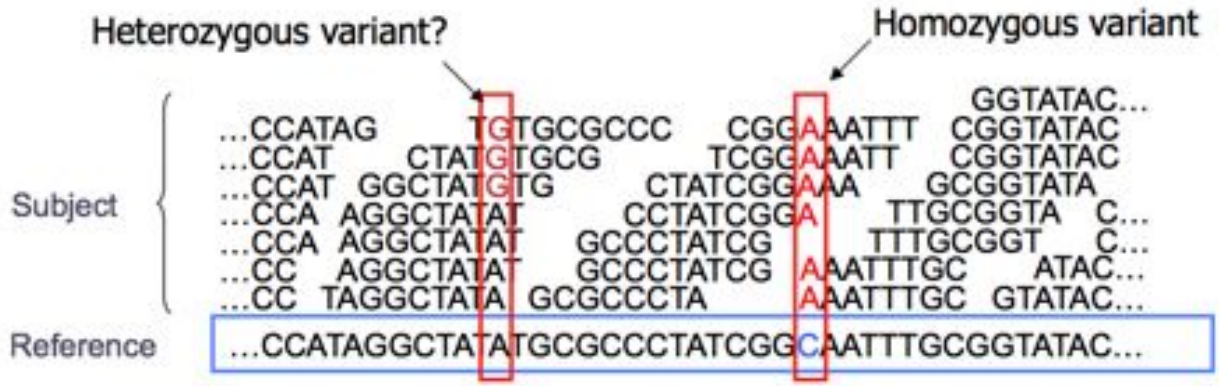
Sequencing Assays

The *Seq List (in chronological order)

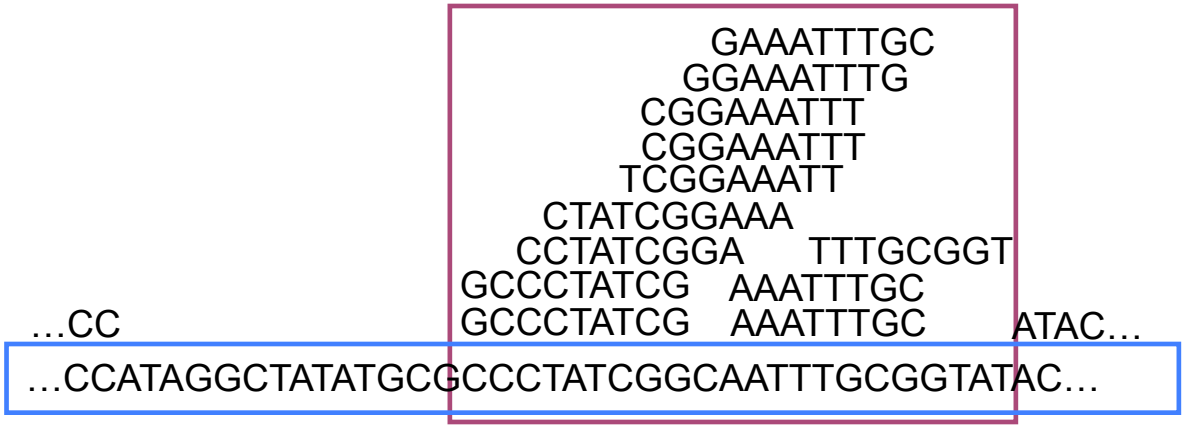
1. Gregory E. Crawford et al., "Genome-wide Mapping of DNase Hypersensitive Sites Using Massively Parallel Signature Sequencing (MPSS)," Genome Research 16, no. 1 (January 1, 2006): 123–131, doi:10.1101/gr.4074106.
2. David S. Johnson et al., "Genome-Wide Mapping of in Vivo Protein-DNA Interactions," Science 316, no. 5830 (June 8, 2007): 1497–1502, doi:10.1126/science.1141319.
3. Tarjei S. Mikkelsen et al., "Genome-wide Maps of Chromatin State in Pluripotent and Lineage-committed Cells," Nature 448, no. 7153 (August 2, 2007): 553–560, doi:10.1038/nature06008.
4. Thomas A. Down et al., "A Bayesian Deconvolution Strategy for Immunoprecipitation-based DNA Methylome Analysis," Nature Biotechnology 26, no. 7 (July 2008): 779–785, doi:10.1038/nbt1414.
5. Ali Mortazavi et al., "Mapping and Quantifying Mammalian Transcriptomes by RNA-Seq," Nature Methods 5, no. 7 (July 2008): 621–628, doi:10.1038/nmeth.1226.
6. Nathan A. Baird et al., "Rapid SNP Discovery and Genetic Mapping Using Sequenced RAD Markers," PLoS ONE 3, no. 10 (October 13, 2008): e3376, doi:10.1371/journal.pone.0003376.
7. Leighton J. Core, Joshua J. Waterfall, and John T. Lis, "Nascent RNA Sequencing Reveals Widespread Pausing and Divergent Initiation at Human Promoters," Science 322, no. 5909 (December 19, 2008): 1845–1848, doi:10.1126/science.1162228.
8. Chao Xie and Martti T. Tammi, "CNV-seq, a New Method to Detect Copy Number Variation Using High-throughput Sequencing," BMC Bioinformatics 10, no. 1 (March 6, 2009): 80, doi:10.1186/1471-2105-10-80.
9. Jay R. Hesselberth et al., "Global Mapping of protein-DNA Interactions in Vivo by Digital Genomic Footprinting," Nature Methods 6, no. 4 (April 2009): 283–289, doi:10.1038/nmeth.1313.
10. Nicholas T. Ingolia et al., "Genome-Wide Analysis in Vivo of Translation with Nucleotide Resolution Using Ribosome Profiling," Science 324, no. 5924 (April 10, 2009): 218–223, doi:10.1126/science.1168978.
11. Alayne L. Brunner et al., "Distinct DNA Methylation Patterns Characterize Differentiated Human Embryonic Stem Cells and Developing Human Fetal Liver," Genome Research 19, no. 6 (June 1, 2009): 1044–1056, doi:10.1101/gr.088773.108.
12. Mayumi Oda et al., "High-resolution Genome-wide Cytosine Methylation Profiling with Simultaneous Copy Number Analysis and Optimization for Limited Cell Numbers," Nucleic Acids Research 37, no. 12 (July 1, 2009): 3829–3839, doi:10.1093/nar/gkp260.
13. Zachary D. Smith et al., "High-throughput Bisulfite Sequencing in Mammalian Genomes," Methods 48, no. 3 (July 2009): 226–232, doi:10.1016/j.ymeth.2009.05.003.
14. Andrew M. Smith et al., "Quantitative Phenotyping via Deep Barcode Sequencing," Genome Research (July 21, 2009), doi:10.1101/gr.

Short Read Applications

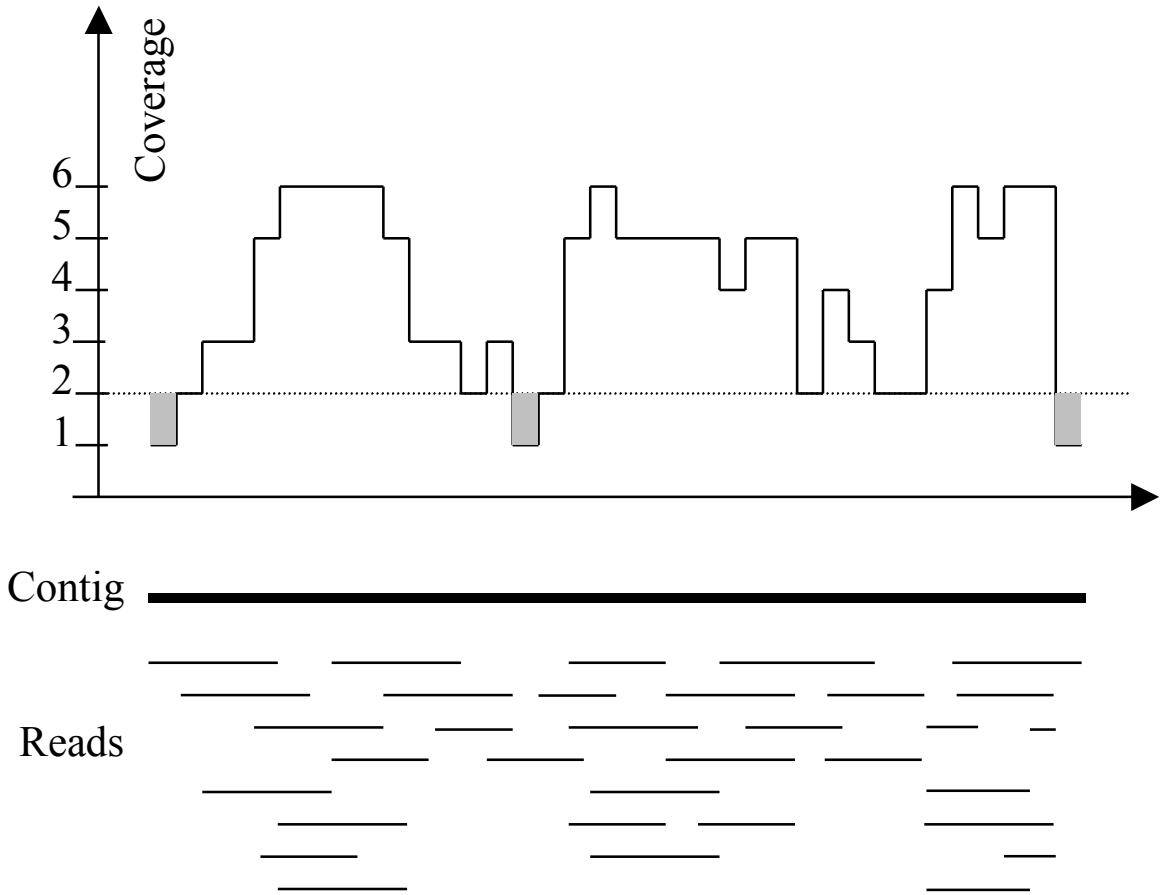
- Genotyping: Identify Variations



- *-seq: Classify & measure significant peaks

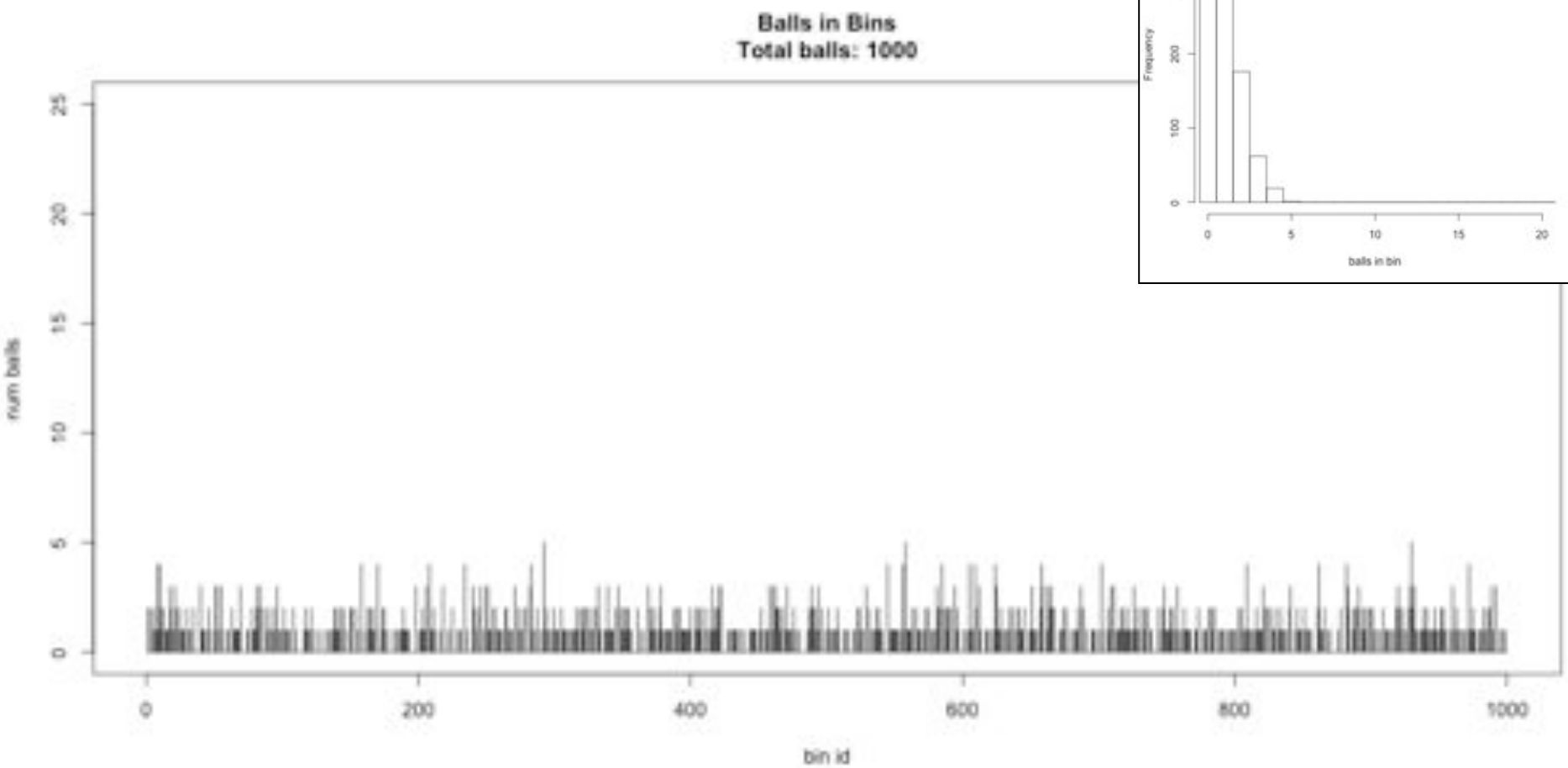


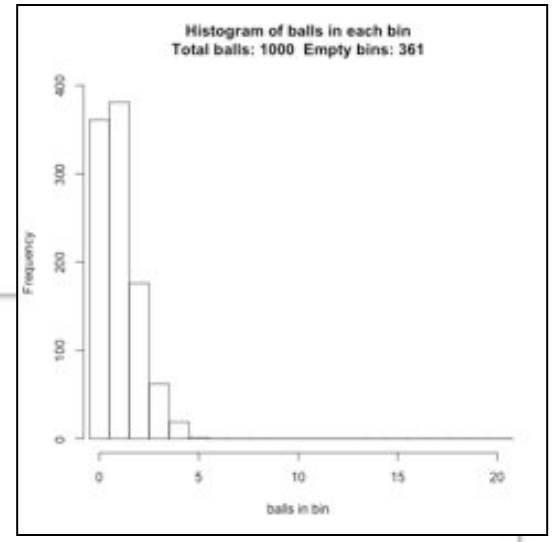
Typical sequencing coverage



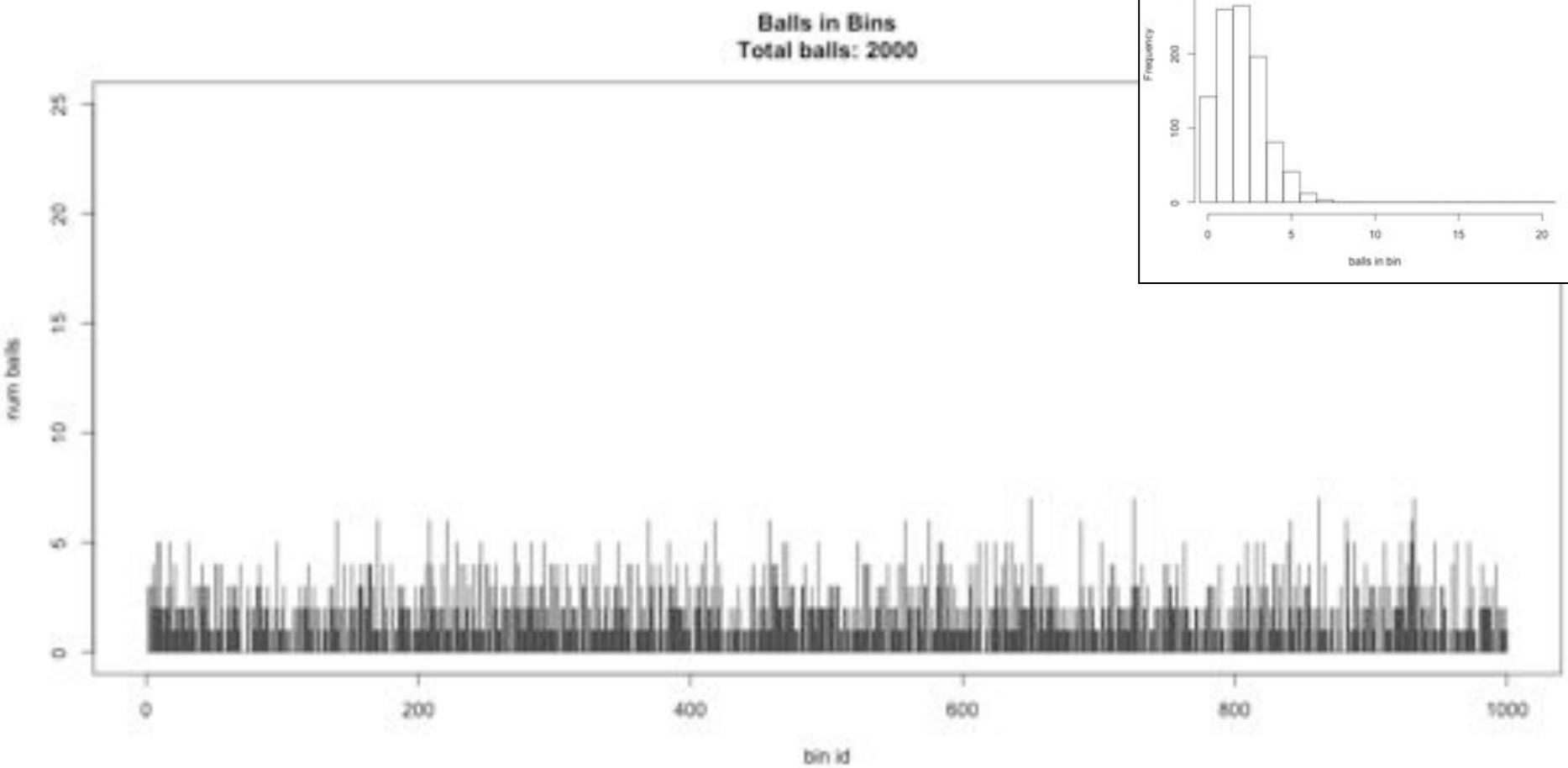
Imagine raindrops on a sidewalk

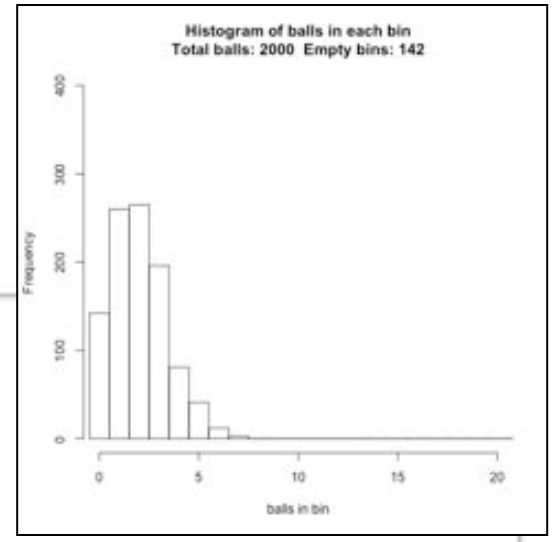
1x sequencing



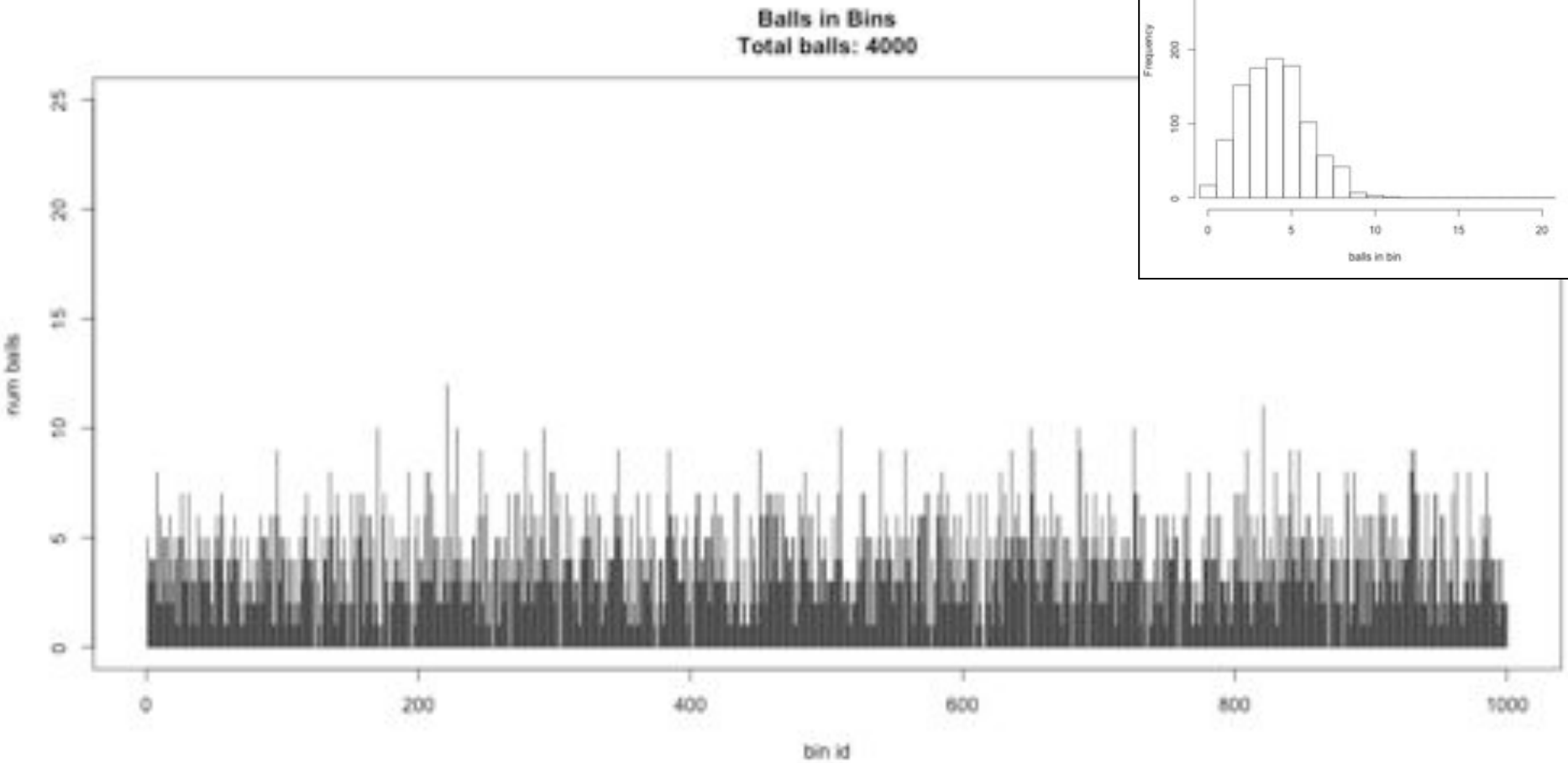


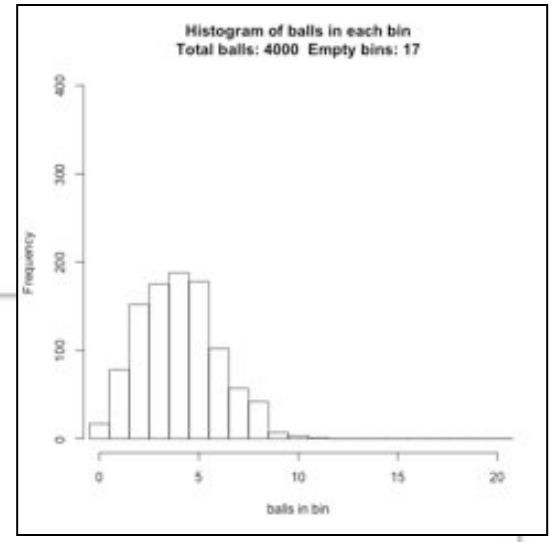
2x sequencing



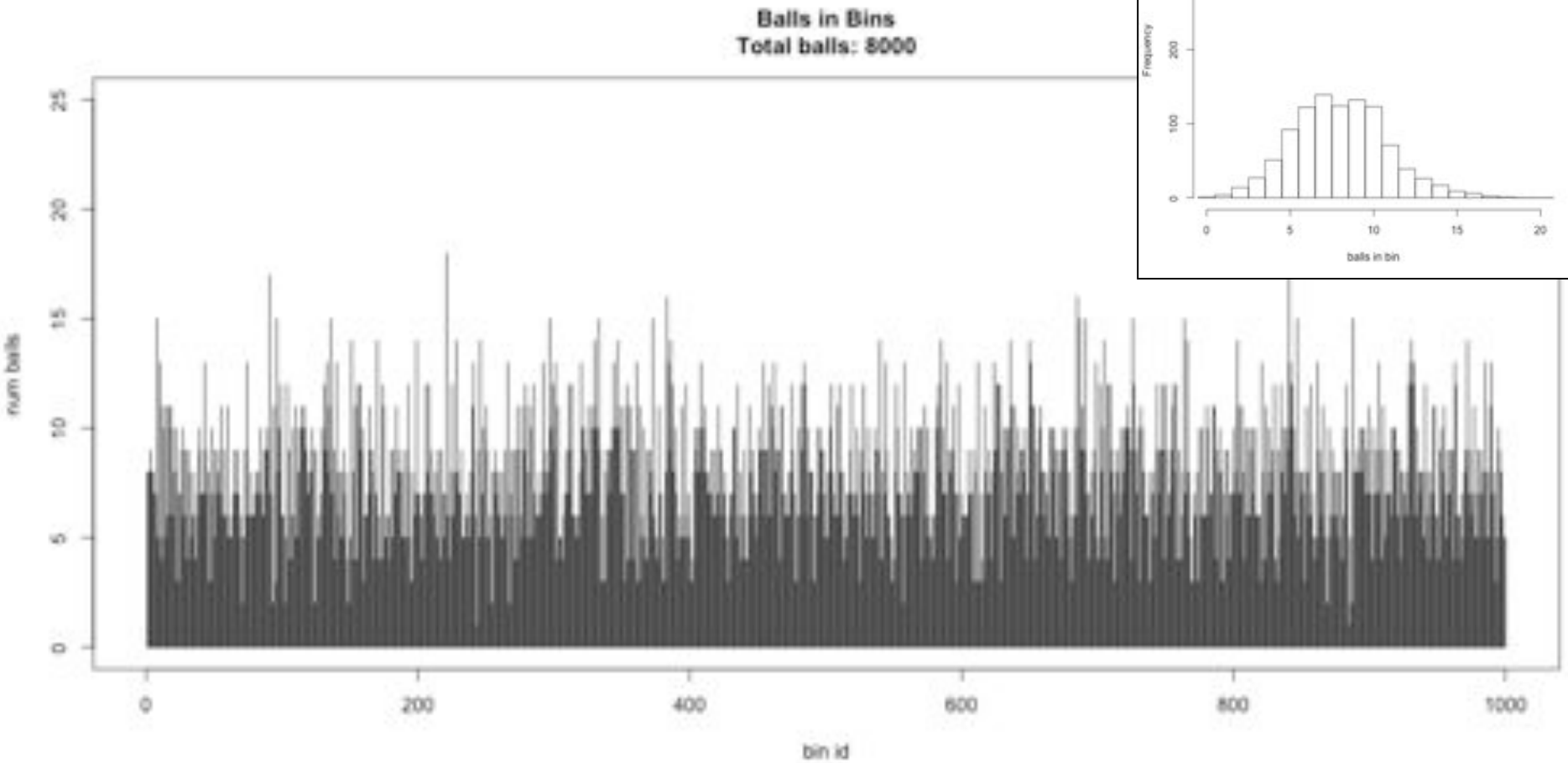


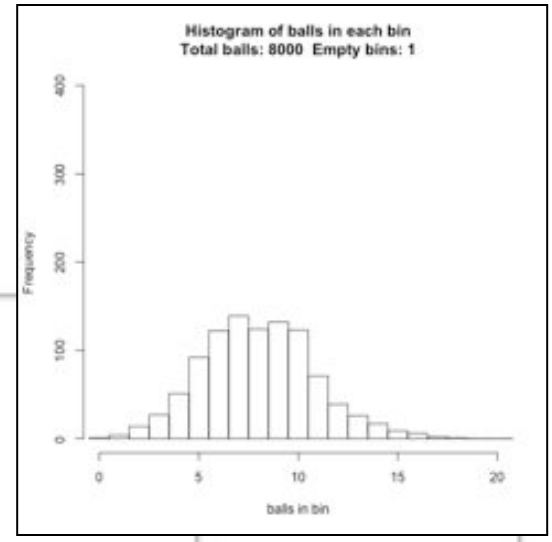
4x sequencing





8x sequencing





Poisson Distribution

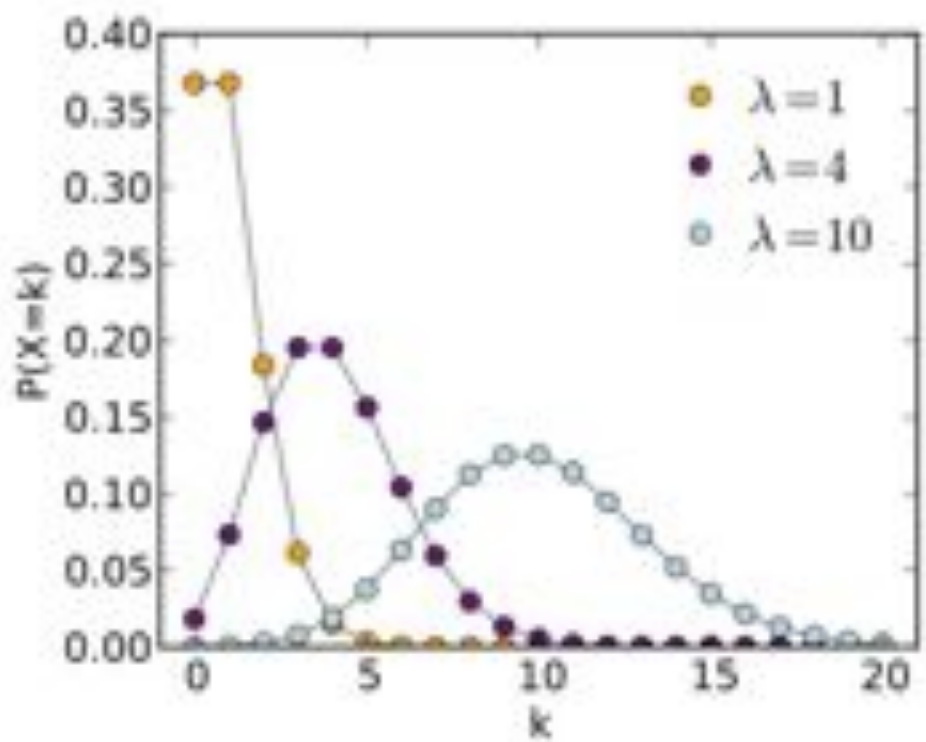
The probability of a given number of events occurring in a fixed interval of time and/or space if these events occur with a known average rate and independently of the time since the last event.

Formulation comes from the limit of the binomial equation

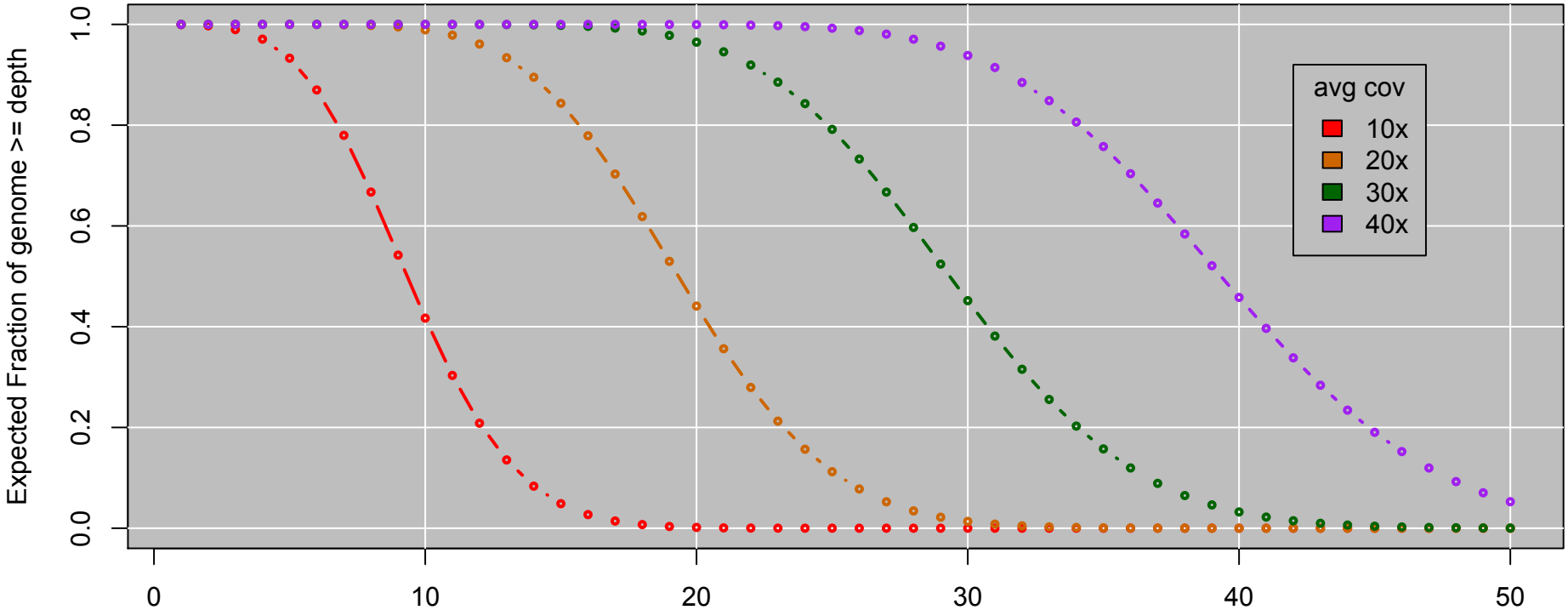
Resembles a normal distribution, but over the positive values, and with only a single parameter.

Key property: The standard deviation is the square root of the mean.

$$P(k) = \frac{\lambda^k}{k!} e^{-\lambda}$$



Genome Coverage Distribution



Expect Poisson distribution on depth

- Standard Deviation = sqrt(cov)

This is the mathematically model => reality may be much worse

- Double your coverage for diploid genomes
- Can use somewhat lower coverage in a population to find common variants

Searching for GATTACA

- Where is GATTACA in the human genome?
- Strategy 1: Brute Force

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	…
T	G	A	T	T	A	C	A	G	A	T	T	A	C	C	…
G	A	T	T	A	C	A									

No match at offset 1

Searching for GATTACA

- Where is GATTACA in the human genome?
- Strategy 1: Brute Force

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	…
T	G	A	T	T	A	C	A	G	A	T	T	A	C	C	…
	G	A	T	T	A	C	A								

Match at offset 2

Searching for GATTACA

- Where is GATTACA in the human genome?
- Strategy 1: Brute Force

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	…
T	G	A	T	T	A	C	A	G	A	T	T	A	C	C	…
		G	A	T	T	A	C	A	…						

No match at offset 3…

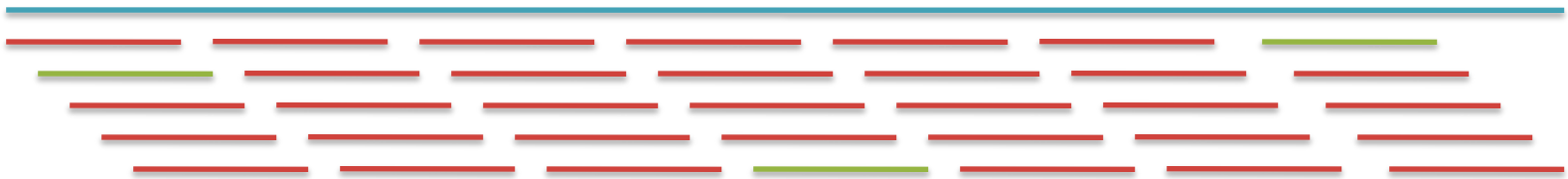
Searching for GATTACA

- Where is GATTACA in the human genome?
- Strategy 1: Brute Force

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	…
T	G	A	T	T	A	C	A	G	A	T	T	A	C	C	…
								G	A	T	T	A	C	A	

No match at offset 9 <- Checking each possible position takes time

Brute Force Analysis



- Brute Force:
  - At every possible offset in the genome:
    - Do all of the characters of the query match?

- Analysis
  - Simple, easy to understand
  - Genome length = n [3B]
  - Query length = m [7]
  - Comparisons: (n-m+1) * m [21B]

- Overall runtime: O(nm)

[How long would it take if we double the genome size, read length?]

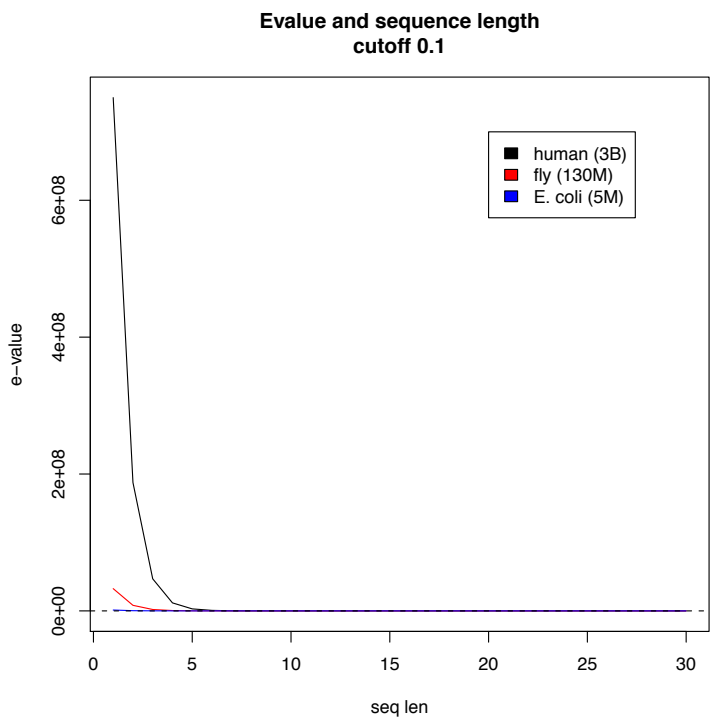
[How long would it take if we double both?]

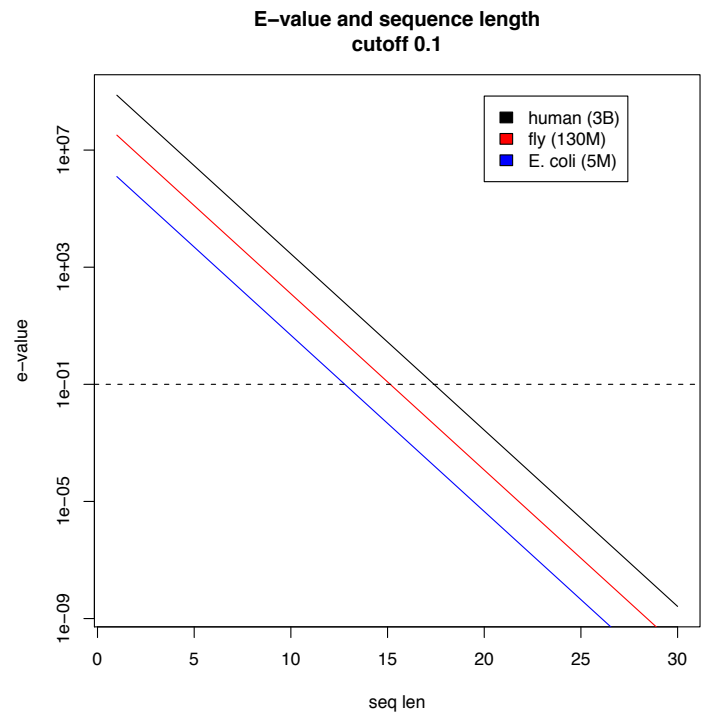
Expected Occurrences

The expected number of occurrences (e-value) of a given sequence in a genome depends on the length of the genome and inversely on the length of the sequence

- 1 in 4 bases are G, 1 in 16 positions are GA, 1 in 64 positions are GAT, …
- 1 in 16,384 should be GATTACA
- $E=n/(4^m)$ [183,105 expected occurrences]

[How long do the reads need to be for a significant match?]





Brute Force Reflections

Why check every position?

– GATTACA can't possibly start at position 15 [WHY?]

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	…
T	G	A	T	T	A	C	A	G	A	T	T	A	C	C	…
								G	A	T	T	A	C	A	

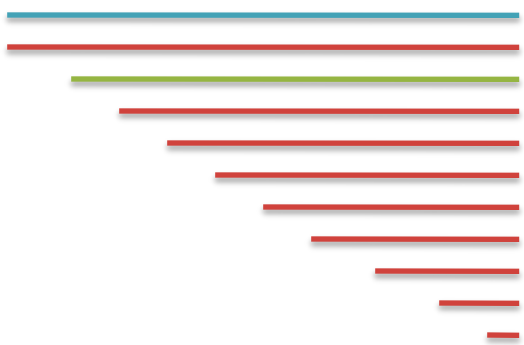
– Improve runtime to O(n + m) [3B + 7]
  - If we double both, it just takes twice as long
  - Knuth-Morris-Pratt, 1977
  - Boyer-Moyer, 1977, 1991

– For one-off scans, this is the best we can do (optimal performance)
  - We have to read every character of the genome, and every character of the query
  - For short queries, runtime is dominated by the length of the genome

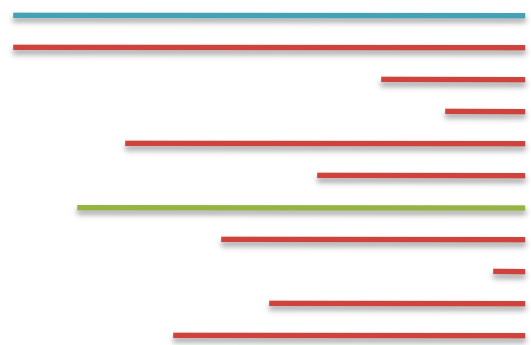
Suffix Arrays: Searching the Phone Book

- What if we need to check many queries?
  - We don't need to check every page of the phone book to find 'Schatz'
  - Sorting alphabetically lets us immediately skip 96% (25/26) of the book *without any loss in accuracy*

- Sorting the genome: Suffix Array (Manber & Myers, 1991)
  - Sort every suffix of the genome



Split into n suffixes



Sort suffixes alphabetically

[Challenge Question: How else could we split the genome?]

Searching the Index

- Strategy 2: Binary search
  - Compare to the middle, refine as higher or lower

- Searching for GATTACA
  - Lo = 1; Hi = 15;

Lo →

#	Sequence	Pos
1	ACAGATTACC…	6
2	ACC…	13
3	AGATTACC…	8
4	ATTACAGATTACC…	3
5	ATTACC…	10
6	C…	15
7	CAGATTACC…	7
8	CC…	14
9	GATTACAGATTACC…	2
10	GATTACC…	9
11	TACAGATTACC…	5
12	TACC…	12
13	TGATTACAGATTACC…	1
14	TTACAGATTACC…	4
15	TTACC…	11

Hi →

Searching the Index

- Strategy 2: Binary search
  - Compare to the middle, refine as higher or lower

- Searching for GATTACA
  - Lo = 1; Hi = 15; Mid = (1+15)/2 = 8
  - Middle = Suffix[8] = CC

Lo →

#	Sequence	Pos
1	ACAGATTACC…	6
2	ACC…	13
3	AGATTACC…	8
4	ATTACAGATTACC…	3
5	ATTACC…	10
6	C…	15
7	CAGATTACC…	7
8	CC…	14
9	GATTACAGATTACC…	2
10	GATTACC…	9
11	TACAGATTACC…	5
12	TACC…	12
13	TGATTACAGATTACC…	1
14	TTACAGATTACC…	4
15	TTACC…	11

Hi →

Searching the Index

- Strategy 2: Binary search
  - Compare to the middle, refine as higher or lower

- Searching for GATTACA
  - Lo = 1; Hi = 15; Mid = (1+15)/2 = 8
  - Middle = Suffix[8] = CC
    => Higher: Lo = Mid + 1

Lo →

#	Sequence	Pos
1	ACAGATTACC…	6
2	ACC…	13
3	AGATTACC…	8
4	ATTACAGATTACC…	3
5	ATTACC…	10
6	C…	15
7	CAGATTACC…	7
8	CC…	14
9	GATTACAGATTACC…	2
10	GATTACC…	9
11	TACAGATTACC…	5
12	TACC…	12
13	TGATTACAGATTACC…	1
14	TTACAGATTACC…	4
15	TTACC…	11

Hi →

Searching the Index

- Strategy 2: Binary search
  - Compare to the middle, refine as higher or lower

- Searching for GATTACA
  - Lo = 1; Hi = 15; Mid = (1+15)/2 = 8
  - Middle = Suffix[8] = CC
    => Higher: Lo = Mid + 1

  - Lo = 9; Hi = 15;

	#	Sequence	Pos
	1	ACAGATTACC…	6
	2	ACC…	13
	3	AGATTACC…	8
	4	ATTACAGATTACC…	3
	5	ATTACC…	10
	6	C…	15
	7	CAGATTACC…	7
Lo →	8	CC…	14
	9	GATTACAGATTACC…	2
	10	GATTACC…	9
	11	TACAGATTACC…	5
	12	TACC…	12
	13	TGATTACAGATTACC…	1
	14	TTACAGATTACC…	4
Hi →	15	TTACC…	11

Searching the Index

- Strategy 2: Binary search
  - Compare to the middle, refine as higher or lower

- Searching for GATTACA
  - Lo = 1; Hi = 15; Mid = (1+15)/2 = 8
  - Middle = Suffix[8] = CC

    => Higher: Lo = Mid + 1

  - Lo = 9; Hi = 15; Mid = (9+15)/2 = 12
  - Middle = Suffix[12] = TACC

	#	Sequence	Pos
	1	ACAGATTACC…	6
	2	ACC…	13
	3	AGATTACC…	8
	4	ATTACAGATTACC…	3
	5	ATTACC…	10
	6	C…	15
	7	CAGATTACC…	7
Lo →	8	CC…	14
	9	GATTACAGATTACC…	2
	10	GATTACC…	9
	11	TACAGATTACC…	5
	12	TACC…	12
	13	TGATTACAGATTACC…	1
	14	TTACAGATTACC…	4
Hi →	15	TTACC…	11

Searching the Index

- Strategy 2: Binary search
  - Compare to the middle, refine as higher or lower

- Searching for GATTACA
  - Lo = 1; Hi = 15; Mid = (1+15)/2 = 8
  - Middle = Suffix[8] = CC
    => Higher: Lo = Mid + 1

  - Lo = 9; Hi = 15; Mid = (9+15)/2 = 12
  - Middle = Suffix[12] = TACC
    => Lower: Hi = Mid - 1

  - Lo = 9; Hi = 11;

	#	Sequence	Pos
	1	ACAGATTACC…	6
	2	ACC…	13
	3	AGATTACC…	8
	4	ATTACAGATTACC…	3
	5	ATTACC…	10
	6	C…	15
	7	CAGATTACC…	7
Lo →	8	CC…	14
	9	GATTACAGATTACC…	2
	10	GATTACC…	9
Hi →	11	TACAGATTACC…	5
	12	TACC…	12
	13	TGATTACAGATTACC…	1
	14	TTACAGATTACC…	4
	15	TTACC…	11

Searching the Index

- Strategy 2: Binary search
  - Compare to the middle, refine as higher or lower

- Searching for GATTACA
  - Lo = 1; Hi = 15; Mid = (1+15)/2 = 8
  - Middle = Suffix[8] = CC
    => Higher: Lo = Mid + 1

  - Lo = 9; Hi = 15; Mid = (9+15)/2 = 12
  - Middle = Suffix[12] = TACC
    => Lower: Hi = Mid - 1

  - Lo = 9; Hi = 11; Mid = (9+11)/2 = 10
  - Middle = Suffix[10] = GATTACC

	#	Sequence	Pos
	1	ACAGATTACC…	6
	2	ACC…	13
	3	AGATTACC…	8
	4	ATTACAGATTACC…	3
	5	ATTACC…	10
	6	C…	15
	7	CAGATTACC…	7
	8	CC…	14
Lo →	9	GATTACAGATTACC…	2
	10	GATTACC…	9
Hi →	11	TACAGATTACC…	5
	12	TACC…	12
	13	TGATTACAGATTACC…	1
	14	TTACAGATTACC…	4
	15	TTACC…	11

Searching the Index

- Strategy 2: Binary search
  - Compare to the middle, refine as higher or lower

- Searching for GATTACA
  - Lo = 1; Hi = 15; Mid = (1+15)/2 = 8
  - Middle = Suffix[8] = CC
    => Higher: Lo = Mid + 1

  - Lo = 9; Hi = 15; Mid = (9+15)/2 = 12
  - Middle = Suffix[12] = TACC
    => Lower: Hi = Mid - 1

  - Lo = 9; Hi = 11; Mid = (9+11)/2 = 10
  - Middle = Suffix[10] = GATTACC
    => Lower: Hi = Mid - 1

  - Lo = 9; Hi = 9;

	#	Sequence	Pos
	1	ACAGATTACC…	6
	2	ACC…	13
	3	AGATTACC…	8
	4	ATTACAGATTACC…	3
	5	ATTACC…	10
	6	C…	15
	7	CAGATTACC…	7
	8	CC…	14
Lo → Hi →	9	GATTACAGATTACC…	2
	10	GATTACC…	9
	11	TACAGATTACC…	5
	12	TACC…	12
	13	TGATTACAGATTACC…	1
	14	TTACAGATTACC…	4
	15	TTACC…	11

Searching the Index

- Strategy 2: Binary search
  - Compare to the middle, refine as higher or lower

- Searching for GATTACA
  - Lo = 1; Hi = 15; Mid = (1+15)/2 = 8
  - Middle = Suffix[8] = CC
    => Higher: Lo = Mid + 1

  - Lo = 9; Hi = 15; Mid = (9+15)/2 = 12
  - Middle = Suffix[12] = TACC
    => Lower: Hi = Mid - 1

  - Lo = 9; Hi = 11; Mid = (9+11)/2 = 10
  - Middle = Suffix[10] = GATTACC
    => Lower: Hi = Mid - 1

  - Lo = 9; Hi = 9; Mid = (9+9)/2 = 9
  - Middle = Suffix[9] = GATTACA…
    => Match at position 2!

#	Sequence	Pos
1	ACAGATTACC…	6
2	ACC…	13
3	AGATTACC…	8
4	ATTACAGATTACC…	3
5	ATTACC…	10
6	C…	15
7	CAGATTACC…	7
8	CC…	14
9	GATTACAGATTACC…	2
10	GATTACC…	9
11	TACAGATTACC…	5
12	TACC…	12
13	TGATTACAGATTACC…	1
14	TTACAGATTACC…	4
15	TTACC…	11

Lo
Hi

Binary Search Analysis

- Binary Search

  Initialize search range to entire list

  mid = (hi+lo)/2; middle = suffix[mid]

  if query matches middle: done

  else if query < middle: pick low range

  else if query > middle: pick hi range

  Repeat until done or empty range [WHEN?]

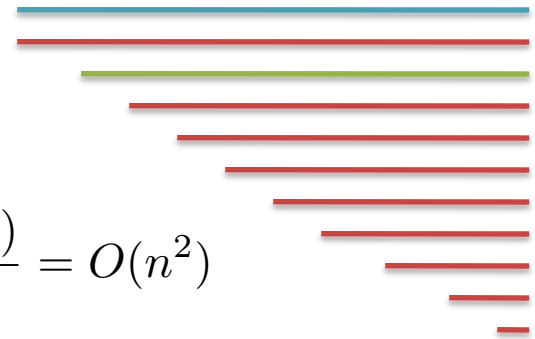
- Analysis
  - More complicated method
  - How many times do we repeat?
    - How many times can it cut the range in half?
    - Find smallest x such that: $n/(2^x) \leq 1$; $x = \lg_2(n)$ [32]

- Total Runtime: O(m lg n)
  - More complicated, but much faster!
  - Looking up a query loops 32 times instead of 3B

[How long does it take to search 6B or 24B nucleotides?]

Suffix Array Construction

- How can we store the suffix array?
  [How many characters are in all suffixes combined?]



$$S = 1 + 2 + 3 + \cdots + n = \sum_{i=1}^{n} i = \frac{n(n+1)}{2} = O(n^2)$$

  - Hopeless to explicitly store 4.5 billion billion characters
  - Instead use implicit representation
    - Keep 1 copy of the genome, and a list of sorted offsets
    - Storing 3 billion offsets fits on a server (12GB)

- Searching the array is very fast, but it takes time to construct
  - This time will be amortized over many, many searches
  - Run it once "overnight" and save it away for all future queries

Pos
6
13
8
3
10
15
7
14
2
9
5
12
1
4
11

TGATTACAGATTACC

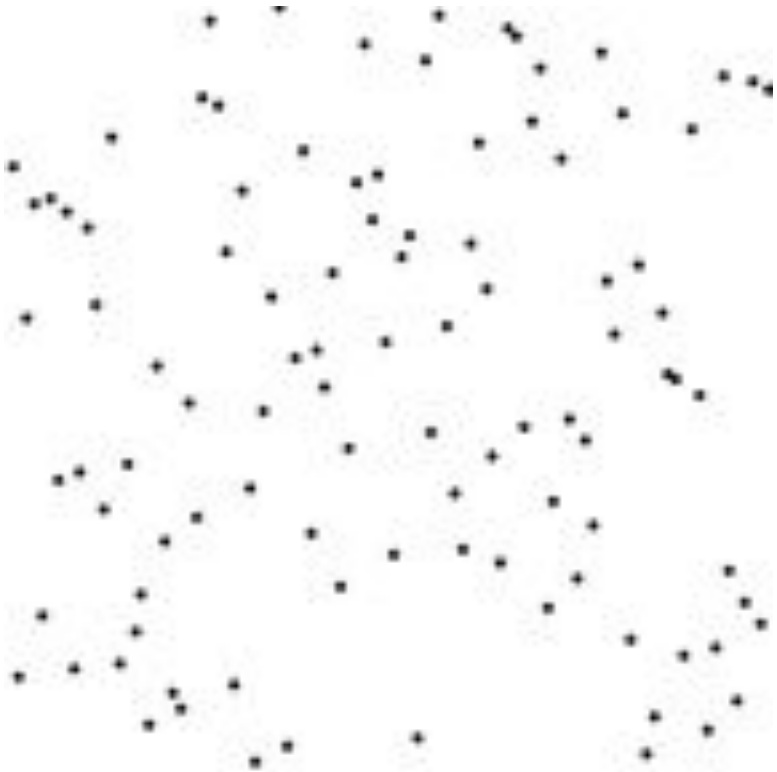
Sorting

Quickly sort these numbers into ascending order:

14, 29, 6, 31, 39, 64, 78, 50, 13, 63, 61, 19

[How do you do it?]

6, 14, 29, 31, 39, 64, 78, 50, 13, 63, 61, 19
6, 13, 14, 29, 31, 39, 64, 78, 50, 63, 61, 19
6, 13, 14, 19, 29, 31, 39, 64, 78, 50, 63, 61
6, 13, 14, 19, 29, 31, 39, 64, 78, 50, 63, 61
6, 13, 14, 19, 29, 31, 39, 64, 78, 50, 63, 61
6, 13, 14, 19, 29, 31, 39, 50, 64, 78, 63, 61
6, 13, 14, 19, 29, 31, 39, 50, 61, 64, 78, 63
6, 13, 14, 19, 29, 31, 39, 50, 61, 63, 64, 78
6, 13, 14, 19, 29, 31, 39, 50, 61, 63, 64, 78
6, 13, 14, 19, 29, 31, 39, 50, 61, 63, 64, 78
6, 13, 14, 19, 29, 31, 39, 50, 61, 63, 64, 78
6, 13, 14, 19, 29, 31, 39, 50, 61, 63, 64, 78



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

Selection Sort Analysis

- Selection Sort (Input: list of n numbers)

```
for pos = 1 to n
    // find the smallest element in [pos, n]
    smallest = pos
    for check = pos+1 to n
        if (list[check] < list[smallest]): smallest = check

    // move the smallest element to the front
    tmp = list[smallest]
    list[pos] = list[smallest]
    list[smallest] = tmp
```

- Analysis

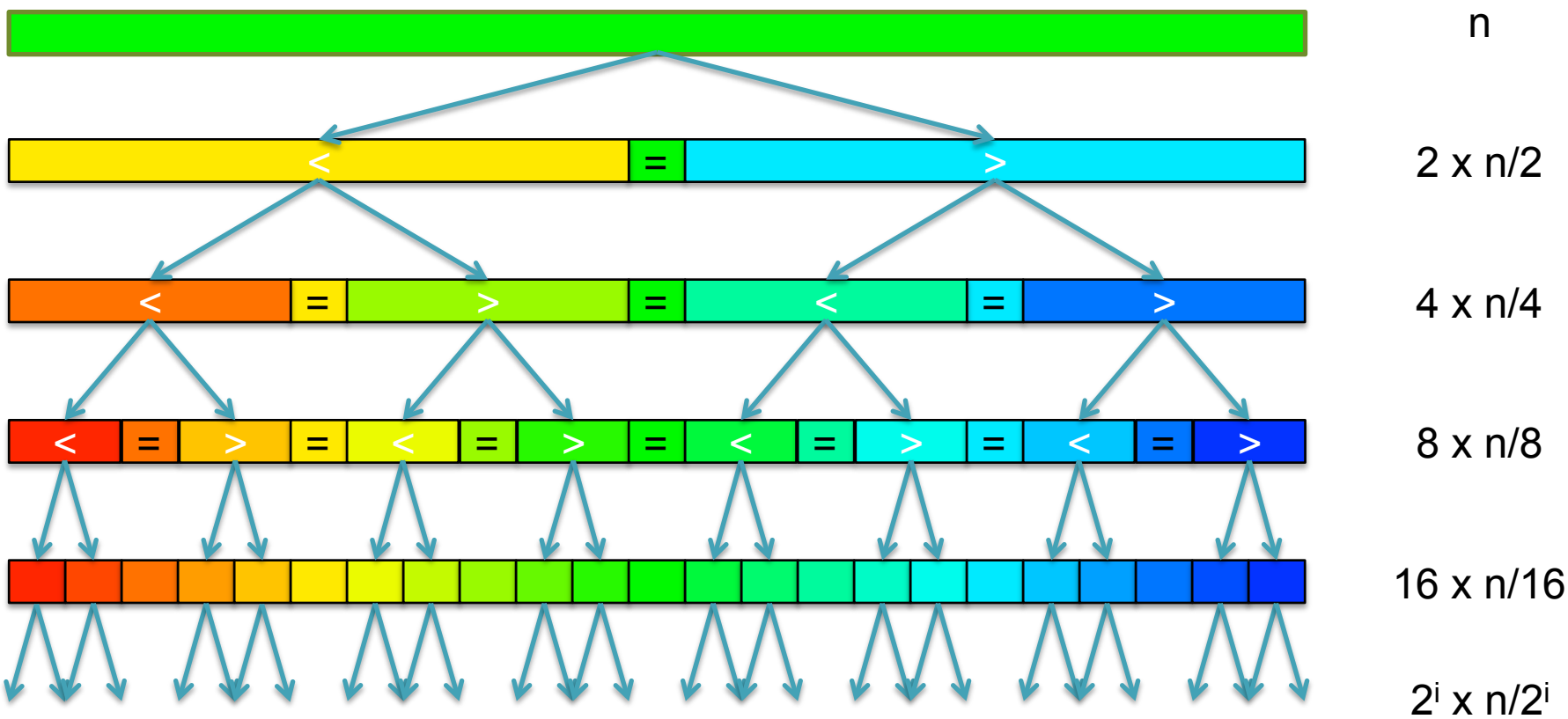
$$T = n + (n-1) + (n-2) + \cdots + 3 + 2 + 1 = \sum_{i=1}^{n} i = \frac{n(n+1)}{2} = O(n^2)$$

  - Outer loop: pos = 1 to n
  - Inner loop: check = pos to n
  - Running time: Outer * Inner = $O(n^2)$ [4.5 Billion Billion]

[Challenge Questions: Why is this slow? / Can we sort any faster?]

Divide and Conquer

- Selection sort is slow because it rescans the entire list for each element
  - How can we split up the unsorted list into independent ranges?
  - Hint 1: Binary search splits up the problem into 2 independent ranges (hi/lo)
  - Hint 2: Assume we know the median value of a list



[How many times can we split a list in half?]

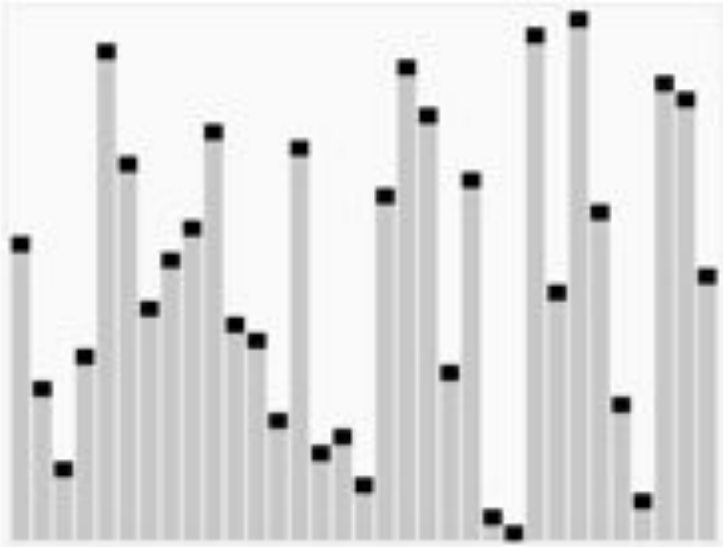
QuickSort Analysis

- QuickSort(Input: list of n numbers)

```
// see if we can quit
if (length(list)) <= 1): return list

// split list into lo & hi
pivot = median(list)
lo = {}; hi = {};
for (i = 1 to length(list))
      if (list[i] < pivot): append(lo, list[i])
      else:                 append(hi, list[i])

// recurse on sublists
return (append(QuickSort(lo), QuickSort(hi))
```



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

- Analysis (Assume we can find the median in O(n))

$$T(n) = \begin{cases} O(1) & \text{if } n \leq 1 \\ O(n) + 2T(n/2) & \text{else} \end{cases}$$

$$T(n) = n + 2(\frac{n}{2}) + 4(\frac{n}{4}) + \cdots + n(\frac{n}{n}) = \sum_{i=0}^{lg(n)} \frac{2^i n}{2^i} = \sum_{i=0}^{lg(n)} n = O(n \lg n)$$ [~94B]

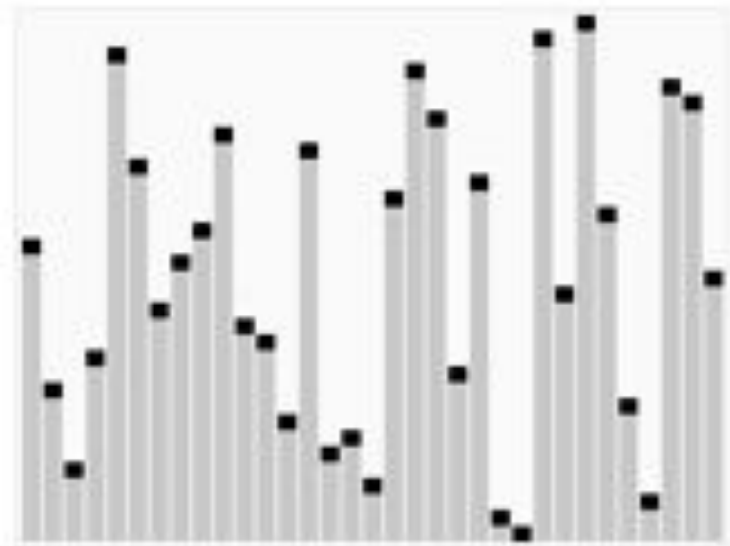
QuickSort Analysis

- QuickSort(Input: list of n numbers)

```
// see if we can quit
if (length(list)) <= 1): return list

// split list into lo & hi
pivot = median(list)
lo = {}; hi = {};
for (i = 1 to length(list))
      if (list[i] < pivot): append(lo, list[i])
      else:                 append(hi, list[i])

// recurse on sublists
return (append(QuickSort(lo), QuickSort(hi))
```



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

- Analysis (Assume we can find the median in O(n))

$$T(n) = \begin{cases} O(1) & \text{if } n \leq 1 \\ O(n) + 2T(n/2) & \text{else} \end{cases}$$

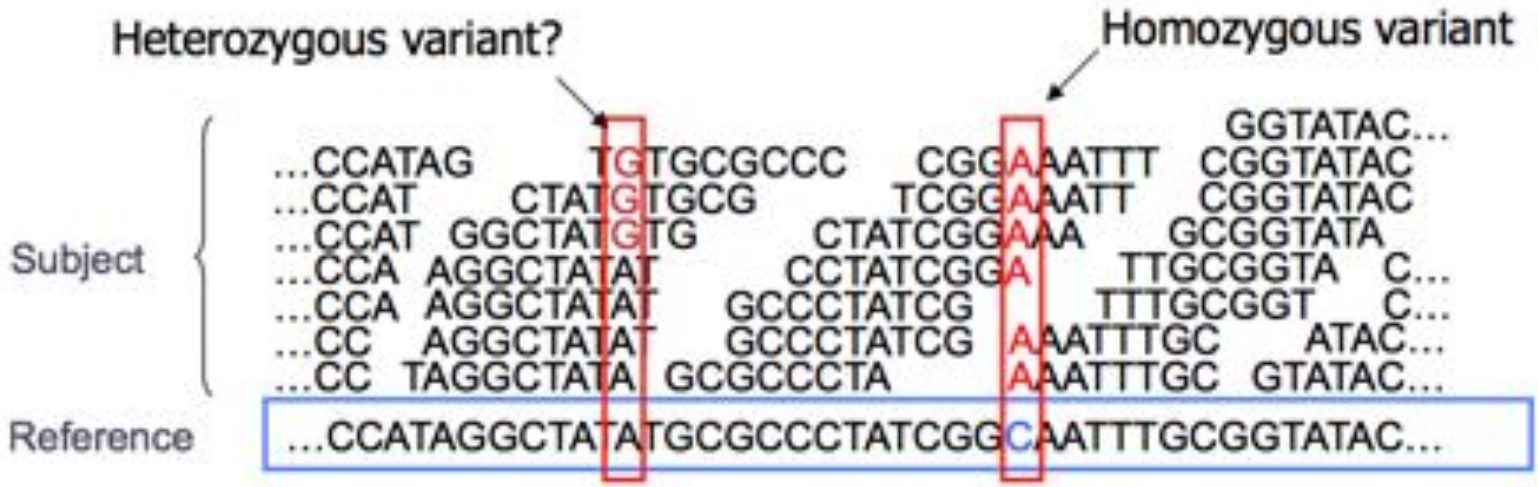
$$T(n) = n + 2(\frac{n}{2}) + 4(\frac{n}{4}) + \cdots + n(\frac{n}{n}) = \sum_{i=0}^{lg(n)} \frac{2^i n}{2^i} = \sum_{i=0}^{lg(n)} n = O(n \lg n)$$ [~94B]

Break

Bowtie: Ultrafast and memory efficient alignment of short DNA sequences to the human genome

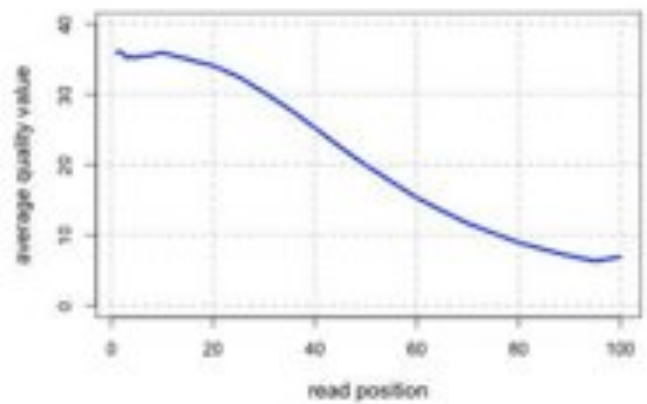
Slides Courtesy of Ben Langmead
(langmead@umiacs.umd.edu)

Genotyping Theory



- Sequencing instruments make mistakes
  - Quality of read decreases over the read length

$$Q_{\text{sanger}} = -10 \log_{10} p$$



- A single read differing from the reference is probably just an error, but it becomes more likely to be real as we see it multiple times
  - Often framed as a Bayesian problem of more likely to be a real variant or chance occurrence of N errors

In-exact alignment

- Where is GATTACA *approximately* in the human genome?
  - And how do we efficiently find them?

- It depends…
  - Define 'approximately'
    - Hamming Distance, Edit distance, or Sequence Similarity
    - Ungapped vs Gapped vs Affine Gaps
    - Global vs Local
    - All positions or the single 'best'?
  - Efficiency depends on the data characteristics & goals
    - Smith-Waterman: Exhaustive search for optimal alignments
    - BLAST: Hash-table based homology searches
    - Bowtie: BWT alignment for short read mapping

Searching for GATTACA

- Where is GATTACA *approximately* in the human genome?

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	…
T	G	A	T	T	A	C	A	G	A	T	T	A	C	C	…
G	A	T	T	A	C	A									

Match Score: 1/7

Searching for GATTACA

- Where is GATTACA *approximately* in the human genome?

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	…
T	G	A	T	T	A	C	A	G	A	T	T	A	C	C	…
	G	A	T	T	A	C	A								

Match Score: 7/7

Searching for GATTACA

- Where is GATTACA *approximately* in the human genome?

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	…
T	G	A	T	T	A	C	A	G	A	T	T	A	C	C	…
		G	A	T	T	A	C	A	…						

Match Score: 1/7

Searching for GATTACA

- Where is GATTACA *approximately* in the human genome?

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	…
T	G	A	T	T	A	C	A	G	A	T	T	A	C	C	…
								G	A	T	T	A	C	A	

Match Score: 6/7 <- We may be very interested in these imperfect matches
Especially if there are no perfect end-to-end matches

Similarity metrics

- Hamming distance
  - Count the number of substitutions to transform one string into another

```
GATTACA        ATTACCC
|||x|||        xx|xx|x
GATCACA        GATTACA
   1              5
```

- Edit distance
  - The minimum number of substitutions, insertions, or deletions to transform one string into another

```
GATTACA        -ATTACCC
|||x|||        x|||||xx
GATCACA        GATTAC-A
   1              3
```

Edit Distance Example

AGCACACA → ACACACTA in 4 steps

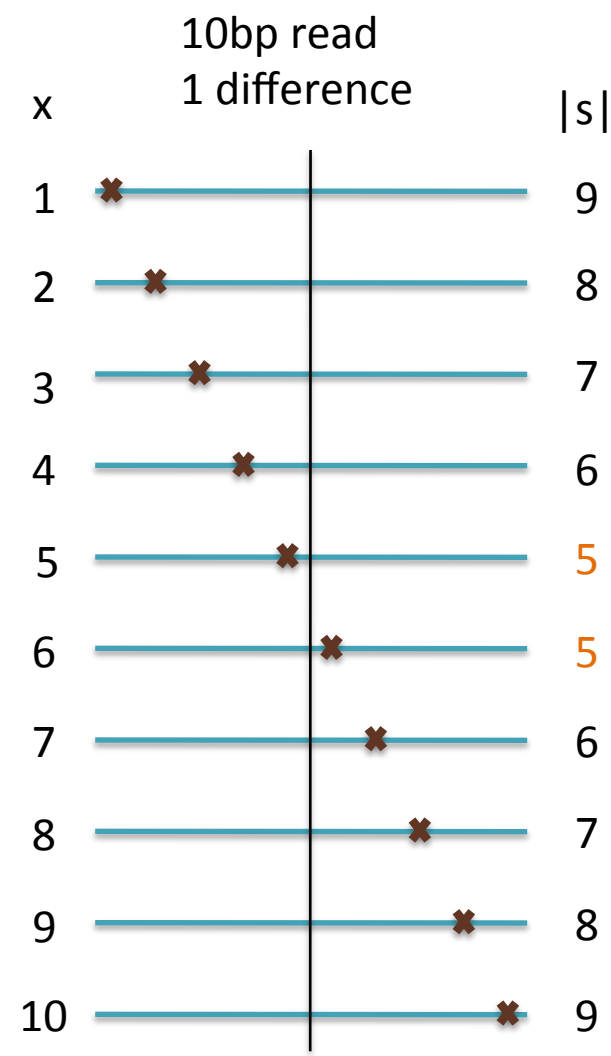
AGCACACA → (1. change G to C)
ACCACACA → (2. delete C)
ACACACA → (3. change A to T)
ACACACT → (4. insert A after T)
ACACACTA → done

[Is this the best we can do?]

Seed-and-Extend Alignment

Notice binary search doesn't work for in-exact alignment because 1st (or any) character could be different

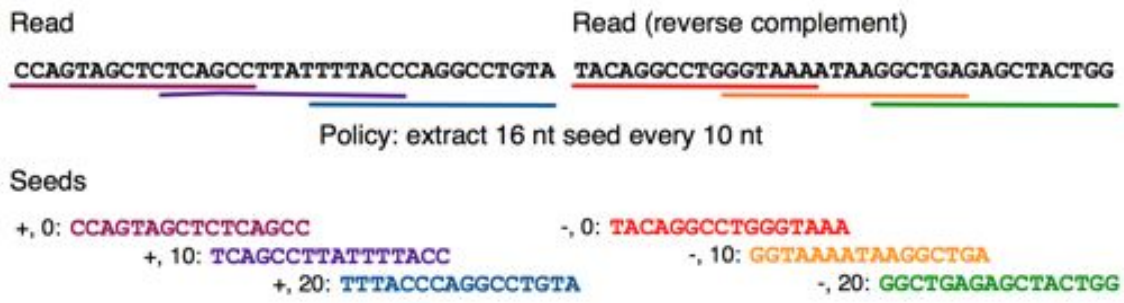
Theorem: An alignment of a sequence of length *m* with at most *k* differences ***must*** contain an exact match at least $s=m/(k+1)$ bp long
(*Baeza-Yates* and Perleberg, 1996)



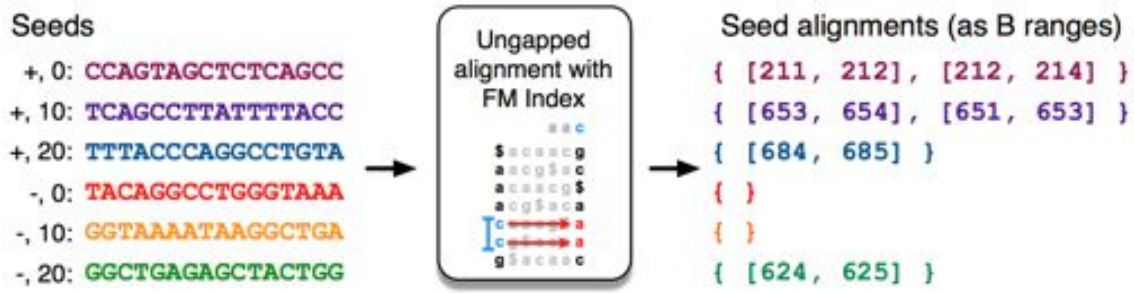
- Proof: Pigeonhole principle
  - 1 pigeon can't fill 2 holes
- Seed-and-extend search
  - Use an index to rapidly find short exact alignments to seed longer in-exact alignments
    - BLAST, MUMmer, Bowtie, BWA, SOAP, …
  - Specificity of the depends on seed length
    - Guaranteed sensitivity for k differences
    - Also finds some (but not all) lower quality alignments <- heuristic

Bowtie2 Overview

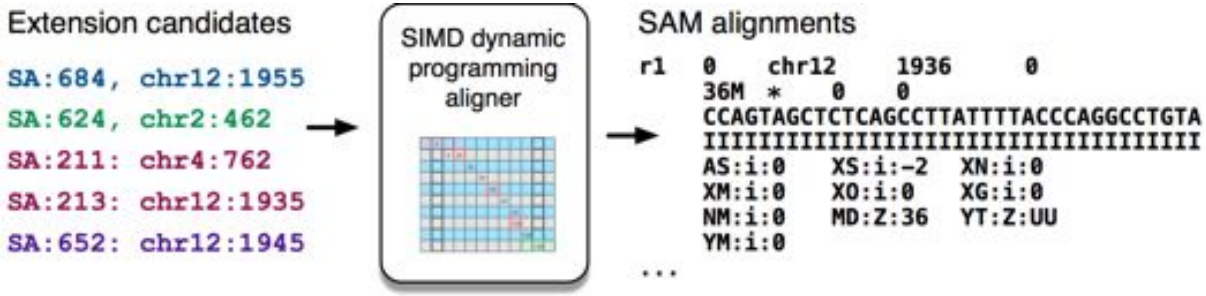
1. Split read into segments



2. Lookup each segment and prioritize



3. Evaluate end-to-end match



Algorithms Summary

- Algorithms choreograph the dance of data inside the machine
  - Algorithms add provable precision to your method
  - A smarter algorithm can solve the same problem with much less work

- Techniques
  - Analysis: Characterize performance, correctness
  - Modeling: Characterize what you expect to see

  - Binary search: Fast lookup in any sorted list
  - Divide-and-conquer: Split a hard problem into an easier problem
  - Recursion: Solve a problem using a function of itself
  - Indexing: Focus on just the important parts
  - Seed-and-extend: Anchor the problem using a portion of it

"Think Harder and Compute Less"

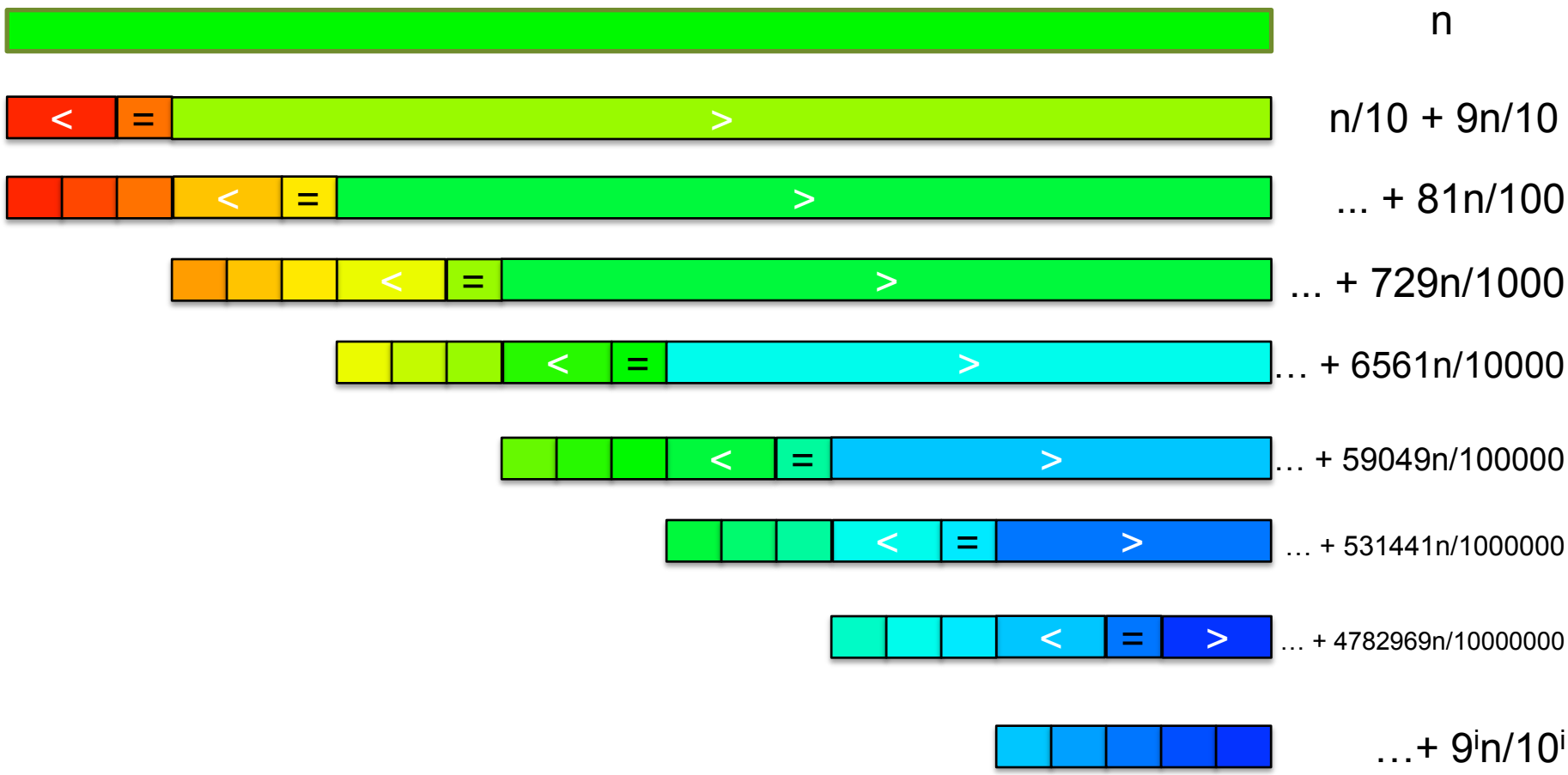
Dan Gusfield ~ UC Davis

Questions?

http://schatzlab.cshl.edu
@mike_schatz

Picking the Median

- What if we miss the median and do a 90/10 split instead?



[How many times can we cut 10% off a list?]

Randomized Quicksort

- 90/10 split runtime analysis

$$T(n) = n + T(\frac{n}{10}) + T(\frac{9n}{10})$$

$$T(n) = n + \frac{n}{10} + T(\frac{n}{100}) + T(\frac{9n}{100}) + \frac{9n}{10} + T(\frac{9n}{100}) + T(\frac{81n}{100})$$

$$T(n) = n + n + T(\frac{n}{100}) + 2T(\frac{9n}{100}) + T(\frac{81n}{100})$$

$$T(n) = \sum_{i=0}^{\log_{10/9}(n)} n = O(n \lg n)$$

Find smallest x s.t.

$$(9/10)^x n \leq 1$$

$$(10/9)^x \geq n$$

$$x \geq \log_{10/9} n$$

- If we randomly pick a pivot, we will get at least a 90/10 split with very high probability
  - Everything is okay as long as we always slice off a fraction of the list

[Challenge Question: What happens if we slice 1 element]

Edit Distance Example

AGCACACA → ACACACTA in 3 steps

AGCACACA → (1. change G to C)
ACCACACA → (2. delete C)
ACACACA → (3. insert T after 3rd C)
ACACACTA → done

[Is this the best we can do?]

Dynamic Programming Matrix

		A	C	A	C	A	C	T	A
	0	1	2	3	4	5	6	7	8
A	1	0	1	2	3	4	5	6	7
G	2	1	1	2	3	4	5	6	7
C	3	2	1	2	2	3	4	5	6
A	4	3	2	1	2	2	3	4	5
C	5	4	3	2	1	2	2	3	4
A	6	5	4	3	2	1	2	3	3
C	7	6	5	4	3	2	1	2	3
A	8	7	6	5	4	3	2	2	2

D[AGCACACA,ACACACTA] = 2

```
AGCACAC-A
|*|||||*|
A-CACACTA
```

[Can we do it any better?]