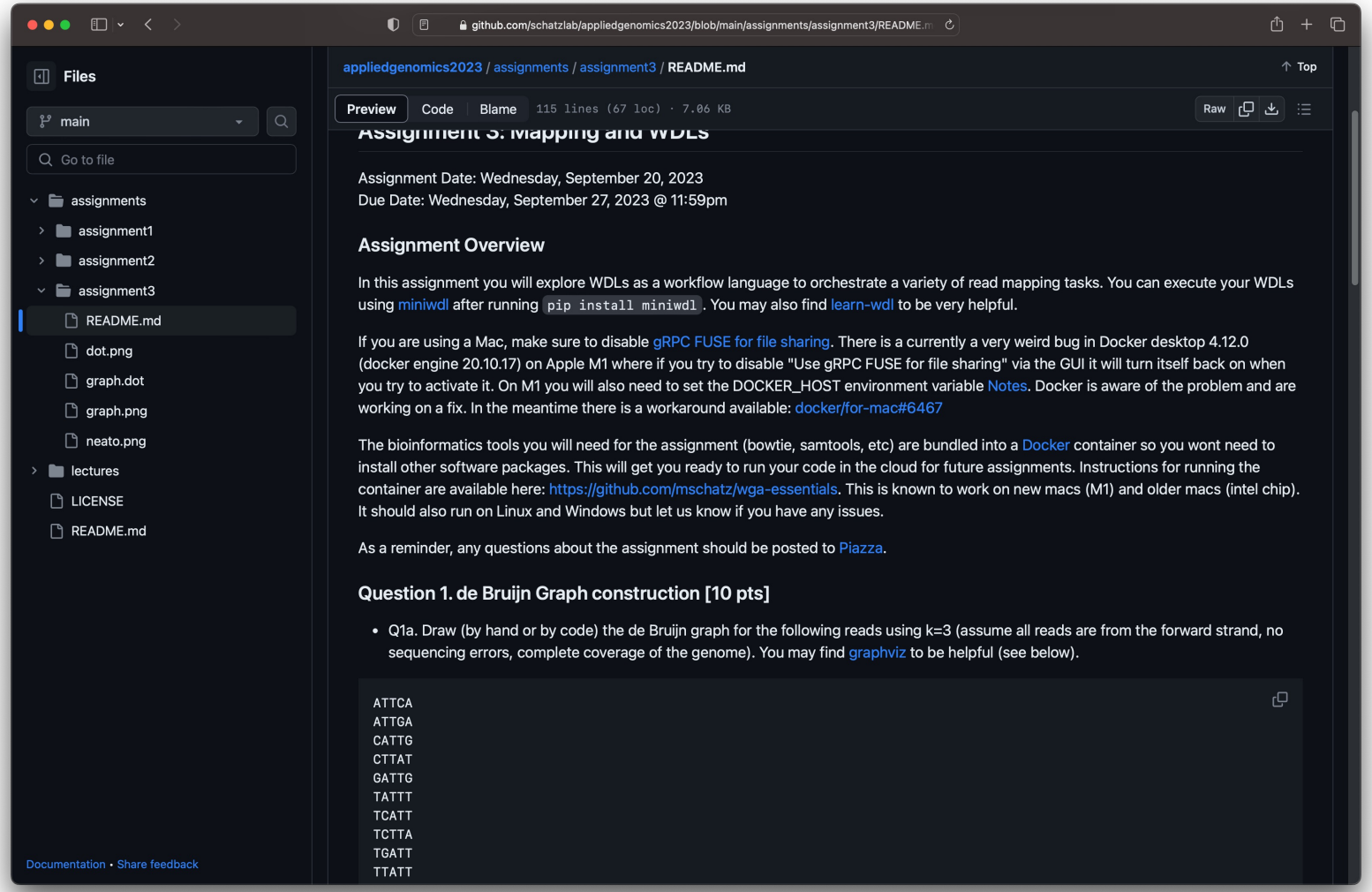
Read Mapping

Michael Schatz

Sept 25, 2023

Lecture 8: Applied Comparative Genomics

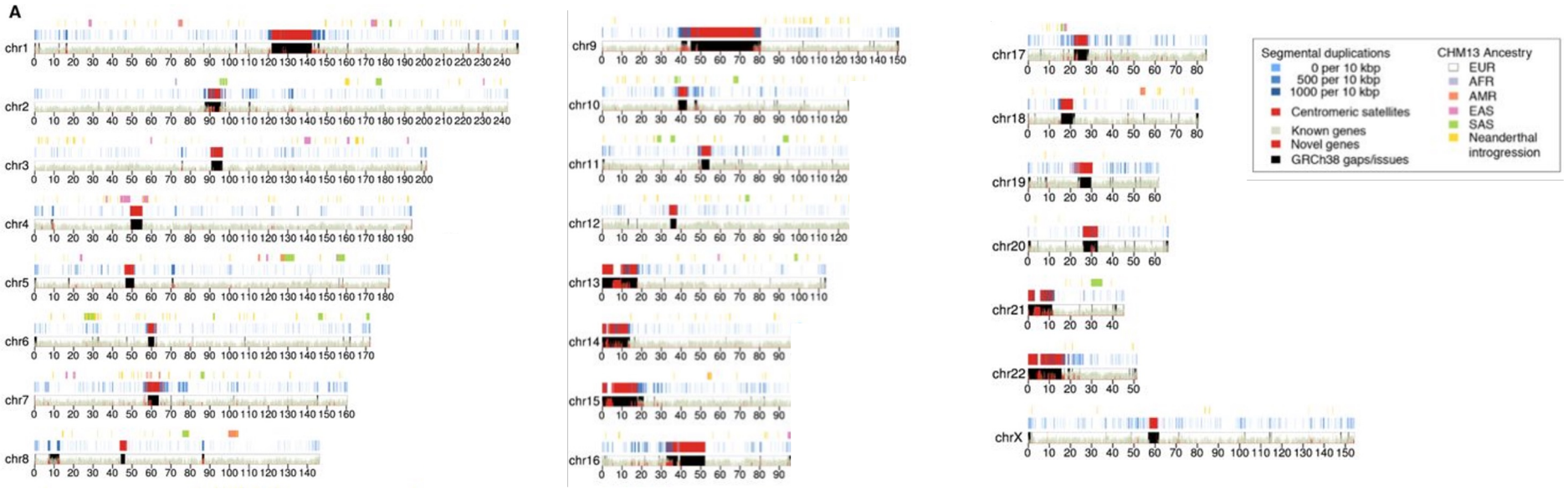
Assignment 3: Genome Assembly
Due Wednesday Sept 27 by 11:59pm



https://github.com/schatzlab/appliedgenomics2023/tree/main/assignments/assignment3

Check Piazza for questions!

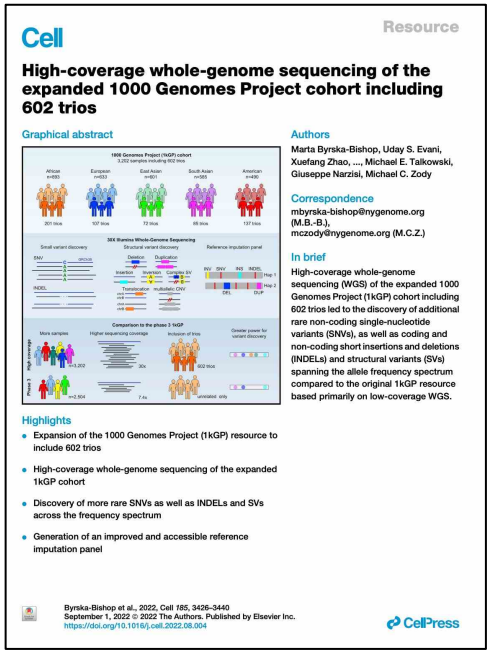
The *complete* sequence of a human genome



CHM13v1.1 genome size is **3.057 Gbp with zero Ns**
Every chromosome is telomere-to-telomere, quality estimated >Q70
~190 Mbp (~8%) of new sequence vs. GRCh38, fixes thousands of errors

(Nurk et al. Science, 2022)

T2T Variants: 1000 Genomes Project



Cell

Resource

High-coverage whole-genome sequencing of the expanded 1000 Genomes Project cohort including 602 trios

Graphical abstract

Authors

Marta Byrska-Bishop, Uday S. Evani, Xuefang Zhao, ..., Michael E. Talkowski, Giuseppe Narzisi, Michael C. Zody

Correspondence

mbyrska-bishop@nygenome.org (M.B.-B.), mczody@nygenome.org (M.C.Z.)

In brief

High-coverage whole-genome sequencing (WGS) of the expanded 1000 Genomes Project (1kGP) cohort including 602 trios led to the discovery of additional rare non-coding single-nucleotide variants (SNVs), as well as coding and non-coding short insertions and deletions (INDELs) and structural variants (SVs) spanning the allele frequency spectrum compared to the original 1kGP resource based primarily on low-coverage WGS.

Highlights

- Expansion of the 1000 Genomes Project (1kGP) resource to include 602 trios
- High-coverage whole-genome sequencing of the expanded 1kGP cohort
- Discovery of more rare SNVs as well as INDELs and SVs across the frequency spectrum
- Generation of an improved and accessible reference imputation panel

Byrska-Bishop et al., 2022, Cell 185, 3426–3440
September 1, 2022 © 2022 The Authors. Published by Elsevier Inc.
https://doi.org/10.1016/j.cell.2022.08.004

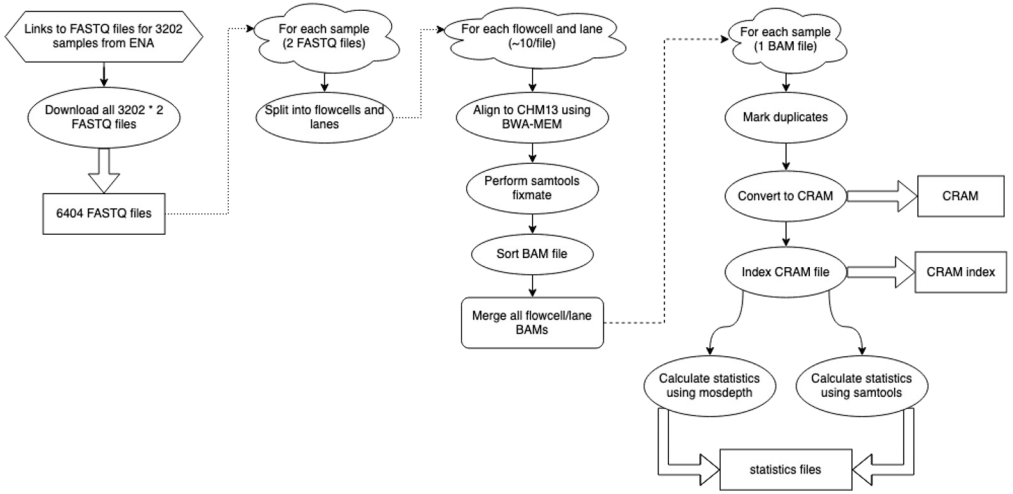
CellPress

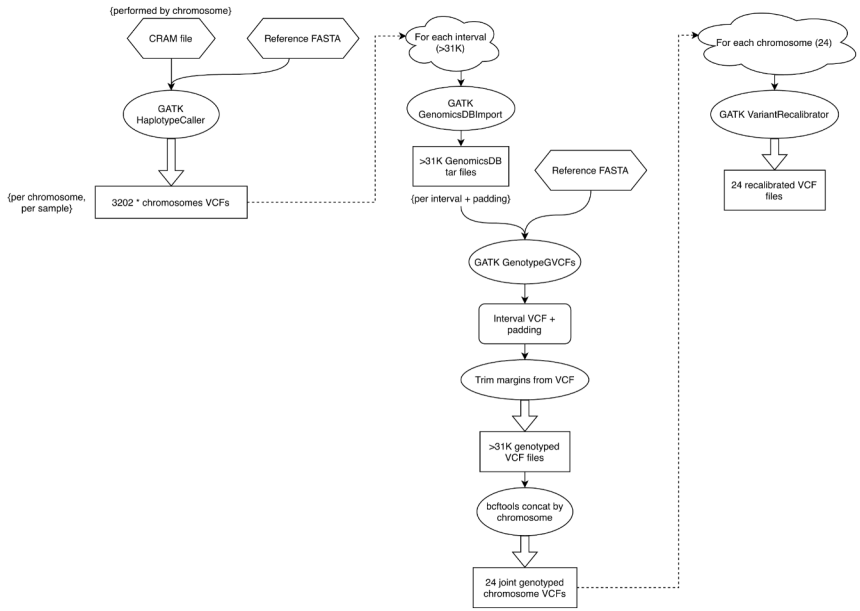
26 populations from 5 superpopulations (continental regions)
3202 samples (2504 core genomes + 698 offspring)

3202 samples x 30Gb = 96Tb input data | >5Pb of intermediate data | >>1M core hours

3202 Genomes to go…

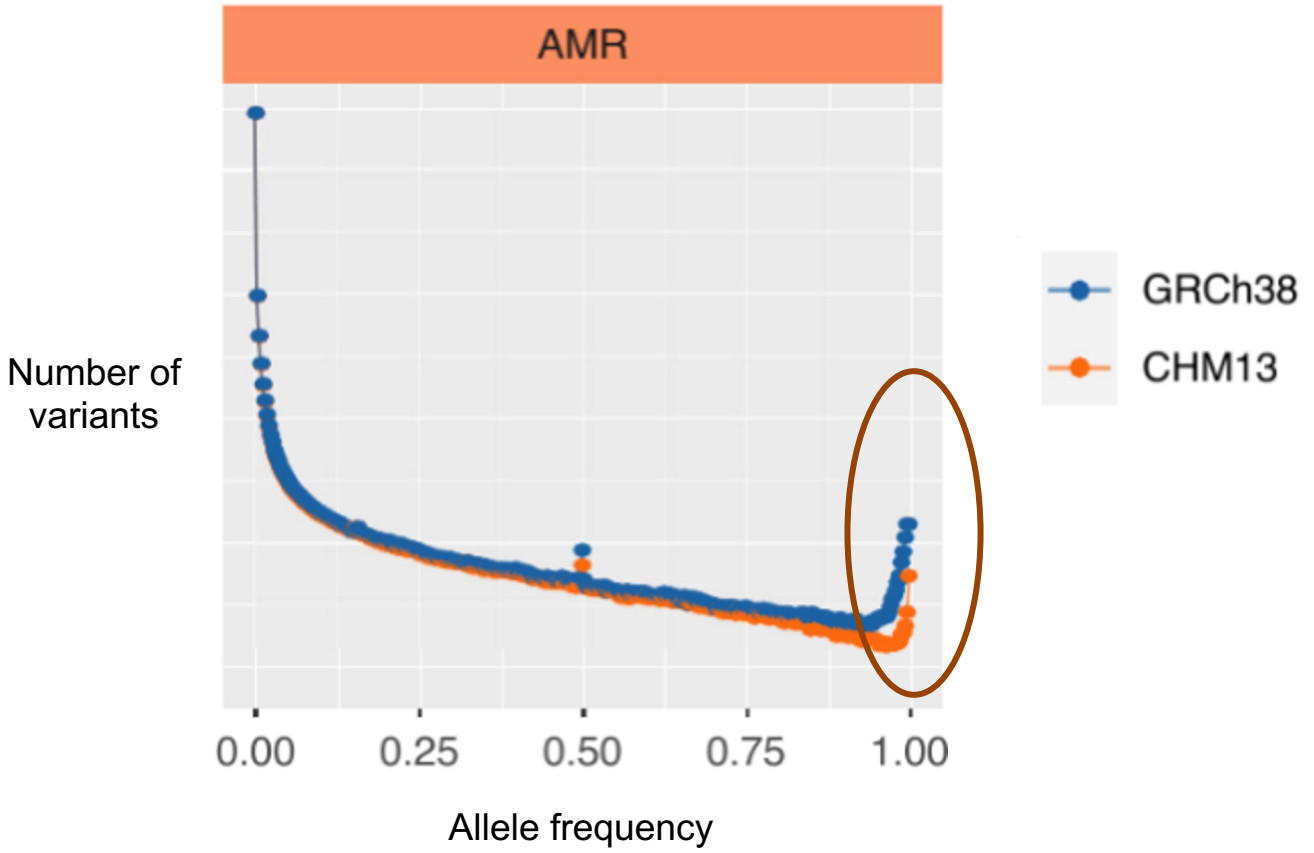
Samantha Zarate



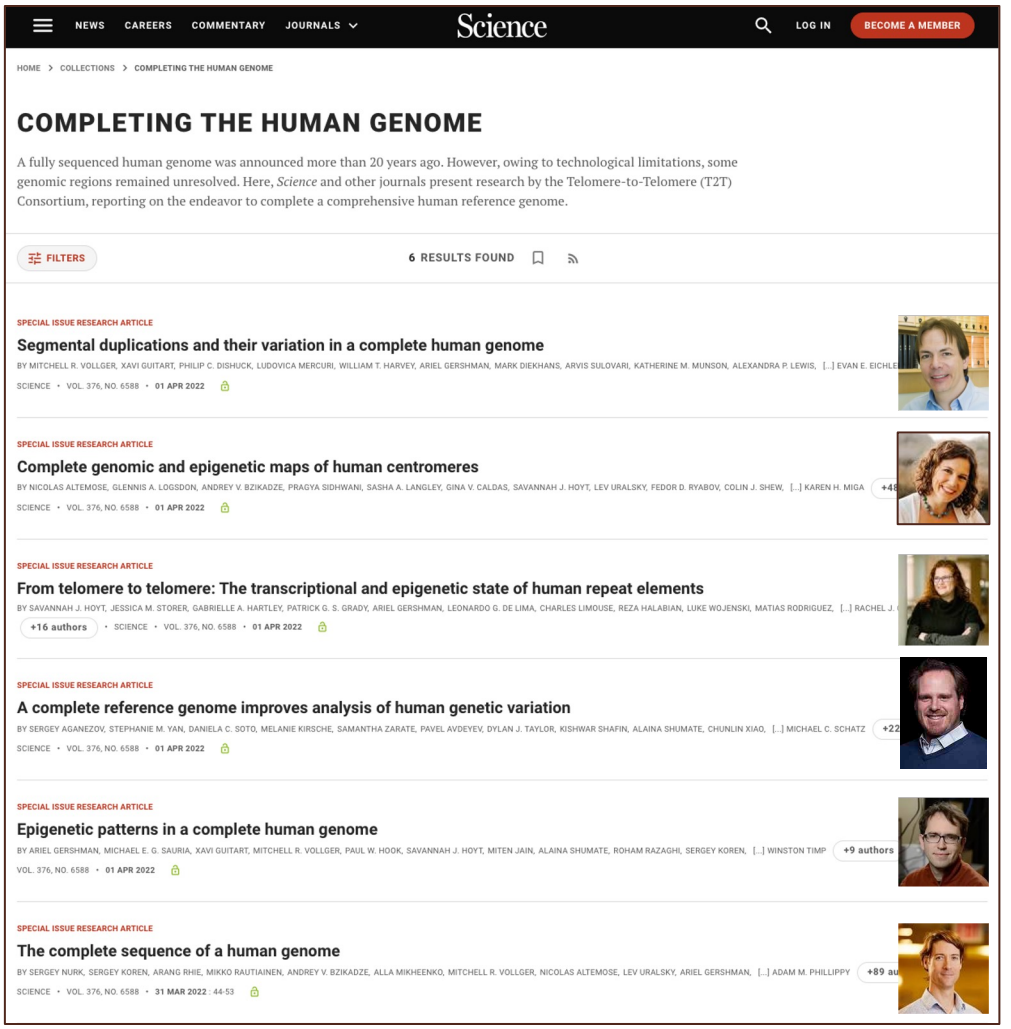


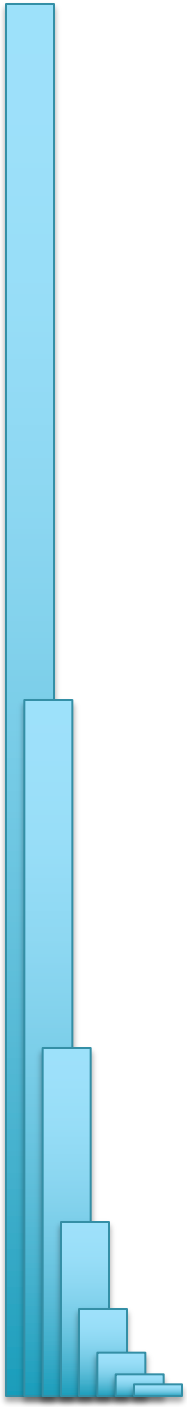

```
task samtoolsStats {
    input {
        File inputCram
        File cramIndex
        File targetRef
        String sampleName
    }

    command <<<
        samtools stats -r "~{targetRef}" \
            --reference "~{targetRef}" \
            -@ "$(nproc)" \
            "~{inputCram}" > "~{sampleName}.samtools.stats.txt"
    >>>

    Int diskGb = ceil(2.0 * size(inputCram, "G"))

    runtime {
        docker : "szarate/t2t_variants:v0.0.2"
        disks : "local-disk ${diskGb} SSD"
        memory: "12G"
        cpu : 16
        preemptible: 3
        maxRetries: 3
    }

    output {
        File stats = "~{sampleName}.samtools.stats.txt"
    }
}
```

Explaining Decreased Per-Sample Count




Earth's heart of iron begins to yield its secrets p. 18
Microglia in chronic pain recovery and relapse pp. 33 & 86
Particle acceleration in a nova explosion p. 77
Science
$15
1 APRIL 2022
SPECIAL ISSUE
science.org
AAAS
FILLING THE GAPS
Closing in on a complete human genome p. 42

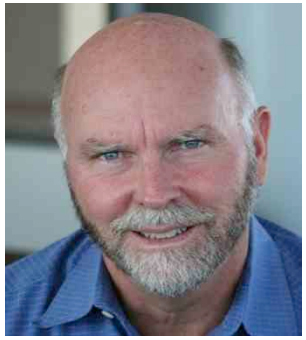

NEWS CAREERS COMMENTARY JOURNALS
Science
LOG IN
BECOME A MEMBER
HOME > COLLECTIONS > COMPLETING THE HUMAN GENOME
COMPLETING THE HUMAN GENOME
A fully sequenced human genome was announced more than 20 years ago. However, owing to technological limitations, some genomic regions remained unresolved. Here, Science and other journals present research by the Telomere-to-Telomere (T2T) Consortium, reporting on the endeavor to complete a comprehensive human reference genome.
FILTERS
6 RESULTS FOUND
SPECIAL ISSUE RESEARCH ARTICLE
Segmental duplications and their variation in a complete human genome
SCIENCE • VOL. 376, NO. 6588 • 01 APR 2022
SPECIAL ISSUE RESEARCH ARTICLE
Complete genomic and epigenetic maps of human centromeres
SCIENCE • VOL. 376, NO. 6588 • 01 APR 2022
SPECIAL ISSUE RESEARCH ARTICLE
From telomere to telomere: The transcriptional and epigenetic state of human repeat elements
+16 authors • SCIENCE • VOL. 376, NO. 6588 • 01 APR 2022
SPECIAL ISSUE RESEARCH ARTICLE
A complete reference genome improves analysis of human genetic variation
SCIENCE • VOL. 376, NO. 6588 • 01 APR 2022
SPECIAL ISSUE RESEARCH ARTICLE
Epigenetic patterns in a complete human genome
+9 authors
VOL. 376, NO. 6588 • 01 APR 2022
SPECIAL ISSUE RESEARCH ARTICLE
The complete sequence of a human genome
SCIENCE • VOL. 376, NO. 6588 • 31 MAR 2022 : 44-53

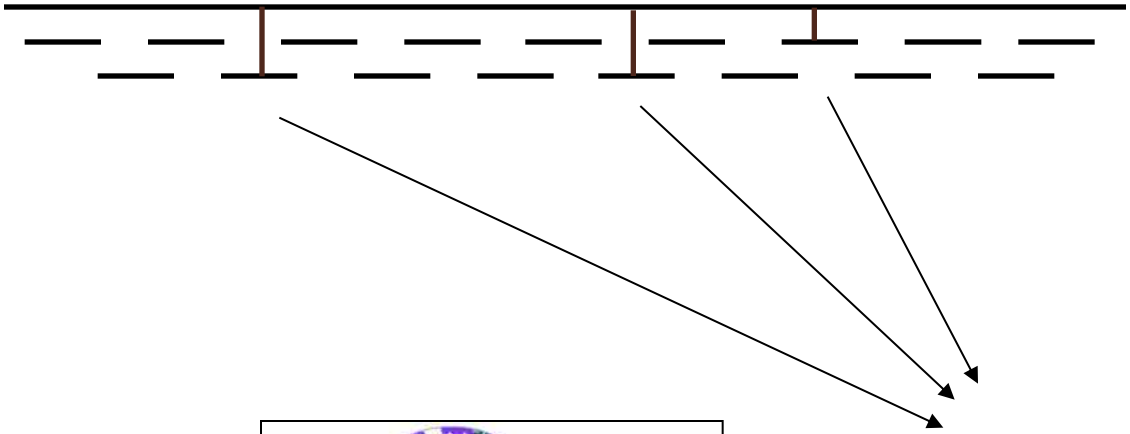


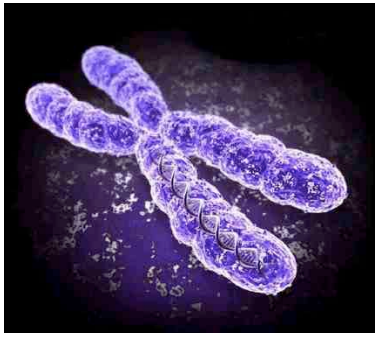
Read Mapping

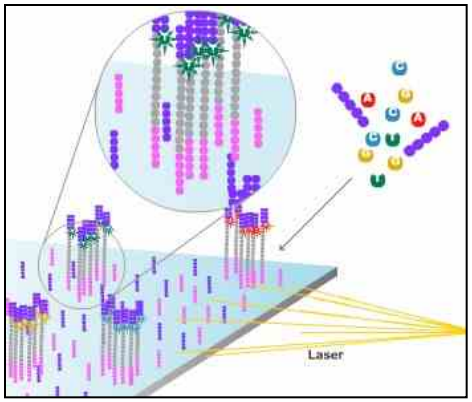
Personal Genomics

How does your genome compare to the reference?









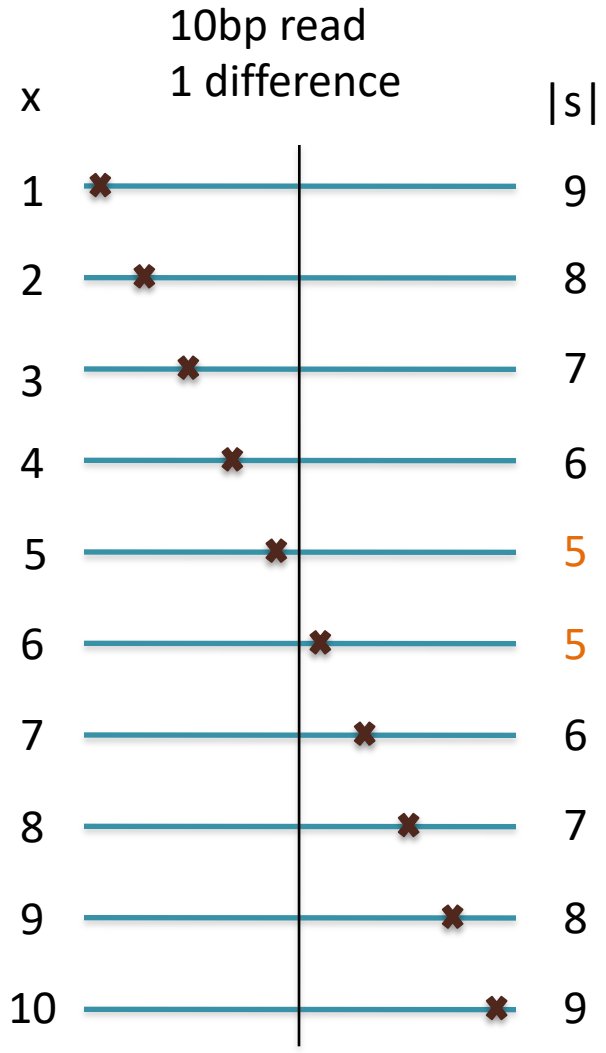
Heart Disease

Cancer

Presidential smile

Seed-and-Extend Alignment

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

SAM / BAM Files

```
Coor      12345678901234  5678901234567890123456789012345
ref       AGCATGTTAGATAA**GATAGCTGTGCTAGTAGGCAGTCAGCGCCAT

+r001/1         TTAGATAAAGGATA*CTG
+r002         aaaAGATAA*GGATA
+r003       gcctaAGCTAA
+r004                       ATAGCT..............TCAGC
-r003                              ttagctTAGGC
-r001/2                                        CAGCGGCAT
```

```
@HD VN:1.6 SO:coordinate
@SQ SN:ref LN:45
r001   99 ref  7 30 8M2I4M1D3M = 37  39 TTAGATAAAGGATACTG *
r002    0 ref  9 30 3S6M1P1I4M *  0   0 AAAAGATAAGGATA    *
r003    0 ref  9 30 5S6M       *  0   0 GCCTAAGCTAA       * SA:Z:ref,29,-,6H5M,17,0;
r004    0 ref 16 30 6M14N5M    *  0   0 ATAGCTTCAGC       *
r003 2064 ref 29 17 6H5M       *  0   0 TAGGC             * SA:Z:ref,9,+,5S6M,30,1;
r001  147 ref 37 30 9M         =  7 -39 CAGCGGCAT         * NM:i:1
```

https://samtools.github.io/hts-specs/SAMv1.pdf

SAM / BAM Files

1.4 The alignment section: mandatory fields

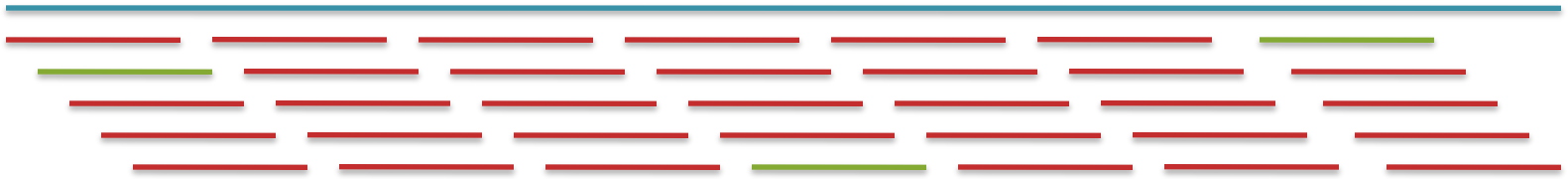
In the SAM format, each alignment line typically represents the linear alignment of a segment. Each line consists of 11 or more TAB-separated fields. The first eleven fields are always present and in the order shown below; if the information represented by any of these fields is unavailable, that field's value will be a placeholder, either '0' or '*' as determined by the field's type. The following table gives an overview of these mandatory fields in the SAM format:

Col	Field	Type	Regexp/Range	Brief description
1	QNAME	String	`[!-?A-~]{1,254}`	Query template NAME
2	FLAG	Int	$[0, 2^{16} - 1]$	bitwise FLAG
3	RNAME	String	`\*|[:rname:^*=][:rname:]*`	Reference sequence NAME[11]
4	POS	Int	$[0, 2^{31} - 1]$	1-based leftmost mapping POSition
5	MAPQ	Int	$[0, 2^{8} - 1]$	MAPping Quality
6	CIGAR	String	`\*|([0-9]+[MIDNSHPX=])+`	CIGAR string
7	RNEXT	String	`\*|=|[:rname:^*=][:rname:]*`	Reference name of the mate/next read
8	PNEXT	Int	$[0, 2^{31} - 1]$	Position of the mate/next read
9	TLEN	Int	$[-2^{31} + 1, 2^{31} - 1]$	observed Template LENgth
10	SEQ	String	`\*|[A-Za-z=.]+`	segment SEQuence
11	QUAL	String	`[!-~]+`	ASCII of Phred-scaled base QUALity+33

[11] Reference sequence names may contain any printable ASCII characters with the exception of certain punctuation characters, and may not start with '*' or '='. See Section 1.2.1 for details and an explanation of the `[:rname:]` notation.

https://samtools.github.io/hts-specs/SAMv1.pdf

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?]

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

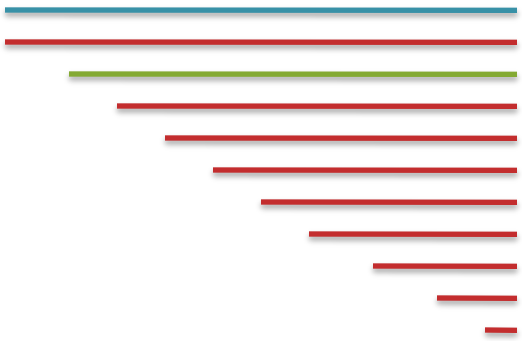
– For one-off scans, this is the best we can do (optimal performance)

- 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

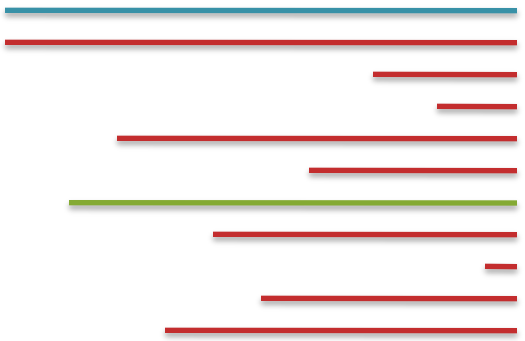
Suffix Arrays: Searching the Phone Book

- 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
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 →

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
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 →

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
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 →

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

Lo →

Hi →

#	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

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
9	GATTACAGATTACC…	2
10	GATTACC…	9
11	TACAGATTACC…	5
12	TACC…	12
13	TGATTACAGATTACC…	1
14	TTACAGATTACC…	4
15	TTACC…	11

Lo
Hi

Searching the Index

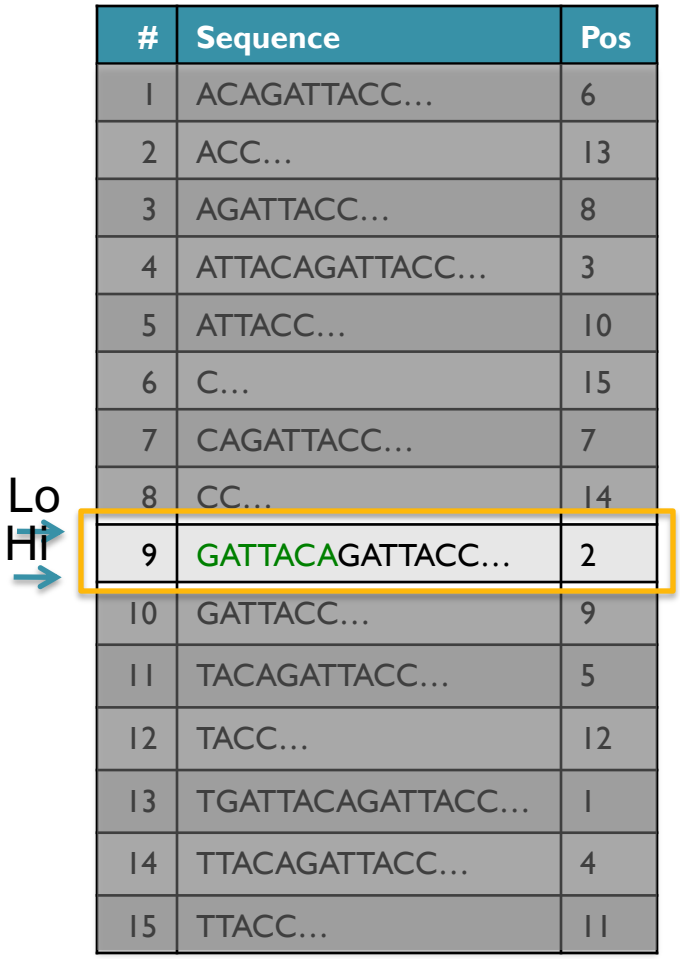
- Strategy 2: Binary search
  - Compare to the middle, refine as higher or lower

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

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

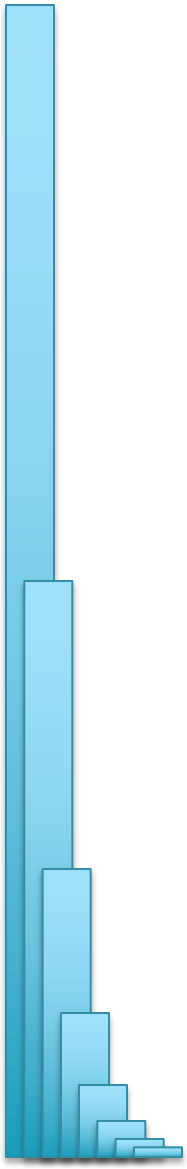
  - Lo = 9; Hi = 11; Mid = (9+11)/2 = 10
  - Middle = Suffix[10] = GATTACC
    => Lower: Hi = Mid - 1

  - 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

Binary Search Analysis



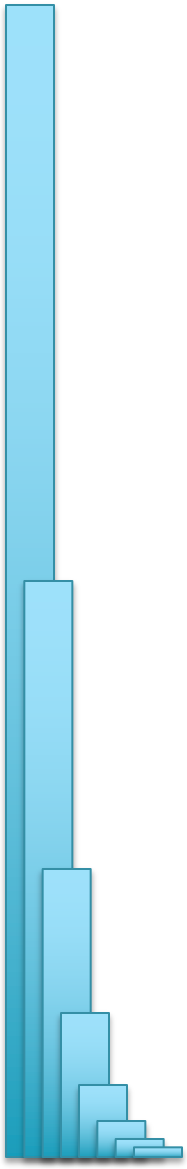
- Binary Search
  - Initialize search range to entire list
    - mid = (hi+lo)/2; middle = suffix[mid]
    - if query matches middle: done
    - else if query < middle: pick low range
    - else if query > middle: pick hi range
  - Repeat until done or empty range [WHEN?]

- 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?]

Binary Search Analysis



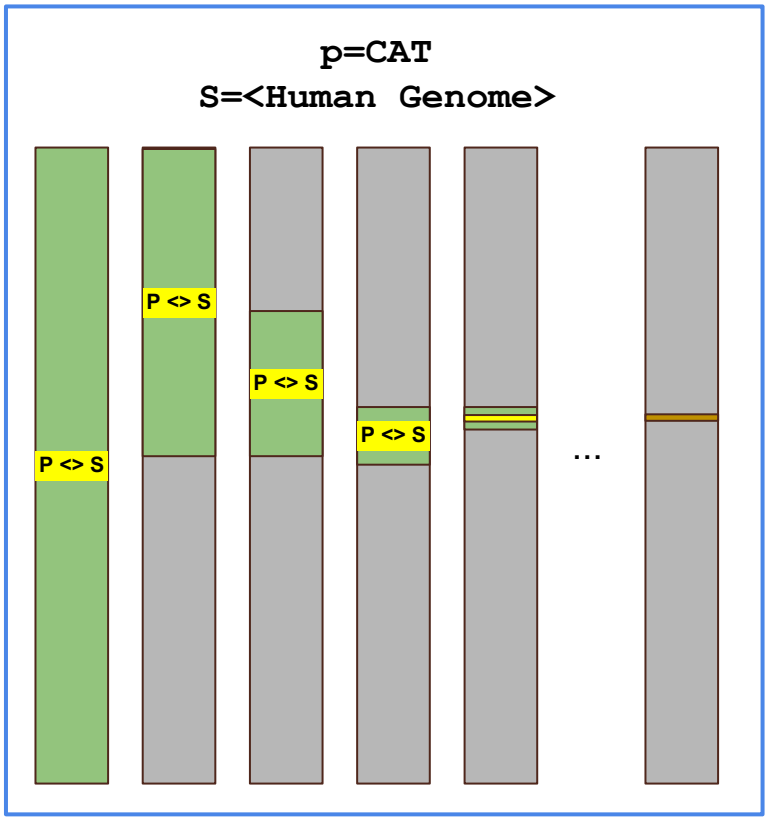
- Binary Search
  - Initialize search range to entire list
    - mid = (hi+lo)/2; middle = suffix[mid]
    - if query matches middle: done
    - else if query < middle: pick low range
    - else if query > middle: pick hi range
  - Repeat until done or empty range [WHEN?]

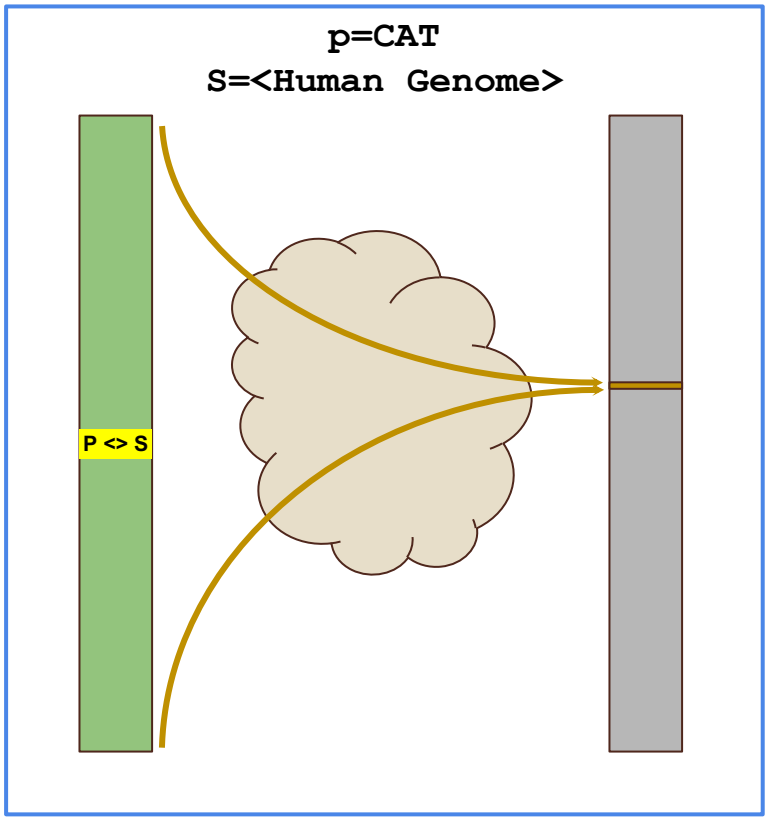
- Analysis
  - More complicated method
  - How many times do we repeat?
    - How many times can it cut the range in half?
    - Find smallest x such that: $n/(2^x) \leq 1$; $x = \lg_2(n)$ [32]

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

Can be reduced to O(m + lg n)
using an auxiliary data structure called the LCP array

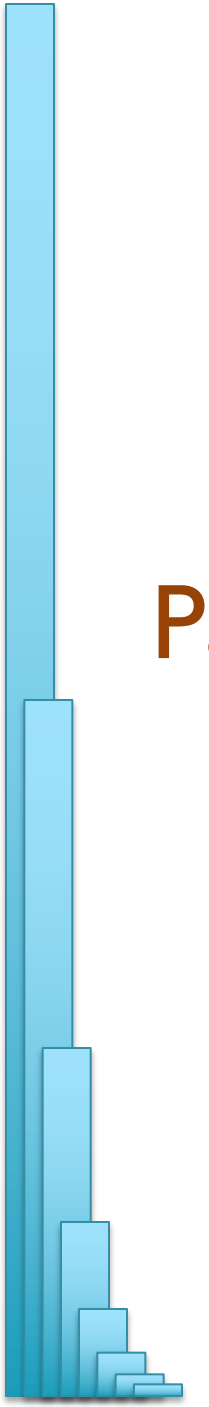
Sapling: Accelerating Suffix Array Queries with Learned Data Models





What if instead of a slow algorithmic approach to find the correct rows, we could somehow quickly guess/predict the correct rows?

Kirsche, M, Das, A, Schatz, MC (2020) Bioinformatics
doi: https://doi.org/10.1093/bioinformatics/btaa911



Part 2: Burrows Wheeler Transform

Algorithmic challenge

How can we combine the speed of a suffix array O(m + lg(n)) (or even O(m)) with the size of a brute force analysis (n bytes)?

What would such an index look like?

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

Slides Courtesy of Ben Langmead

Algorithm Overview

1. Split read into segments

Read: CCAGTAGCTCTCAGCCTTATTTTACCCAGGCCTGTA

Read (reverse complement): TACAGGCCTGGGTAAAATAAGGCTGAGAGCTACTGG

Policy: extract 16 nt seed every 10 nt

Seeds

+, 0: CCAGTAGCTCTCAGCC

+, 10: TCAGCCTTATTTTACC

+, 20: TTTACCCAGGCCTGTA

-, 0: TACAGGCCTGGGTAAA

-, 10: GGTAAAATAAGGCTGA

-, 20: GGCTGAGAGCTACTGG

2. Lookup each segment and prioritize

Seeds

+, 0: CCAGTAGCTCTCAGCC

+, 10: TCAGCCTTATTTTACC

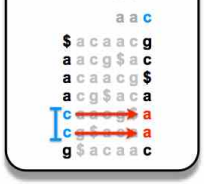
+, 20: TTTACCCAGGCCTGTA

-, 0: TACAGGCCTGGGTAAA

-, 10: GGTAAAATAAGGCTGA

-, 20: GGCTGAGAGCTACTGG

→ Ungapped alignment with FM Index



→ Seed alignments (as B ranges)

{ [211, 212], [212, 214] }

{ [653, 654], [651, 653] }

{ [684, 685] }

{ }

{ }

{ [624, 625] }

3. Evaluate end-to-end match

Extension candidates

SA:684, chr12:1955

SA:624, chr2:462

SA:211: chr4:762

SA:213: chr12:1935

SA:652: chr12:1945

→ SIMD dynamic programming aligner

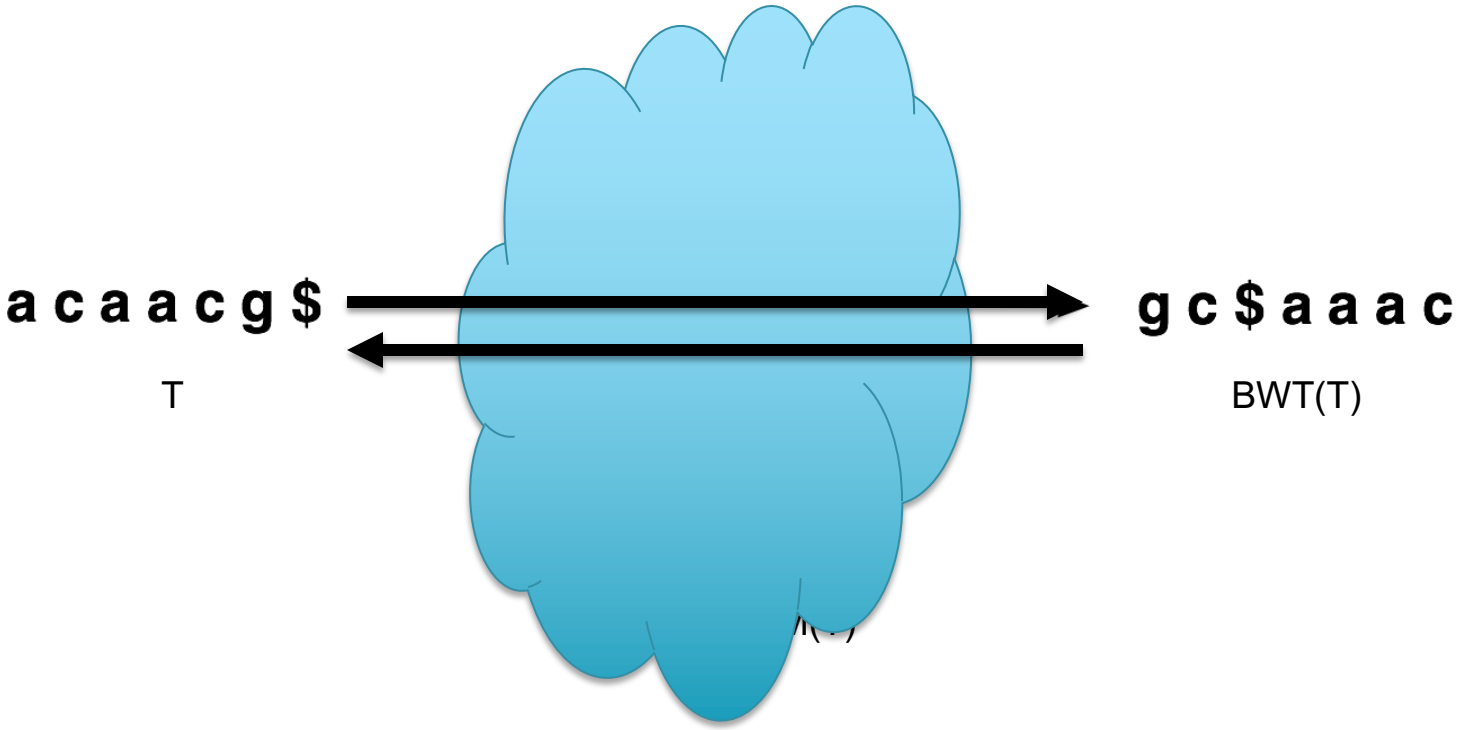
→ SAM alignments

```
r1   0    chr12    1936     0
     36M  *    0    0
     CCAGTAGCTCTCAGCCTTATTTTACCCAGGCCTGTA
     IIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIII
     AS:i:0    XS:i:-2   XN:i:0
     XM:i:0    XO:i:0    XG:i:0
     NM:i:0    MD:Z:36   YT:Z:UU
     YM:i:0
...
```

(Langmead & Salzberg, 2012)

Burrows-Wheeler Transform

- Reversible permutation of the characters in a text



A block sorting lossless data compression algorithm.
Burrows M, Wheeler DJ (1994) *Digital Equipment Corporation.* Technical Report 124