

Variant Calling

Michael Schatz

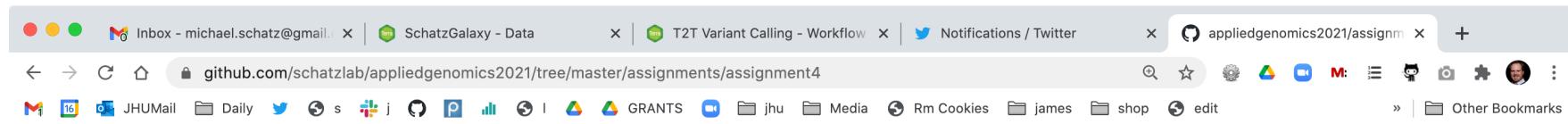
Feb 24, 2021

Lecture 10: Computational Biomedical Research



Assignment 4: Variant Analysis & Mappability

Due March 3 @ 11:59pm



Assignment 4: Variant Analysis and Mappability

Assignment Date: Wednesday, Feb. 17, 2020

Due Date: Wednesday, Mar. 3 , 2020 @ 11:59pm

Some of the tools you will need to use may only run in a linux or mac environment. If you do not have access to a linux/mac machine, download and install a virtual machine or ubuntu instance following the directions here:

<https://github.com/schatzlab/appliedgenomics2018/blob/master/assignments/virtualbox.md>

Alternatively, you might also want to try out this docker instance that has these tools preinstalled:

<https://github.com/mschatz/wga-essentials>

Assignment Overview

In this assignment you will take a look at mappability, get some experience with small variant analysis, and analyze some variant data.

For questions 1, 2 and 4 of this assignment, you will be using chromosome 22 from build 38 of the human genome – download it here: <http://hgdownload.cse.ucsc.edu/goldenPath/hg38/chromosomes/chr22.fa.gz>. Whenever we refer to chromosome 22 in this assignment, this is what you should use.

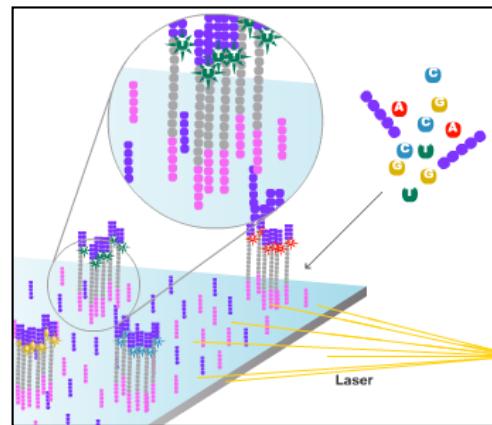
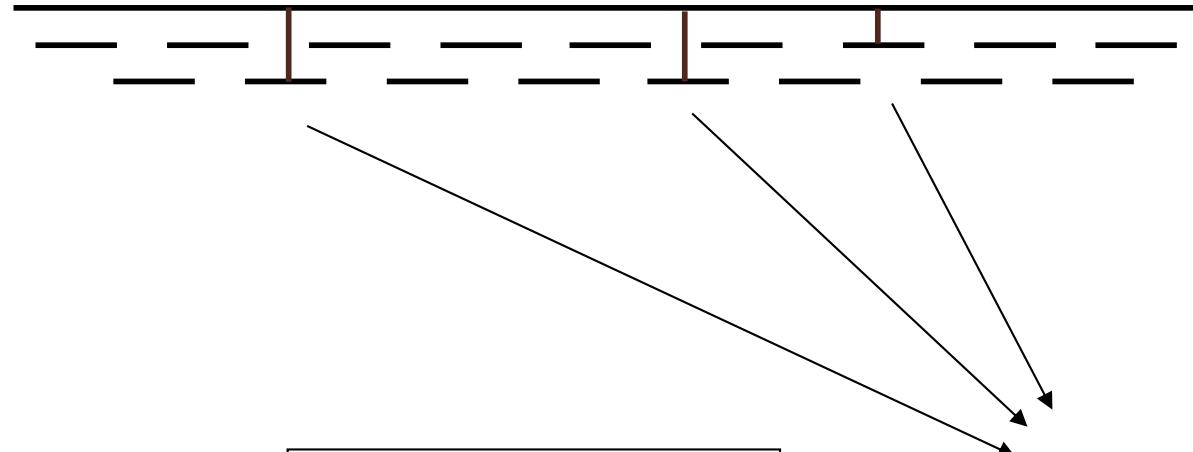
As a reminder, any questions about the assignment should be posted to [Piazza](#).

<https://github.com/schatzlab/appliedgenomics2021>

Read Mapping

Personal Genomics

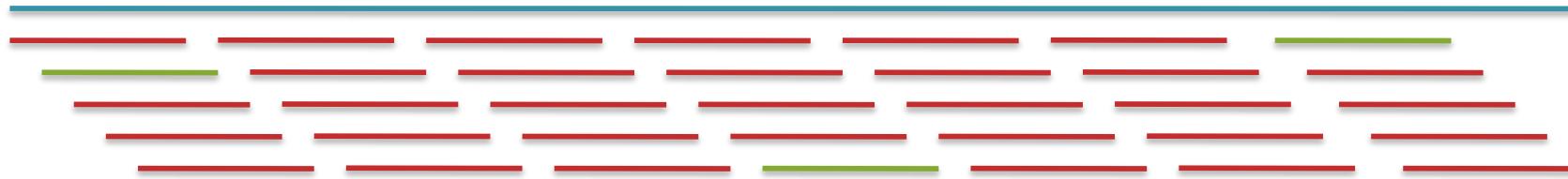
How does your genome compare to the reference?



Heart Disease
Cancer
Presidential smile

Three sets of horizontal lines representing DNA sequence alignments. The first set, labeled "Heart Disease", has three lines. The second set, labeled "Cancer", has four lines. The third set, labeled "Presidential smile", has three lines. Arrows from the text labels point to the corresponding sets of lines.

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?]

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?]

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?

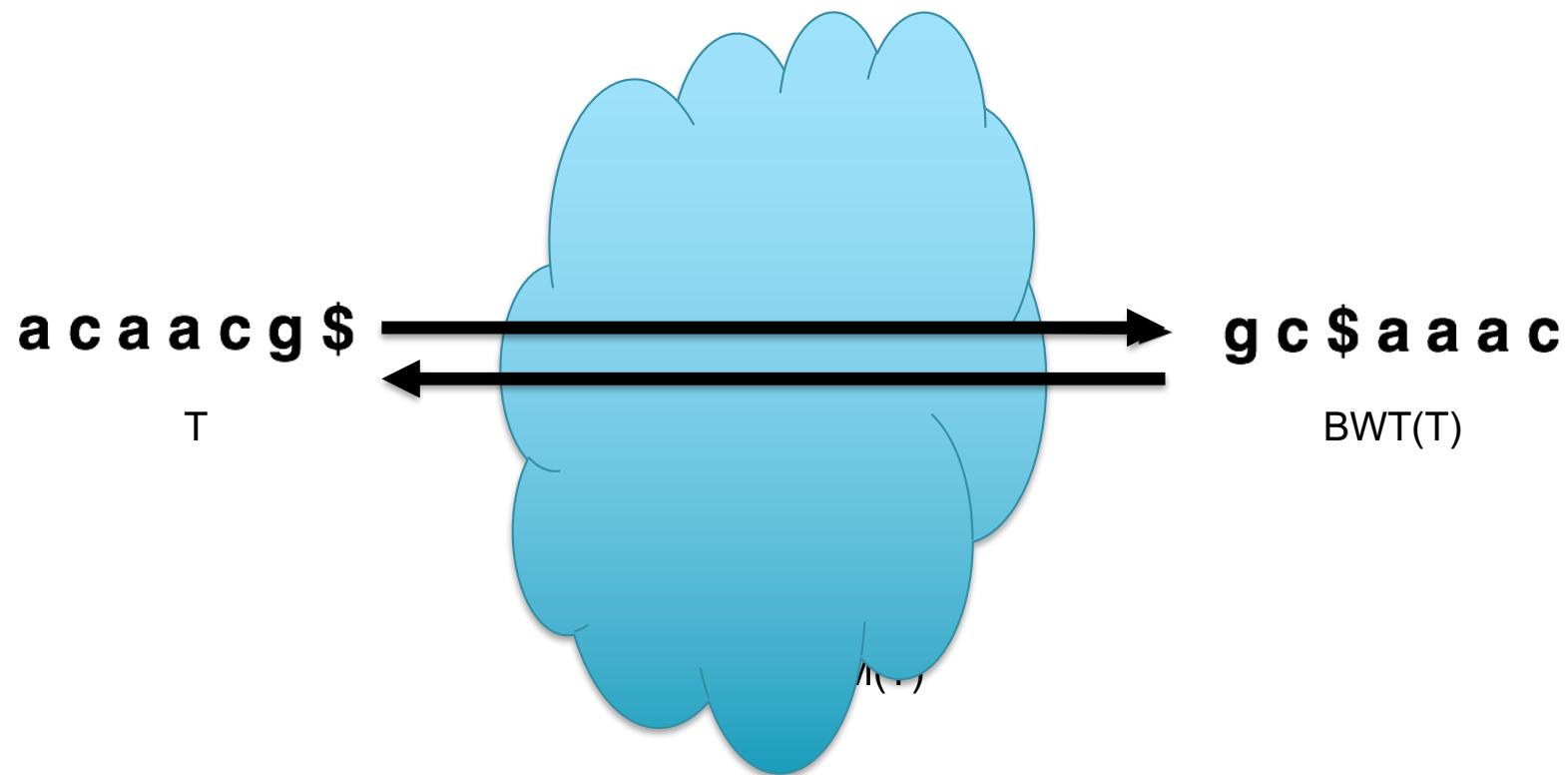


Bowtie: Ultrafast and memory efficient alignment of short DNA sequences to the human genome

Slides Courtesy of Ben Langmead

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

Burrows-Wheeler Transform

- Permutation of the characters in a text

a c a a c g \$ →

T

A block sorting lossless data compression algorithm.

Burrows M, Wheeler DJ (1994) *Digital Equipment Corporation. Technical Report 124*

Burrows-Wheeler Transform

- Permutation of the characters in a text

a c a a c g \$ →

T

a c a a c g \$	→	a c a a c g \$
		c a a c g \$ a
		a a c g \$ a c
		a c g \$ a c a
		c g \$ a c a a
		g \$ a c a a c
		\$ a c a a c g

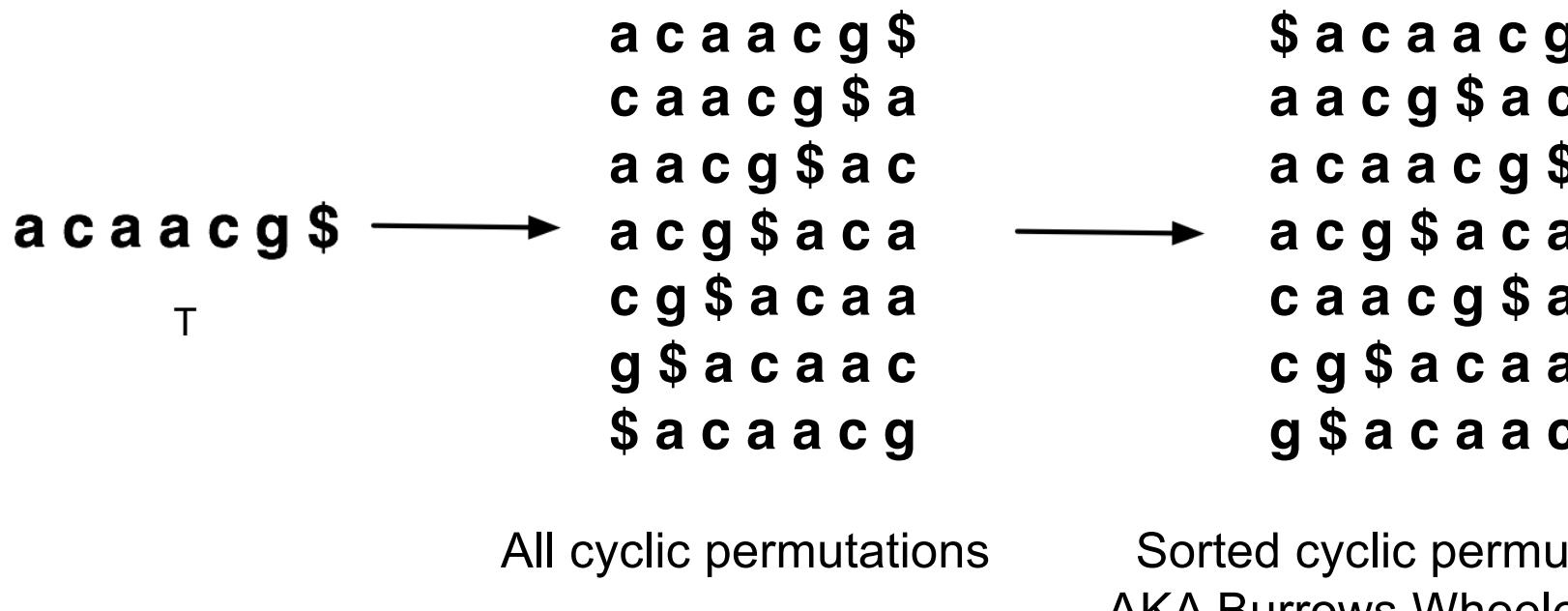
All cyclic permutations

A block sorting lossless data compression algorithm.

Burrows M, Wheeler DJ (1994) *Digital Equipment Corporation. Technical Report 124*

Burrows-Wheeler Transform

- 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*

Burrows-Wheeler Transform

- Permutation of the characters in a text

a c a a c g \$ →
T

\$ a c a a c g
a a c g \$ a c
a c a a c g \$
a c g \$ a c a
c a a c g \$ a
c g \$ a c a a
g \$ a c a a c

Sorted cyclic permutations
AKA Burrows Wheeler Matrix

A block sorting lossless data compression algorithm.

Burrows M, Wheeler DJ (1994) *Digital Equipment Corporation. Technical Report 124*

Burrows-Wheeler Transform

- Permutation of the characters in a text

a c a a c g \$ →
T

\$	a	c	a	a	c	g
a	a	c	g	\$	a	c
a	c	a	a	c	g	\$
a	c	g	\$	a	c	a
c	a	a	c	g	\$	a
c	g	\$	a	c	a	a
g	\$	a	c	a	a	c

Sorted cyclic permutations
AKA Burrows Wheeler Matrix

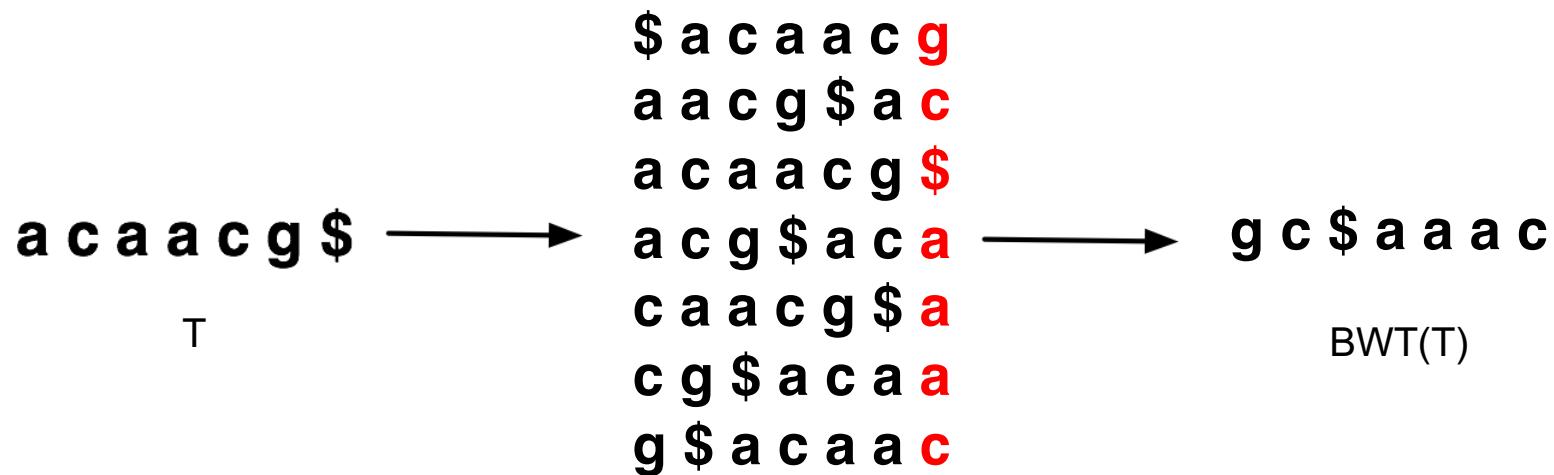
Last Column = Burrows Wheeler Transform

A block sorting lossless data compression algorithm.

Burrows M, Wheeler DJ (1994) *Digital Equipment Corporation. Technical Report 124*

Burrows-Wheeler Transform

- Permutation of the characters in a text



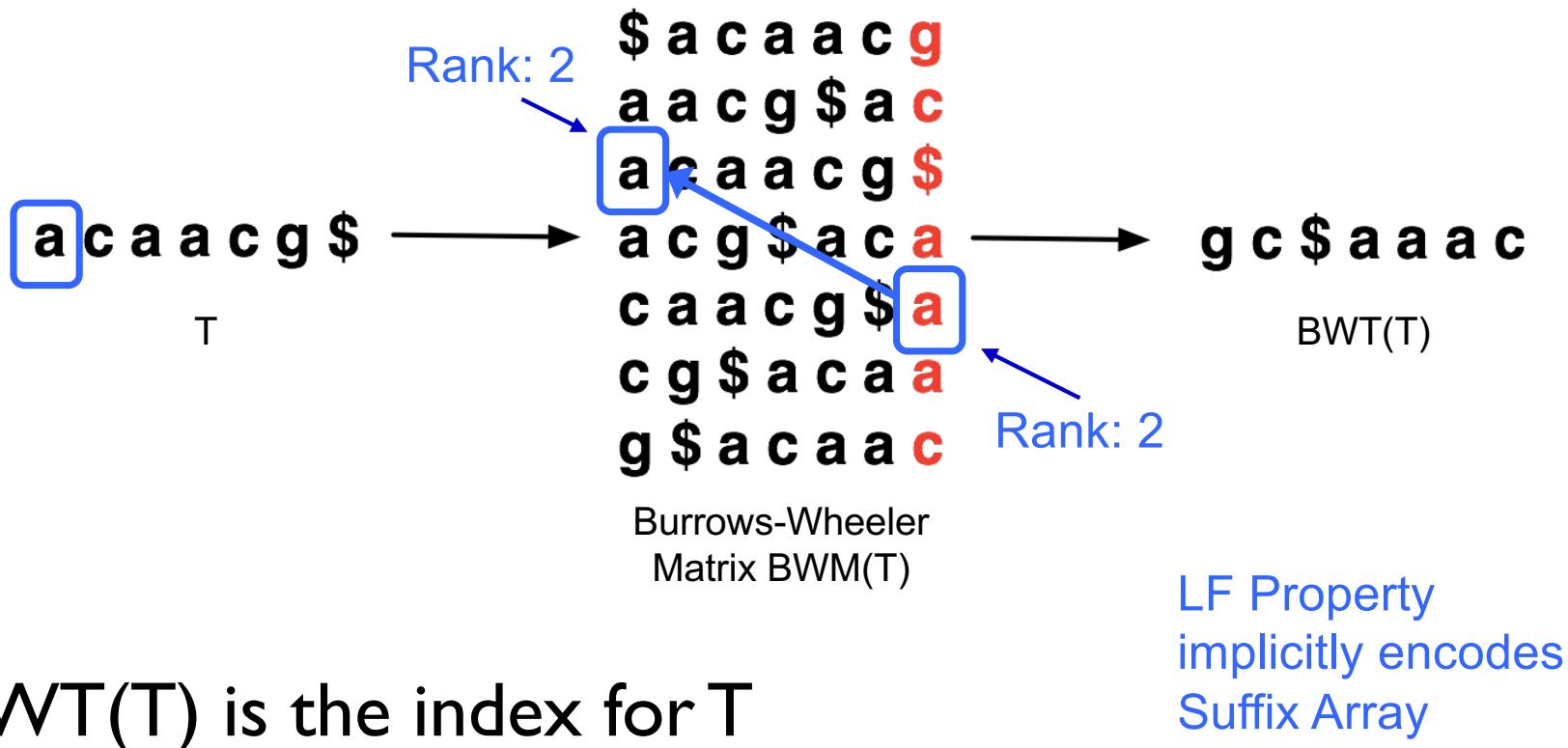
$\text{BWT}(T)$ is the index for T

A block sorting lossless data compression algorithm.

Burrows M, Wheeler DJ (1994) *Digital Equipment Corporation. Technical Report 124*

Burrows-Wheeler Transform

- Reversible permutation of the characters in a text



- $\text{BWT}(T)$ is the index for T

A block sorting lossless data compression algorithm.

Burrows M, Wheeler DJ (1994) Digital Equipment Corporation. Technical Report 124

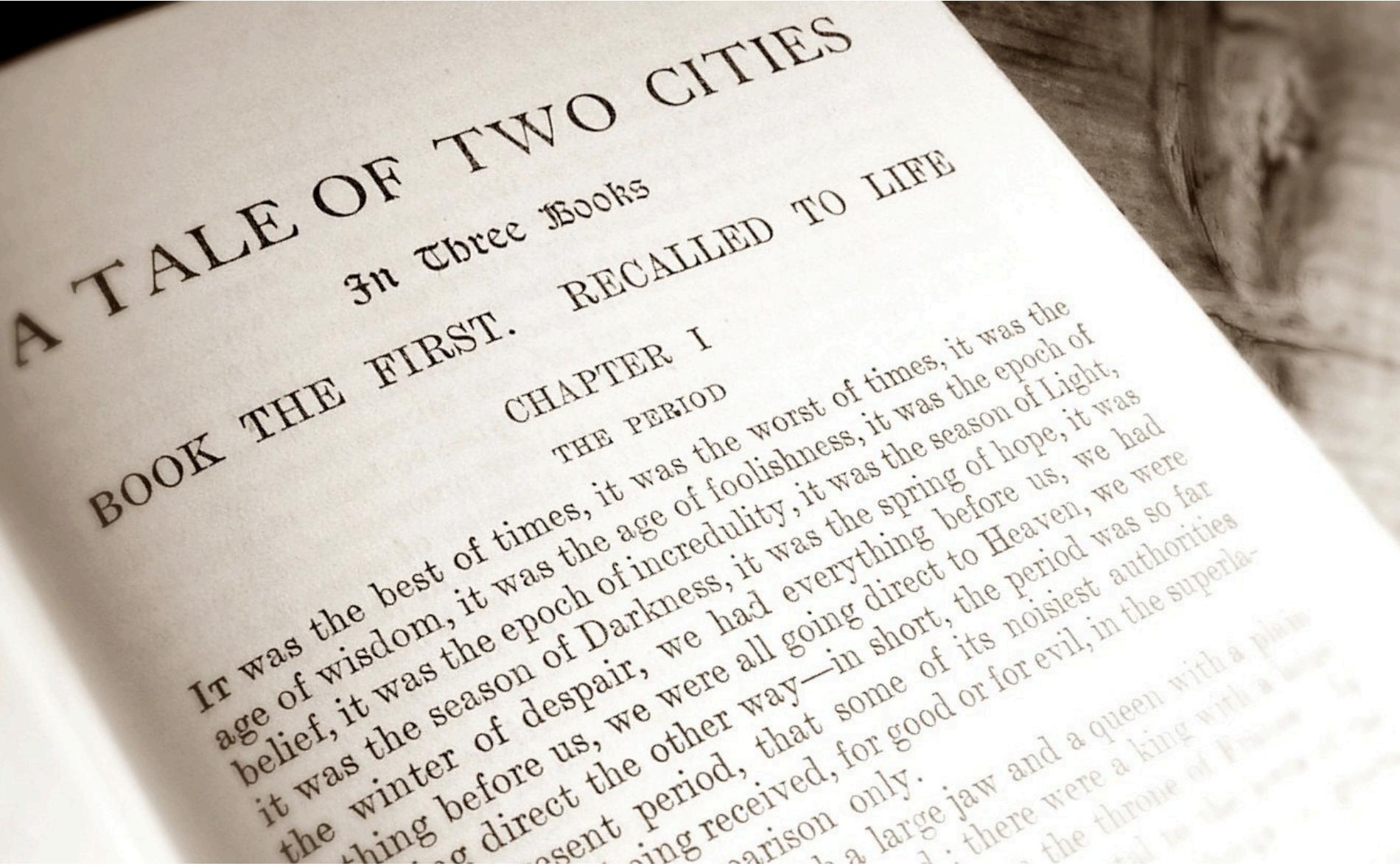
Burrows-Wheeler Transform

- Recreating T from $\text{BWT}(T)$
 - Start in the first row and apply **LF** repeatedly, accumulating predecessors along the way



[Decode this BWT string: ACTGA\$TTA]

Run Length Encoding



Run Length Encoding

`ref[614]:`

```
It_was_the_best_of_times,_it_was_the_worst_of_times,_it_was_the_age_
of_wisdom,_it_was_the_age_of_foolishness,_it_was_the_epoch_of_belief
,_it_was_the_epoch_of_incredulity,_it_was_the_season_of_Light,_it_wa
s_the_season_of_Darkness,_it_was_the_spring_of_hope,_it_was_the_wint
er_of_despair,_we_had_everything_before_us,_we_had_nothing_before_us
,_we_were_all_going_direct_to_Heaven,_we_were_all_going_direct_the_o
ther_way_-_in_short,_the_period_was_so_far_like_the_present_period,_
that_some_of_its_noisiest_authorities_insisted_on_its_being_received
,_for_good_or_for_evil,_in_the_superlative_degree_of_comparison_only.$
```

Run Length Encoding:

- Replace a “run” of a character X with a single X followed by the length of the run
- GAAAAAAAATTACA => GA8T2ACA (reverse is also easy to implement)
- If your text contains numbers, then you will need to use a (slightly) more sophisticated encoding

Run Length Encoding

```
ref[614]:
```

```
It_was_the_best_of_times,_it_was_the_worst_of_times,_it_was_the_age_
of_wisdom,_it_was_the_age_of_foolishness,_it_was_the_epoch_of_belief
,_it_was_the_epoch_of_incredulity,_it_was_the_season_of_Light,_it_wa
s_the_season_of_Darkness,_it_was_the_spring_of_hope,_it_was_the_wint
er_of_despair,_we_had_everything_before_us,_we_had_nothing_before_us
,_we_were_all_going_direct_to_Heaven,_we_were_all_going_direct_the_o
ther_way_-in_short,_the_period_was_so_far_like_the_present_period,_
that_some_of_its_noisiest_authorities_insisted_on_its_being_received
,_for_good_or_for_evil,_in_the_superlative_degree_of_comparison_only.$
```

```
rle(ref)[614]:
```

```
It_was_the_best_of_times,_it_was_the_worst_of_times,_it_was_the_age_
of_wisdom,_it_was_the_age_of_fo2lishnes2,_it_was_the_epoch_of_belief
,_it_was_the_epoch_of_incredulity,_it_was_the_season_of_Light,_it_wa
s_the_season_of_Darknes2,_it_was_the_spring_of_hope,_it_was_the_wint
er_of_despair,_we_had_everything_before_us,_we_had_nothing_before_us
,_we_were_al2_going_direct_to_Heaven,_we_were_al2_going_direct_the_o
ther_way_-in_short,_the_period_was_so_far_like_the_present_period,_
that_some_of_its_noisiest_authorities_insisted_on_its_being_received
,_for_go2d_or_for_evil,_in_the_superlative_degre2_of_comparison_only.$
```

Run Length Encoding

ref[614]:

```
It_was_the_best_of_times,_it_was_the_worst_of_times,_it_was_the_age_
of_wisdom,_it_was_the_age_of_foolishness,_it_was_the_epoch_of_belief
,_it_was_the_epoch_of_incredulity,_it_was_the_season_of_Light,_it_wa
s_the_season_of_Darkness,_it_was_the_spring_of_hope,_it_was_the_wint
er_of_despair,_we_had_everything_before_us,_we_had_nothing_before_us
,_we_were_all_going_direct_to_Heaven,_we_were_all_going_direct_the_o
ther_way_-in_short,_the_period_was_so_far_like_the_present_period,
_that_some_of_its_noisiestAuthorities_insisted_on_its_being_received
,_for_good_or_for_evil,_in_the_superlative_degree_of_comparison_only.$
```

bwt[614]:

```
.dlmssftysesdtrsns_y__$yfofeeetggsfefefggeedrofr,llreef-,fs,,,
,,nfrsdnnhereghettedndeteegreenstee,ssssst,esssnssffteedtttttttr,,,
,,eeeefehh_p_fpDwwwwwwwwwwweehl_ew_____eo_neeeoaaeoo____sephrrvh
hwwegmghhhhhhkkrrwwhssHrrrvtrribdbcbvs_thwwpppvmmirdnnib_eoooooo
oooooo____eennnnnnaai____ecc_ttts_tsgltsLlvtt____hhoor
e_wrraddwlors_____r_lteirillre_ouaanooioeooooiiihkiiiiiio_iei
tsppioi_____gnodsc_sss_gfhf_fffhwh_nsmo_uee_sioooaeeeeoo_ii
cgppeeaoaeooeesseutetaaaaaaaaai_ei_in_aaie_eeerei_hrsssnacciII
iiiiisn_____oyoui_a_iids_aiaee_____tlar
```

Run Length Encoding

ref[614]:

```
It_was_the_best_of_times,_it_was_the_worst_of_times,_it_was_the_age_
of_wisdom,_it_was_the_age_of_foolishness,_it_was_the_epoch_of_belief
,_it_was_the_epoch_of_incredulity,_it_was_the_season_of_Light,_it_wa
s_the_season_of_Darkness,_it_was_the_spring_of_hope,_it_was_the_wint
er_of_despair,_we_had_everything_before_us,_we_had_nothing_before_us
,_we_were_all_going_direct_to_Heaven,_we_were_all_going_direct_the_o
ther_way_-in_short,_the_period_was_so_far_like_the_present_period,
_that_some_of_its_noisiestAuthorities_insisted_on_its_being_received
,_for_good_or_for_evil,_in_the_superlative_degree_of_comparison_only.$
```

bwt[614]:

```
.dlmssftysesdtrsns_y__$yfofeeeetggsfefefggeedrofr,llreef-,fs,,,
,,nfrsdnnhereghettedndeteegreenstee,ssssst,esssnssffteedtttttttr,,,
,,eeeefehh_p_fpDwwwwwwwwwwweehl_ew_____eoo_neeeoaaeoo____sephrrvh
hwwegmghhhhhhkkrrwwhssHrrrvtrribdbcbvs_thwwpppvmmirdnnib_eoooooo
oooooooo____eennnnnnaai____ecc_tttttttttttts_tsgltsLlvtt__hhoor
e_wrraddwlors_____r_lteirillre_ouaanooioeooooiiihkiiiiio_iei
tsppioi_____gnodsc_sss_gfhf_fffhwh_nsmo_uee_sioooaeeeeoo_ii
cgppeeaoaeooeesseutetaaaaaaaaai_ei_in_aaie_eeerei_hrsssnacciII
iiiiisn_____ Why does the BWT tend to make runs in english text? _____tlar
```

Run Length Encoding

bwt[614]:

```
.dlmssftysesdtrsns_y$_yfofeeeetggsfefefggeedrofr,llreef-,fs,,,,,
,,nfrsdnnhereghettedndeteegreenstee,ssssst,esssnssffteedtttttttr.,
,,eefehh_p_fpDwwwwwwwwwwweehl_ew_____eo_neeeoaaeoo____sephrrvh
hwwegmghhhhhhkrrwwhssHrrrvtrribdbcbvs__thwwpppvmmirdnnib__oooooooo
oooooooo____eennnnnnaai____ecc__tttttttttttts_tsgltsLlvtt__hhoor
e_wrraddwlors_____r_lteirillre_ouaanooioeooooiiihkiiiiio_iei
tsppioi_____gnodsc_sss_gfhf_fffhwh_nsмо_uee_siоооаеееоо_ii
cgppeeaoaeooeesseutetaaaaaaaaai__ei_in_aaie_eeerei_hrsssnacciII
iiiiisn_____oyoui_a_iids_aiaee_____tlar
```

rle(bwt)[464]:

```
.dlms2ftysesdtrsns_y_2$yfofe4tg2sfefefg2e2drofr,l2re2f-,fs,9nfrsdn2
hereghet2edndete2ge2nste2,s5t,es3ns2f2te2dt10r,4e3feh2_2p_2fpDw11e2h
1_ew_5eo2_ne3oa2eo2_4seph2r2hvh2w2egmgh7kr2w2h2s2Hr3vtr2ib2dbcbvs_2t
hw2p3vm2irdn2ib_2eo12_4e2n6a2i_3ec2_2t18s_tsgltsLlv2_3h2o2re_wr2ad2
wlors_9r_2lteiril2re_oua2no2i2oeo4i3hki6o_2ieitsp2ioi_12g2nodsc_s3_g
fhf_f3hwh_nsмо_2ue2_siо3ae4o2_i2cgp2e2aoaeo2e2s2eu2tetallи_2ei_in_2a
2ie_e3rei_hrs3nac2i2Ii7sn_15oyoui_2a_i3ds_2ai2ae2_21tlar
```

Run Length Encoding

bwt[614]:

```
.dlmssftysesdtrsns_y__$yfofeeeetggsfefefggeedrofr,llreef-,fs,,,,,
,,nfrsdnnhereghettedndeteegreenstee,ssssst,esssnssffteedtttttttttr.,
,,eeefehh_p_fpDWWWWWWWWWWWWWeehl_ew_____eo_neeeoaaeoo_____sephrrvh
hwwegmghhhhhhkrrwwhssHrrrvtrribdbcbvs__thwwpppvmmirdnnib__oooooooo
oooooooo____eennnnnnaai____ecc__ttttttttttttts_tsgltsLlvtt____hhoor
e_wrraddwlors_____r_lteirillre_ouaanooioeooooiiihkiiiiio_iei
tsppioi_____gnodsc_sss_gfhf_fffhwh_nsмо_uee_siоооаеееоо_ii
cgppeeaoaeooeesseuutetaaaaaaaaai__ei_in_aaie_eeerei_hrsssnacciII
iiiiisn_____oyoui_a_iids_aiaee_____tlar
```

rle(bwt)[464]:

```
.dlms2ftysesdtrsns_y_2$yfofe4tg2sfefefg2e2drofr,l2re2f-,fs,9nfrsdn2
hereghet2edndete2ge2nste2,s5t,es3ns2f2te2dt10r,4e3feh2_2p_2fpDw11e2h
1_ew_5eo2_ne3oa2eo2_4seph2r2hvh2w2egmgh7kr2w2h2s2Hr3vtr2ib2dbcbvs_2t
hw2p3vm2irdn2ib_2eo12_4e2n6a2i_3ec2_2t18s_tsgltsLlv2_3h2o2re_wr2ad2
wlors_9r_2lteiril2re_oua2no2i2oeo4i3hki6o_2ieitsp2ioi_12g2nodsc_s3_g
fhf_f3hwh_nsмо_2ue2_siо3ae4o2_i2cgp2e2aoaeo2e2s2eu2tet11i_2ei_in_2a
2ie_e3rei_hrs3nac2i2Ii7sn_15oyoui_2a_i3ds_2ai2ae2_21tlar
```

Run Length Encoding

ref[614]:

```
It_was_the_best_of_times,_it_was_the_worst_of_times,_it_was_the_age_
of_wisdom,_it_was_the_age_of_foolishness,_it_was_the_epoch_of_belief
,_it_was_the_epoch_of_incredulity,_it_was_the_season_of_Light,_it_wa
s_the_season_of_Darkness,_it_was_the_spring_of_hope,_it_was_the_wint
er_of_despair,_we_had_everything_before_us,_we_had_nothing_before_us
,_we_were_all_going_direct_to_Heaven,_we_were_all_going_direct_the_o
ther_way_-in_short,_the_period_was_so_far_like_the_present_period,
_that_some_of_its_noisiestAuthorities_insisted_on_its_being_received
,_for_good_or_for_evil,_in_the_superlative_degree_of_comparison_only.$
```

rle(bwt)[464]:

```
.dlms2ftysesdtrsns_y_2$_yfofe4tg2sfefefg2e2drofr,l2re2f-,fs,9nfrsdn2
hereghet2edndete2ge2nste2,s5t,es3ns2f2te2dt10r,4e3feh2_2p_2fpDw11e2h
l_ew_5eo2_ne3oa2eo2_4seph2r2hvh2w2egmgh7kr2w2h2s2Hr3vtr2ib2dbcbvs_2t
hw2p3vm2irdn2ib_2eo12_4e2n6a2i_3ec2_2t18s_tsgltsLlvt2_3h2o2re_wr2ad2
wlors_9r_2lteiril2re_oua2no2i2oeo4i3hki6o_2ieitsp2ioi_12g2nodsc_s3_g
fhf_f3hwh_nsmo_2ue2_sio3ae4o2_i2cgp2e2aoaeo2e2s2eu2tet11i_2ei_in_2a
2ie_e3rei_hrs3nac2i2Ii7sn_15oyoui_2a_i3ds_2ai2ae2_21tlar
```

Run Length Encoding

ref[614]:

```
It_was_the_best_of_times,_it_was_the_worst_of_times,_it_was_the_age_
of_wisdom,_it_was_the_age_of_foolishness,_it_was_the_epoch_of_belief
,_it_was_the_epoch_of_incredulity,_it_was_the_season_of_Light,_it_wa
s_the_season_of_Darkness,_it_was_the_spring_of_hope,_it_was_the_wint
er_of_despair,_we_had_everything_before_us,_we_had_nothing_before_us
,_we_were_all_going_direct_to_Heaven,_we_were_all_going_direct_the_o
ther_way_-in_short,_the_period_was_so_far_like_the_present_period,
_that_some_of_its_noisiestAuthorities_insisted_on_its_being_received
,_for_good_or_for_evil,_in_the_superlative_degree_of_comparison_only.$
```

rle(bwt)[464]:

```
.dlms2ftysesdtrsns_y_2$_yfofe4tg2sfefefg2e2drofr,l2re2f-,fs,9nfrsdn2
hereghet2edndete2ge2nste2,s5t,es3ns2f2te2dt10r,4e3feh2_2p_2fpDw11e2h
l_ew_5eo2_ne3oa2eo2_4seph2r2hvh2w2egmgh7kr2w2h2s2Hr3vtr2ib2dbcbvs_2t
hw2p3vm2irdn2ib_2eo12_4e2n6a2i_3ec2_2t18s_tsgltsLlvt2_3h2o2re_wr2ad2
wlors_9r_2lteiril2re_oua2no2i2oeo4i3hki6o_2ieitsp2ioi_12g2nodsc_s3_g
fhf_f3hwh_nsмо_2ue2_sio3ae4o2_i2cgp2e2aoaeo2e2s2eu2tet11i_2ei_in_2a
2ie_e3rei
```

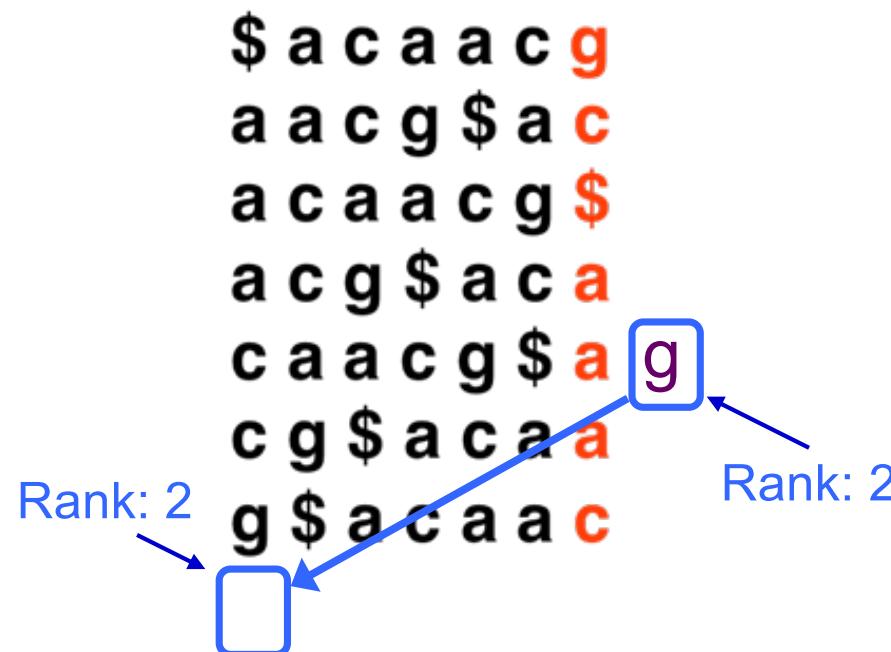
Saved 614-464 = 150 bytes (24%) with zero loss of information!

Common to save 50% to 90% on real world files with bzip2

BWT Exact Matching

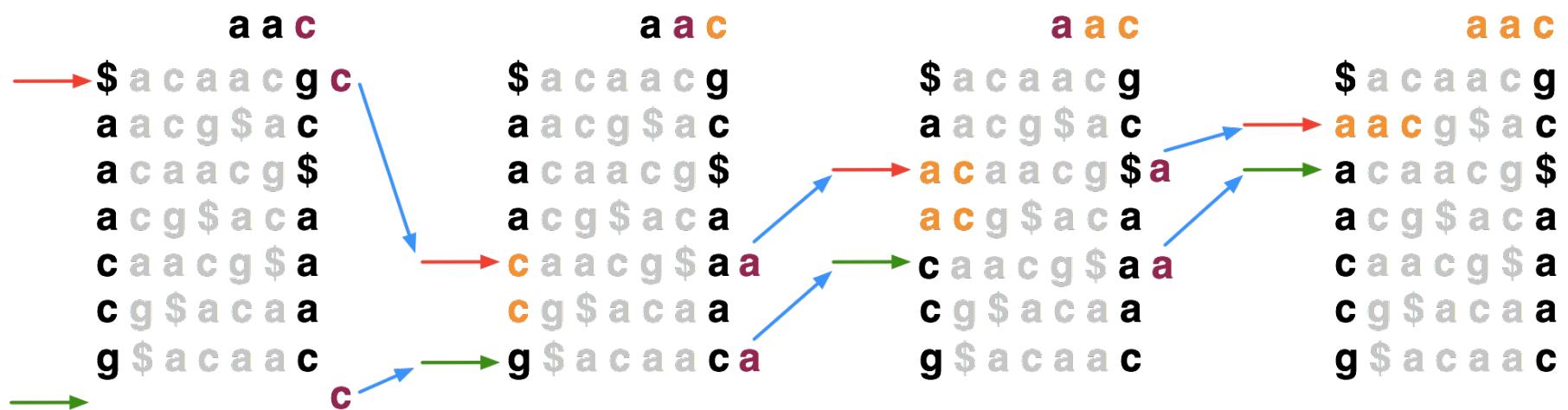
- $\text{LF}_c(r, c)$ does the same thing as $\text{LF}(r)$ but it ignores r 's actual final character and “pretends” it's c :

$$\text{LF}_c(5, g) = 8$$



BWT Exact Matching

- Start with a range, (**top**, **bot**) encompassing all rows and repeatedly apply **LFc**:
top = **LFc**(**top**, **qc**); **bot** = **LFc**(**bot**, **qc**)
qc = the next character to the left in the query



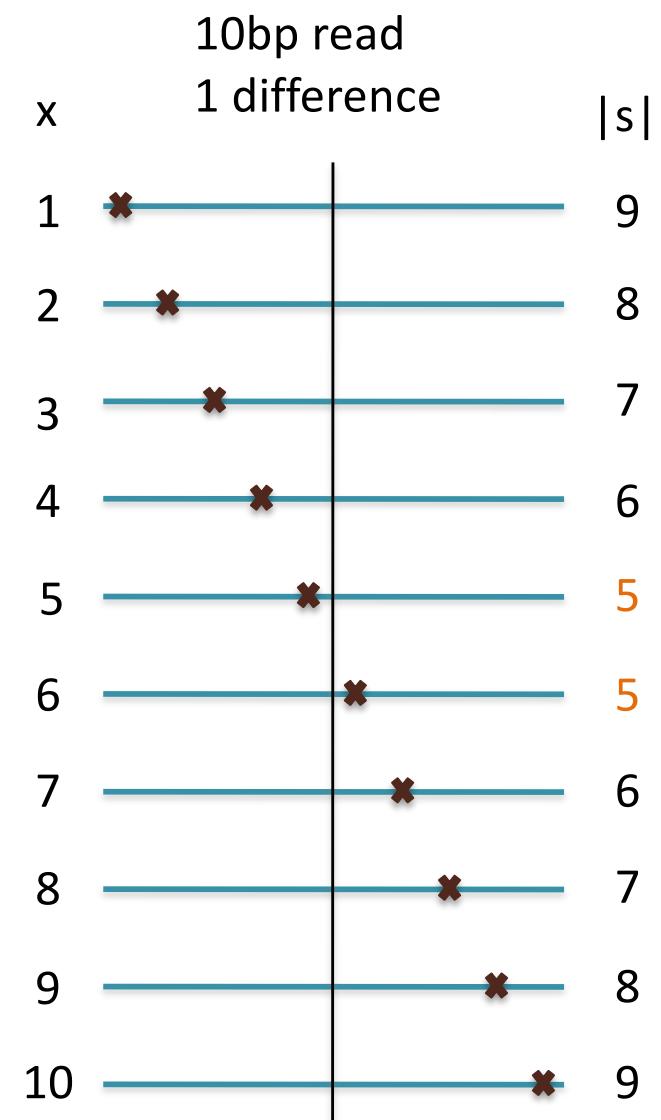
Ferragina P, Manzini G: Opportunistic data structures with applications. FOCS. IEEE Computer Society; 2000.

[Search for TTA this BWT string: ACTGA\$TTA]

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



Algorithm Overview

1. Split read into segments

Read (reverse complement)

CCAGTAGCTCTCAGCCTTATTTACCCAGGCCTGTA **TACAGGCCTGGGTAAAATAAGGCTGAGAGCTACTGG**

Policy: extract 16 nt seed every 10 nt

Seeds

+ , 0: CCAGTAGCTCTCAGCC	- , 0: TACAGGCCTGGGTAAA
+ , 10: TCAGCCTTATTTTACC	- , 10: GGTAAAATAAGGCTGA
+ , 20: TTTACCCAGGCCTGTA	- , 20: GGCTGAGAGCTACTGG

2. Lookup each segment and prioritize

The diagram illustrates the workflow for aligning DNA sequences. It starts with a list of **Seeds** on the left, which are then processed to create an **Ungapped alignment with FM Index** in the middle. This intermediate step is shown in a rounded rectangle with a black border. Finally, the **Seed alignments (as B ranges)** are derived from the aligned sequence.

Seeds	Ungapped alignment with FM Index	Seed alignments (as B ranges)
+, 0: CCAGTAGCTCTCAGCC	[Sequence: a a c \$ a c a a c g a a c g \$ a c a c a a c \$ a c g \$ a c a [c c - - a [c c - - a g \$ a c a a c]	{ [211, 212], [212, 214] }
+, 10: TCAGCCTTATTTIACC		{ [653, 654], [651, 653] }
+, 20: TTTACCCCAGGCCTGTA		{ [684, 685] }
-, 0: TACAGGCCTGGGTAAA		{ }
-, 10: GGTAAAATAAAGGCTGA		{ }
-, 20: GGCTGAGAGCTACTGG		{ [624, 625] }

3. Evaluate end-to-end match

The diagram illustrates a bioinformatics pipeline. On the left, a list of "Extension candidates" is shown in green and blue text:

- SA:684, chr12:1955**
- SA:624, chr2:462**
- SA:211: chr4:762**
- SA:213: chr12:1935**
- SA:652: chr12:1945**

An arrow points from these candidates to a central box labeled "SIMD dynamic programming aligner". Inside this box is a small grid visualization representing sequence alignment.

An arrow points from the aligner box to the right, leading to "SAM alignments" in green text:

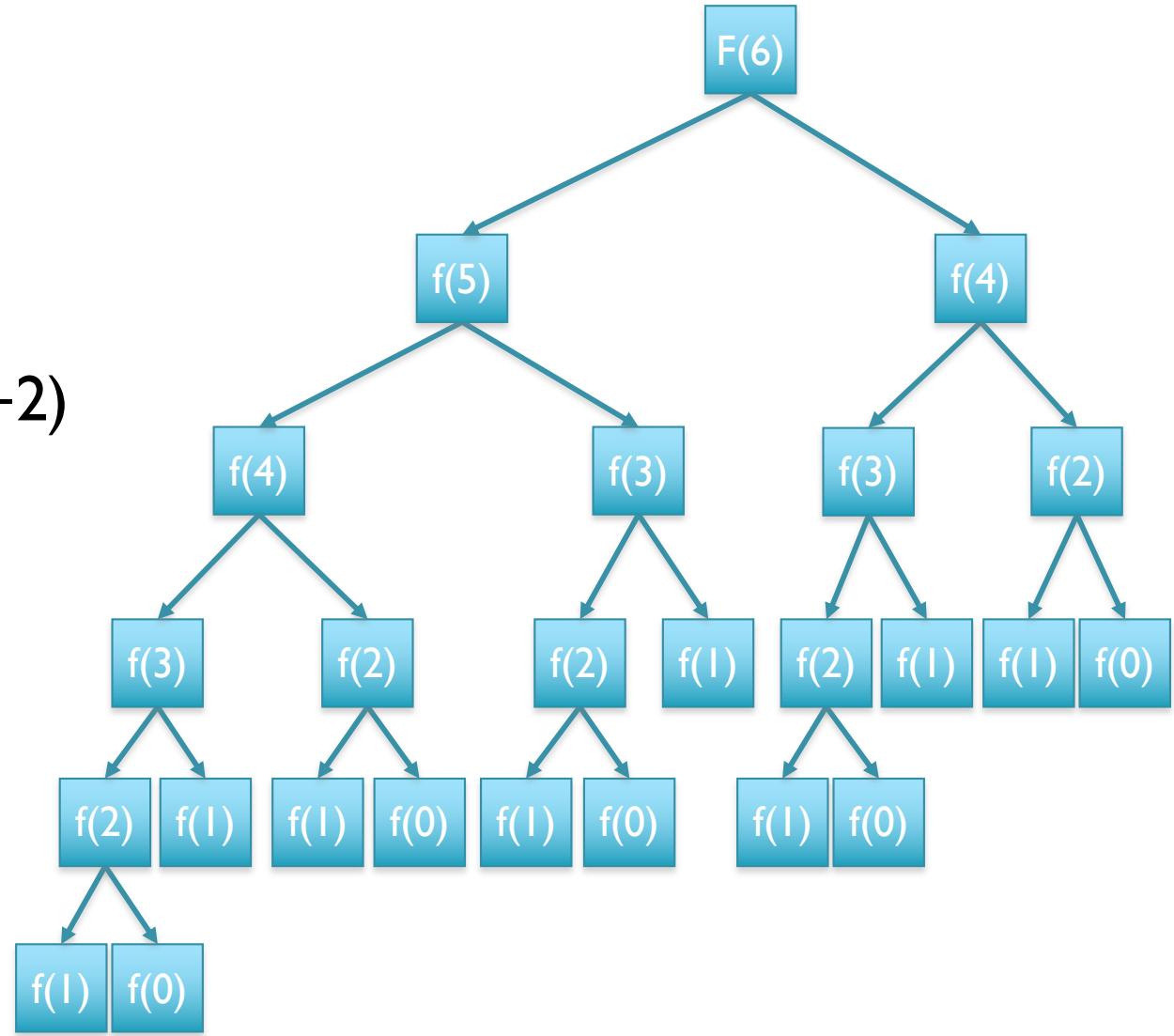
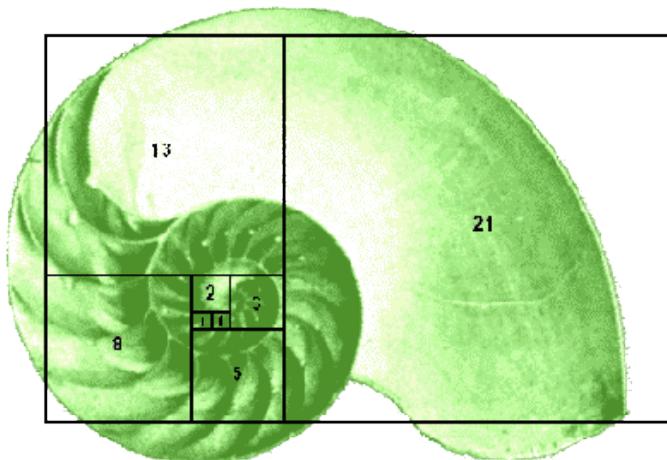
r1	0	chr12	1936	0
36M	*	0	0	
CCAGTAGCTCTAGCCTTATTTACCCAGGCCTGTA				
II				
AS:i:0	XS:i:-2	XN:i:0		
XM:i:0	XO:i:0	XG:i:0		
NM:i:0	MD:Z:36	YT:Z:UU		
YM:i:0				

Ellipses (...) are shown below the NM row, indicating more data follows. A final bracket on the right groups the SAM alignments and the text "(Langmead & S...)"

Part 2: Dynamic Programming

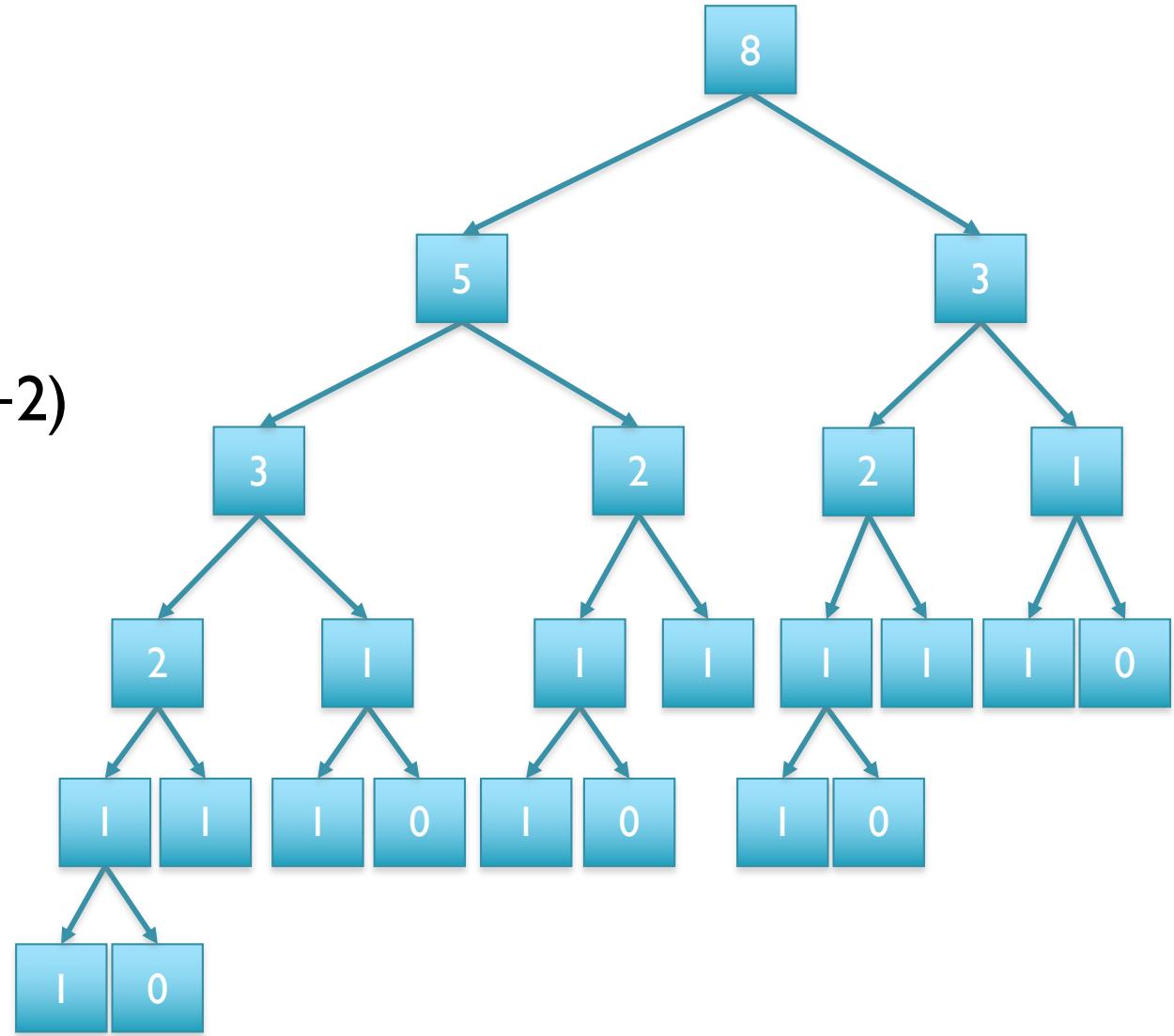
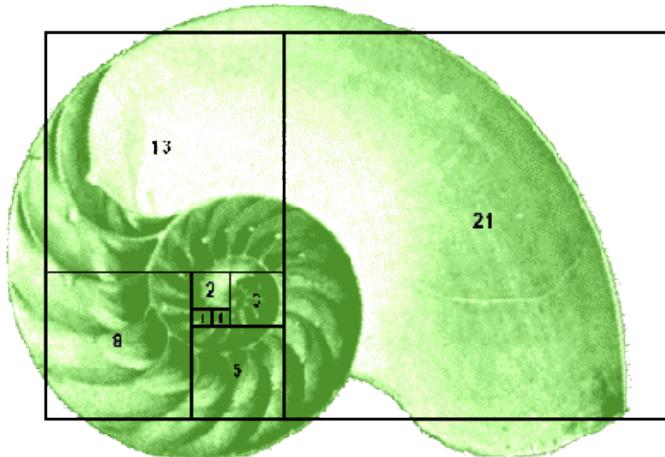
Fibonacci Sequence

```
def fib(n):
    if n == 0 or n == 1:
        return n
    else:
        return fib(n-1) + fib(n-2)
```



Fibonacci Sequence

```
def fib(n):  
    if n == 0 or n == 1:  
        return n  
  
    else:  
        return fib(n-1) + fib(n-2)
```



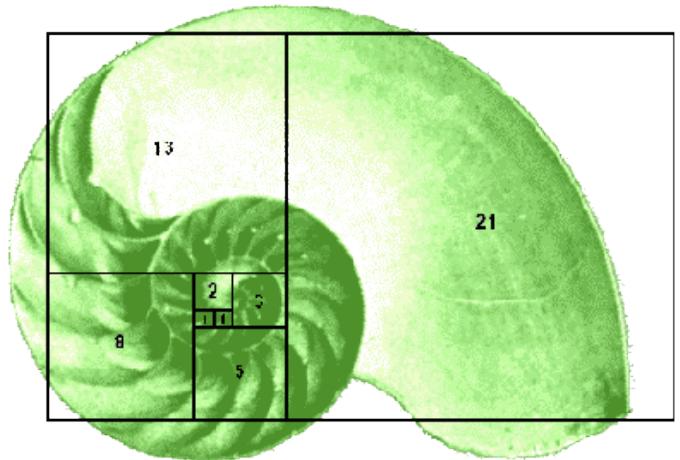
[What is the running time?]

Bottom-up Fibonacci Sequence

```
def fib(n):  
    table = [0] * (n+1)  
    table[0] = 0  
    table[1] = 1  
    for i in range(2,n+1):  
        table[i] = table[i-2] + table[i-1]  
    return table[n]
```

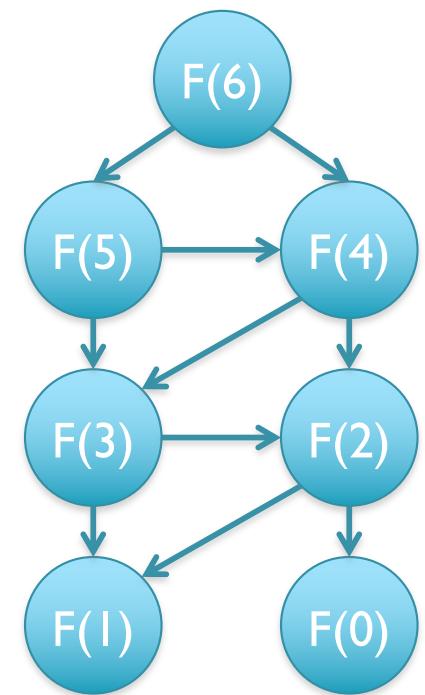
0	1	2	3	4	5	6
0	1	1	2	3	5	8

[What is the running time?]



Dynamic Programming

- General approach for solving (some) complex problems
 - When applicable, the method takes far less time than naive methods.
 - Polynomial time ($O(n)$ or $O(n^2)$) instead of exponential time ($O(2^n)$ or $O(3^n)$)
- Requirements:
 - Overlapping subproblems
 - Optimal substructure
- Applications:
 - Fibonacci
 - Longest Increasing Subsequence (Bonus Slides!)
 - Sequence alignment, Dynamic Time Warp, Viterbi
- Not applicable:
 - Traveling salesman problem, Clique finding, Subgraph isomorphism, ...
 - The cheapest flight from airport A to airport B involves a single connection through airport C, but the cheapest flight from airport A to airport C involves a connection through some other airport D.



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

Similarity metrics

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

MIKESCHATZ
| | x | | xxxx |
MICESHATZZ
5

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

MIKESCHAT-Z
| | x | | x | | | x |
MICES-HATZZ

Edit Distance Example

AGCACACACA → ACACACTA in 4 steps

AGCACACACA → (1. change G to C)

ACCACACACA → (2. delete C)

ACACACACA → (3. change A to T)

ACACACTT → (4. insert A after T)

ACACACTA → done

[Is this the best we can do?]

Edit Distance Example

AGCACACCA → ACACACTA in 3 steps

AGCACACCA → (1. change G to C)

ACCACACCA → (2. delete C)

ACACACCA → (3. insert T after 3rd C)

ACACACTA → done

[Is this the best we can do?]

Reverse Engineering Edit Distance

$$D(\text{AGCACACA}, \text{ACACACTA}) = ?$$

Imagine we already have the optimal alignment of the strings, the last column can only be 1 of 3 options:

... M	... I	... D
...A	...-	...A
...A	...A	...-

The optimal alignment of last two columns is then 1 of 9 possibilities

... MM	... IM	... DM	... MI	... II	... DI	... MD	... ID	... DD
...CA	...-A	...CA	...A-	...--	...A-	...CA	...-A	...CA
...TA	...TA	...-A	...TA	...TA	...-A	...A-	...A-	...--

The optimal alignment of the last three columns is then 1 of 27 possibilities...

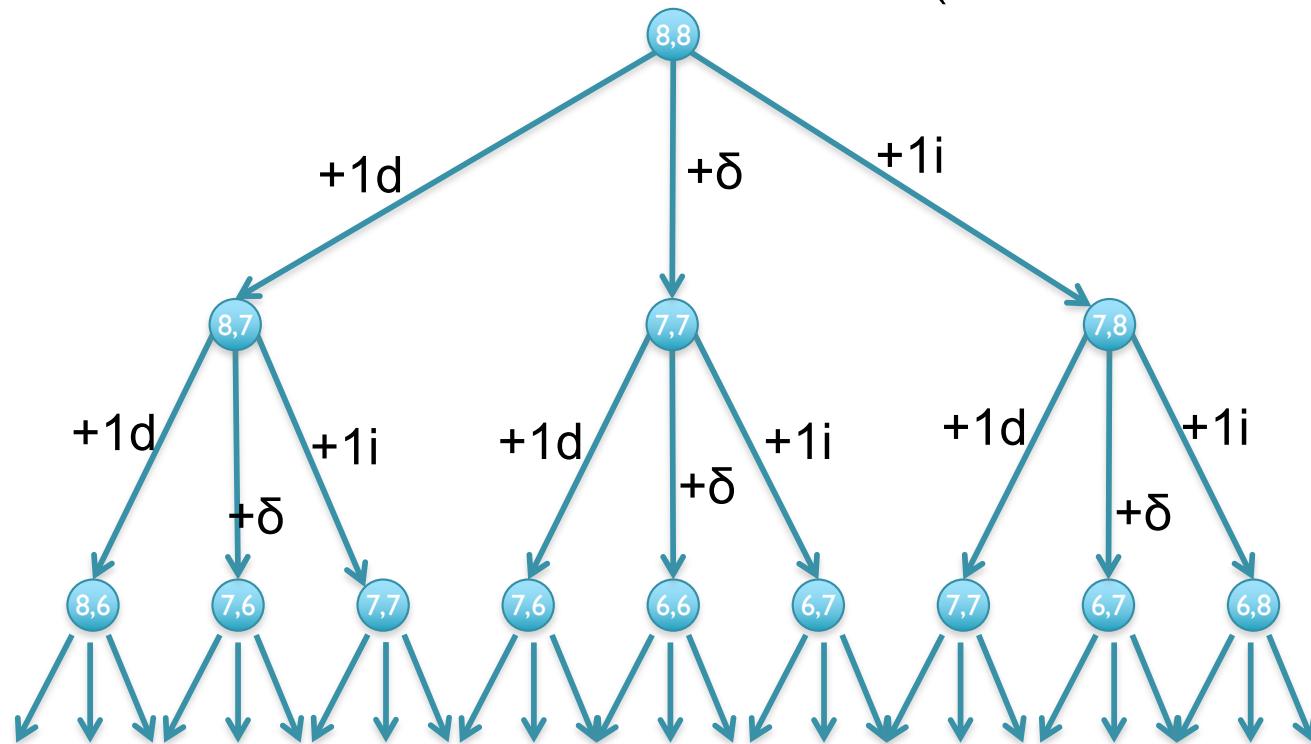
... M I D ...
...X...	...-...	...X...
...Y...	...Y...	...-...

Eventually spell out every possible sequence of {I,M,D}

Recursive solution

- Computation of D is a recursive process.
 - At each step, we only allow matches, substitutions, and indels
 - $D(i,j)$ in terms of $D(i',j')$ for $i' \leq i$ and $j' \leq j$.

$$D(\text{AGCACACA}, \text{ACACACTA}) = \min\{D(\text{AGCACACA}, \text{ACACACT}) + 1, \\ D(\text{AGCACAC}, \text{ACACACTA}) + 1, \\ D(\text{AGCACAC}, \text{ACACACT}) + \delta(A, A)\}$$



[What is the running time?]

Dynamic Programming

- We could code this as a recursive function call...
...with an exponential number of function evaluations
- There are only $(n+1) \times (m+1)$ pairs i and j
 - We are evaluating $D(i,j)$ multiple times
- Compute $D(i,j)$ bottom up.
 - Start with smallest $(i,j) = (1,1)$.
 - Store the intermediate results in a table.
 - Compute $D(i,j)$ after $D(i-1,j)$, $D(i,j-1)$, and $D(i-1,j-1)$

Recurrence Relation for D

Find the edit distance (minimum number of operations to convert one string into another) in $O(mn)$ time

- Base conditions:

- $D(i,0) = i$, for all $i = 0, \dots, n$
- $D(0,j) = j$, for all $j = 0, \dots, m$

- For $i > 0, j > 0$:

$$D(i,j) = \min \{ \begin{aligned} & D(i-1,j) + 1, && // \text{align 0 chars from S, 1 from T} \\ & D(i,j-1) + 1, && // \text{align 1 chars from S, 0 from T} \\ & D(i-1,j-1) + \delta(S(i),T(j)) && // \text{align } i+1 \text{ chars} \end{aligned} \}$$

[Why do we want the min?]

Dynamic Programming Matrix

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

[What does the initialization mean?]

Dynamic Programming Matrix

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

$$D[A, A] = \min\{D[A,] + 1, D[, A] + 1, D[,] + \delta(A, A)\}$$

Dynamic Programming Matrix

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

$$D[A, AC] = \min\{D[A, A]+1, D[AC]+1, D[A]+\delta(A, C)\}$$

Dynamic Programming Matrix

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

$$D[A,ACA] = \min\{D[A,AC]+1, D[,ACA]+1, D[,AC]+\delta(A,A)\}$$

Dynamic Programming Matrix

		A	C	A	C	A	C	T	A
	<u>0</u>	<u>1</u>	<u>2</u>	<u>3</u>	<u>4</u>	<u>5</u>	<u>6</u>	<u>7</u>	<u>8</u>
A	I	0	I	2	3	4	5	6	7
G	2								
C	3								
A	4								
C	5								
A	6								
C	7								
A	8								

$$D[A, ACACACTA] = 7$$

-----A

***** |

ACACACTA

[What about the other A?]

Dynamic Programming Matrix

		A	C	A	C	A	C	T	A
	<u>0</u>	<u>1</u>	<u>2</u>	<u>3</u>	<u>4</u>	<u>5</u>	<u>6</u>	<u>7</u>	<u>8</u>
A	I	0	I	2	3	4	5	6	7
G	2	I	I	2	3	4	5	6	7
C	3								
A	4								
C	5								
A	6								
C	7								
A	8								

$$D[AG, ACACACTA] = 7$$

----AG--

*** | ***

ACACACTA

Dynamic Programming Matrix

		A	C	A	C	A	C	T	A
	<u>0</u>	1	2	3	4	5	6	7	8
A	1	<u>0</u>	1	2	3	4	5	6	7
G	2	<u>1</u>	1	2	3	4	5	6	7
C	3	2	<u>1</u>	2	2	3	4	5	6
A	4	3	2	<u>1</u>	2	2	3	4	5
C	5	4	3	2	<u>1</u>	2	2	3	4
A	6	5	4	3	2	<u>1</u>	2	3	3
C	7	6	5	4	3	2	<u>1</u>	<u>2</u>	3
A	8	7	6	5	4	3	2	2	<u>2</u>

$$D[AGCACACA, ACACACTA] = 2$$

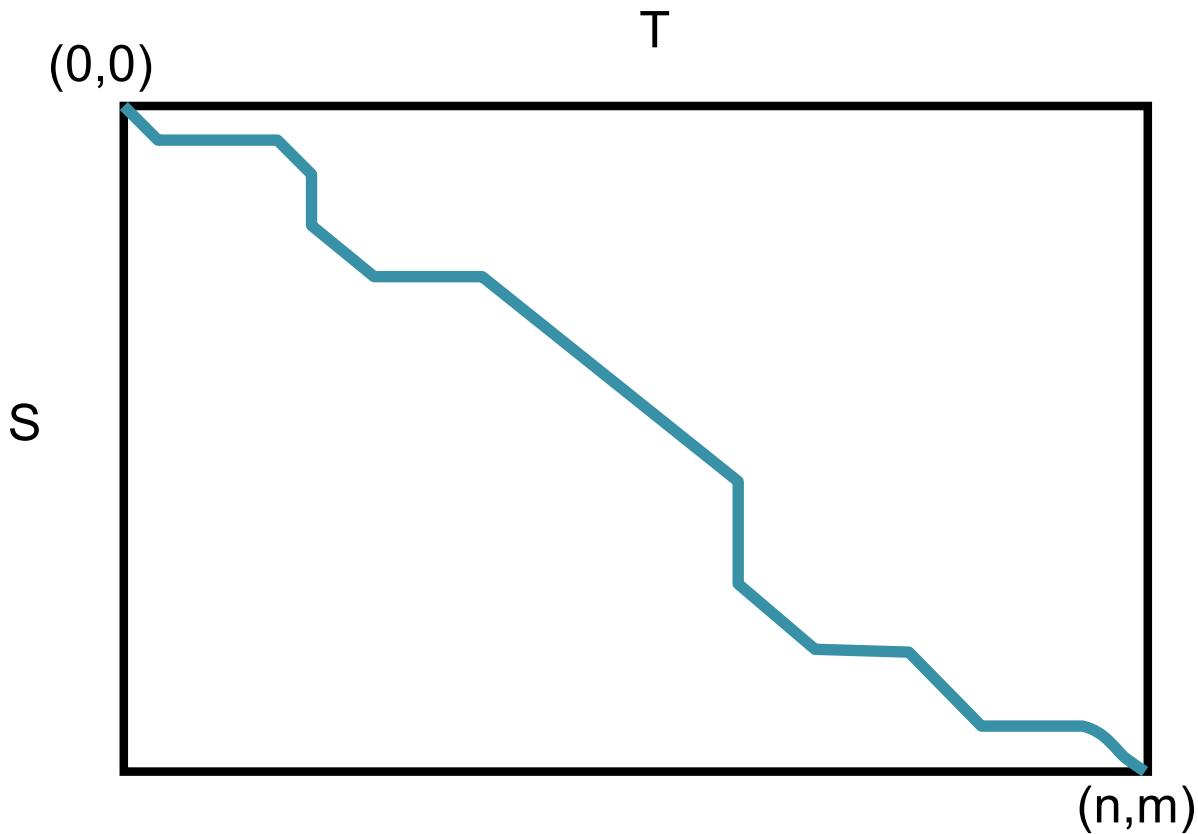
AGCACAC-A

| * | | | | * |

A-CACACTA

[Can we do it any better?]

Global Alignment Schematic

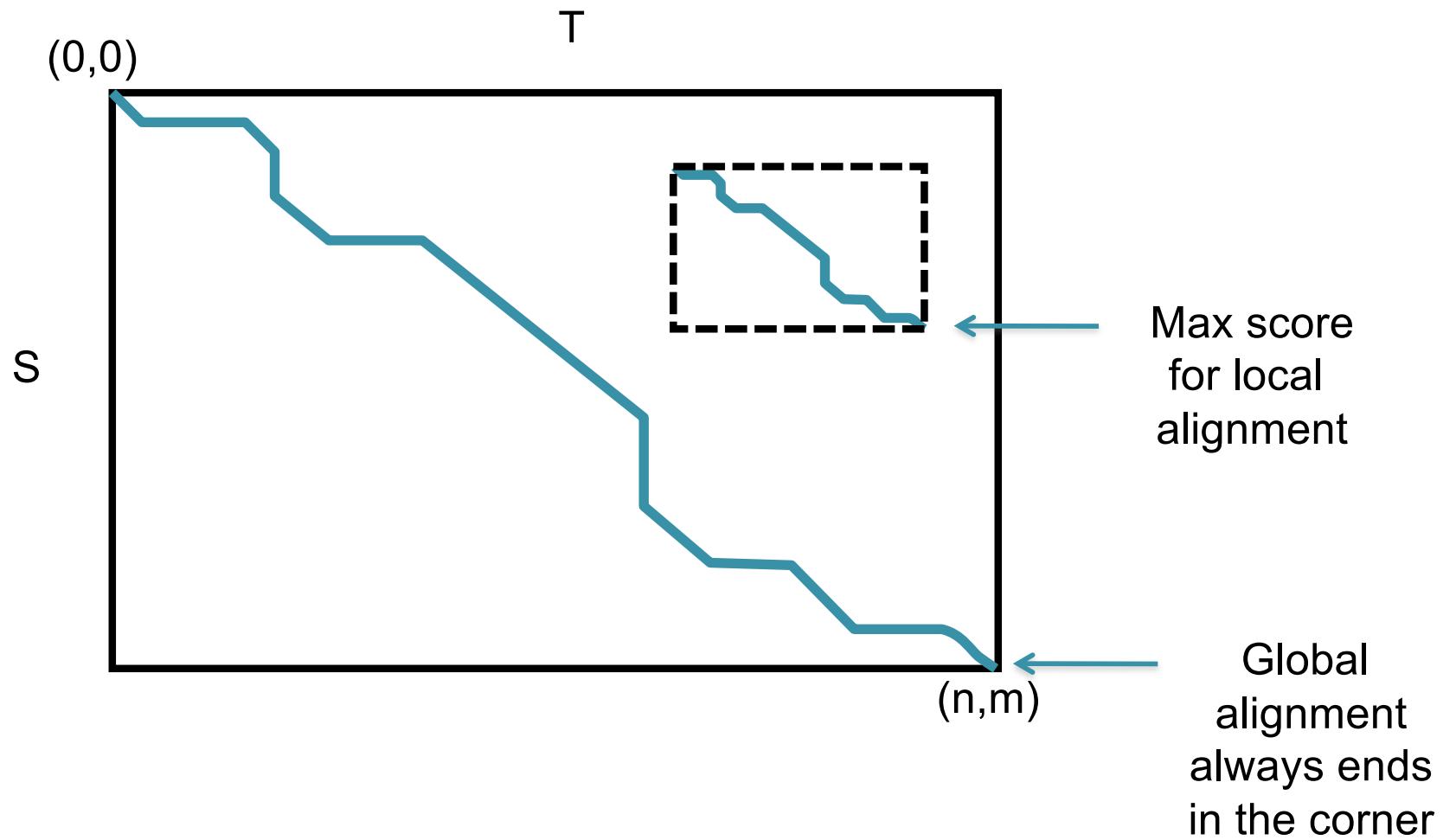


- A high quality alignment will stay close to the diagonal
 - If we are only interested in high quality alignments, we can skip filling in cells that can't possibly lead to a high quality alignment
 - Find the global alignment with at most edit distance d : $O(2dn)$

Local vs. Global Alignment

- The Global Alignment Problem tries to find the best end-to-end alignment between the two strings
 - Only applicable for very closely related sequences
- The Local Alignment Problem tries to find pairs of **substrings** with highest similarity.
 - Especially important if one string is substantially longer than the other
 - Especially important if there is only a distant evolutionary relationship

Global vs Local Alignment Schematic



Local vs. Global Alignment (cont' d)

- **Global Alignment**

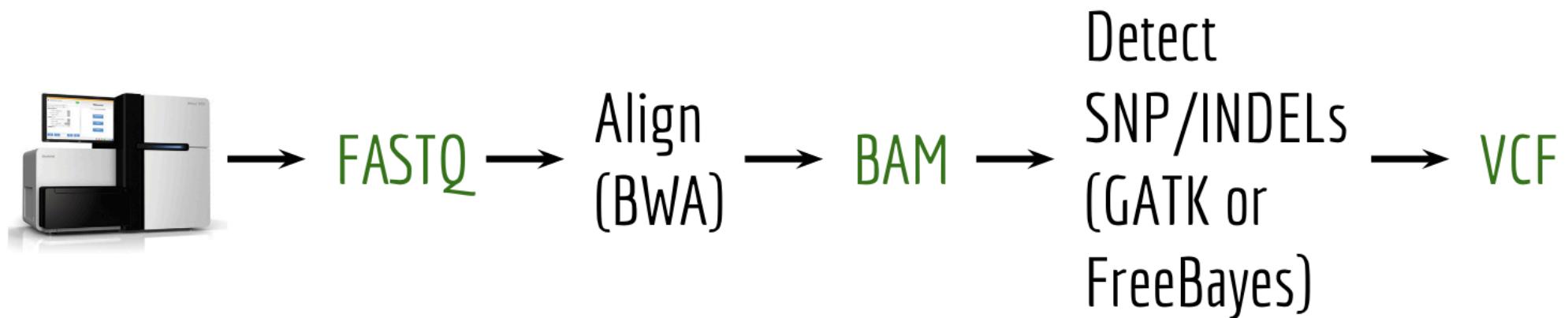
```
--T---CC-C-AGT--TATGT-CAGGGGACACG-A-GCATGCAGA-GAC  
| | | | | | | | | | | | | | | | | | | | | | | | | | |  
AATTGCCGCC-GTCGT-T-TTCAG----CA-GTTATG-T-CAGAT--C
```

- **Local Alignment**—better alignment to find conserved segment

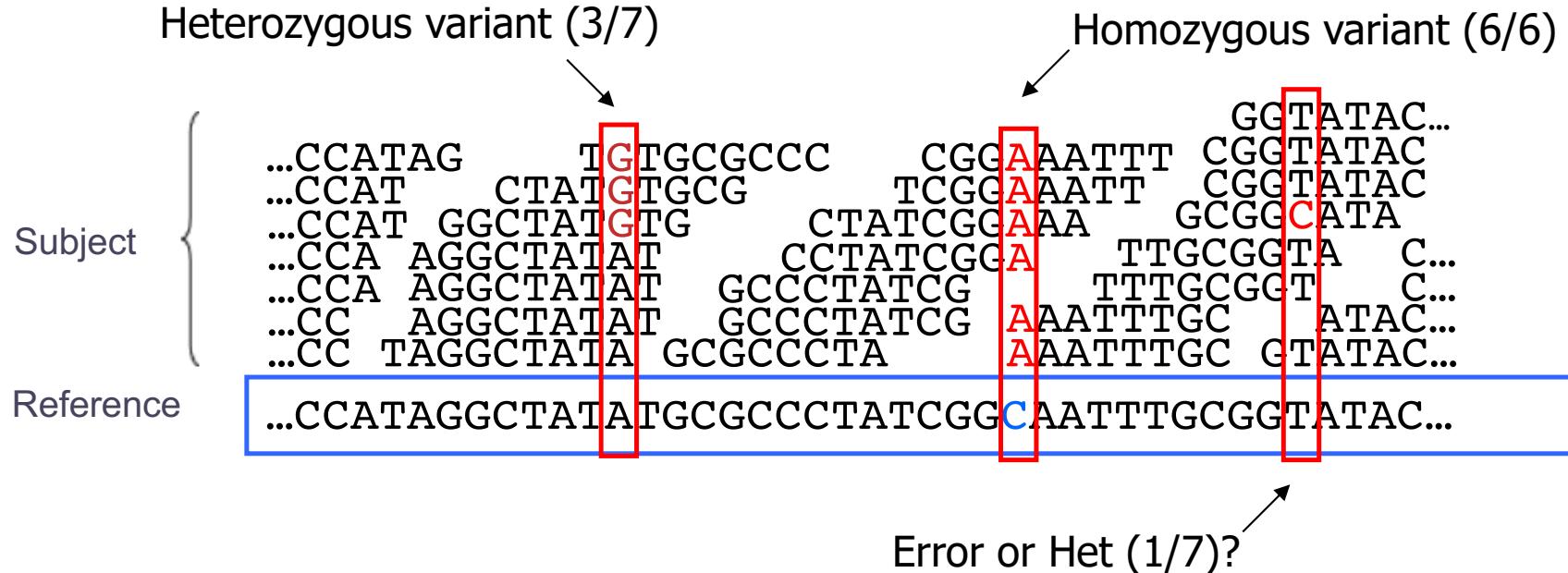
```
tccCAGTTATGTCAGggacacgagcatgcagagac  
|||||||||||  
aattgccgcgtcgatcagCAGTTATGTCAGatc
```

Part 3: Variant Calling

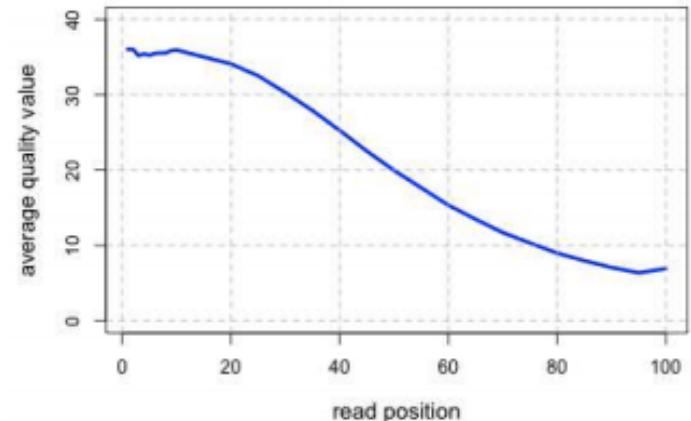
Variant Calling Overview



Genotyping Theory



- If there were no sequencing errors, identifying SNPs would be very easy: any time a read disagrees with the reference, it must be a variant!
- Sequencing instruments make mistakes
 - Quality of read decreases over the read length
- 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



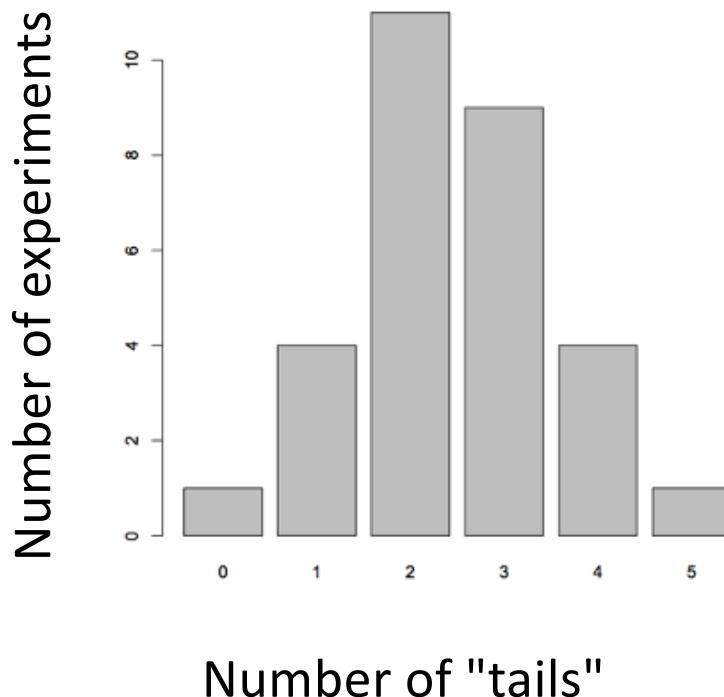
The Binomial Distribution: Adventures in Coin Flipping



$$P(\text{heads}) = 0.5$$

$$P(\text{tails}) = 0.5$$

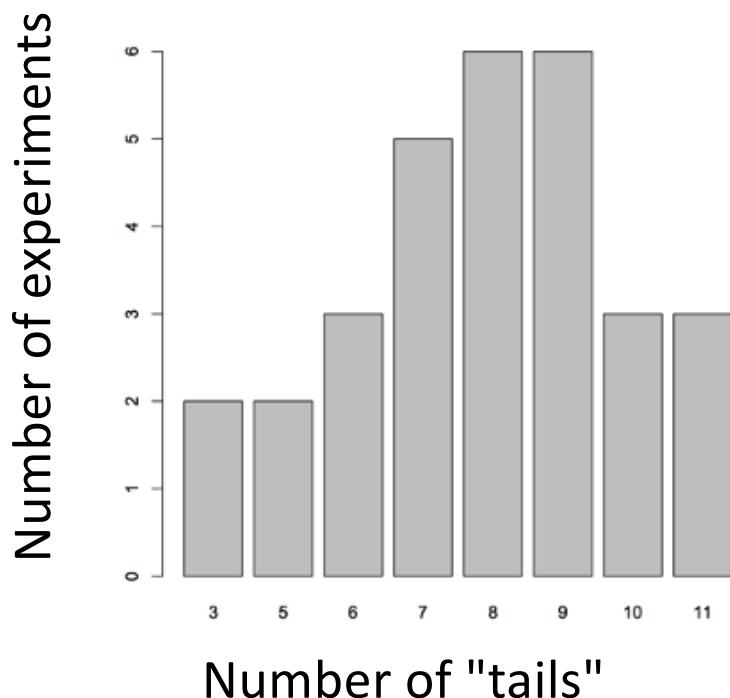
What is the distribution of tails (alternate alleles) do we expect to see after 5 tosses (sequence reads)?



R code:

```
barplot(table(rbinom(30, 5, 0.5)))  
30 experiments (students tossing coins)  
5 tosses each  
Probability of Tails
```

What is the distribution of tails (alternate alleles) do we expect to see after 15 tosses (sequence reads)?



R code:

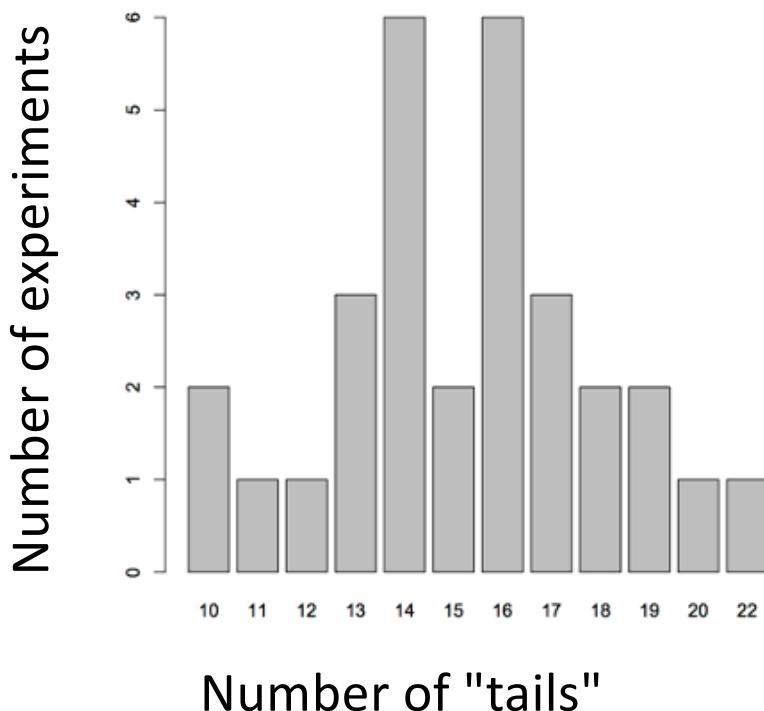
```
barplot(table(rbinom(30, 15, 0.5)))
```

30 experiments (students tossing coins)

15 tosses each

Probability of Tails

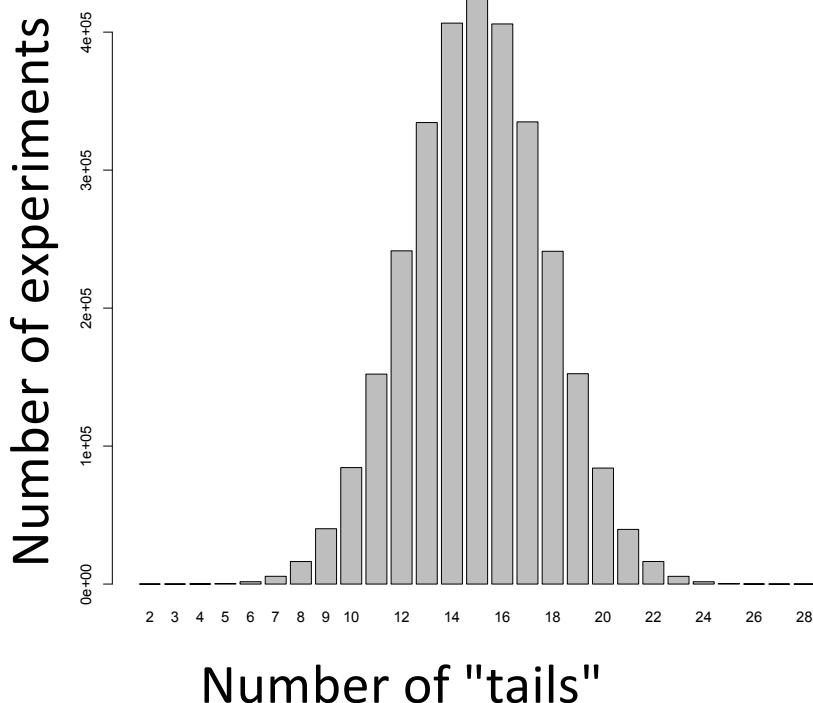
What is the distribution of tails (alternate alleles) do we expect to see after 30 tosses (sequence reads)?



R code:

```
barplot(table(rbinom(30, 30, 0.5)))  
30 experiments (students tossing coins)  
30 tosses each  
Probability of Tails
```

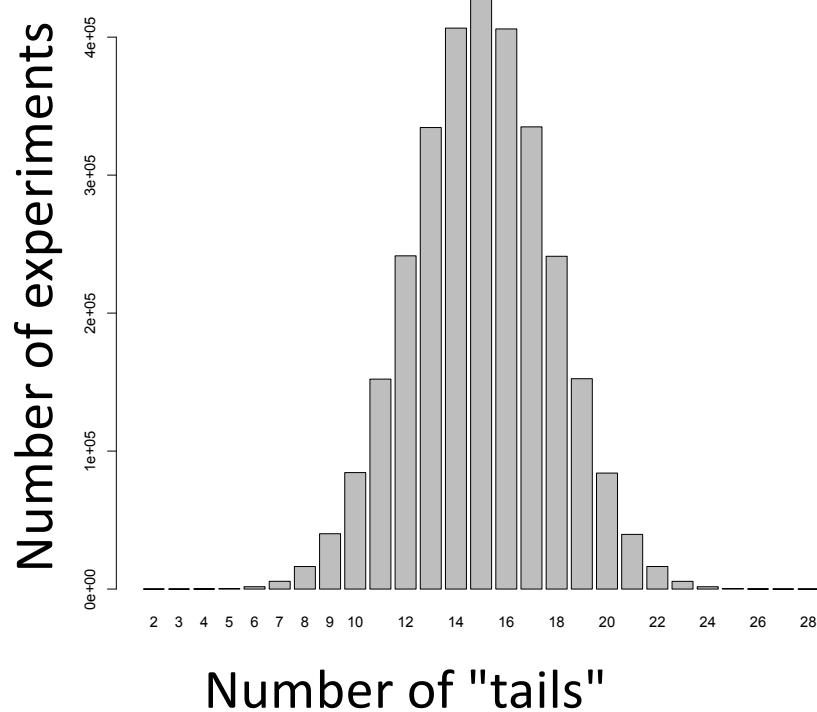
What is the distribution of tails (alternate alleles) do we expect to see after 30 tosses (sequence reads)?



R code:

```
barplot(table(rbinom(3e6, 30, 0.5)))  
3M experiments (students tossing coins)  
30 tosses each  
Probability of Tails
```

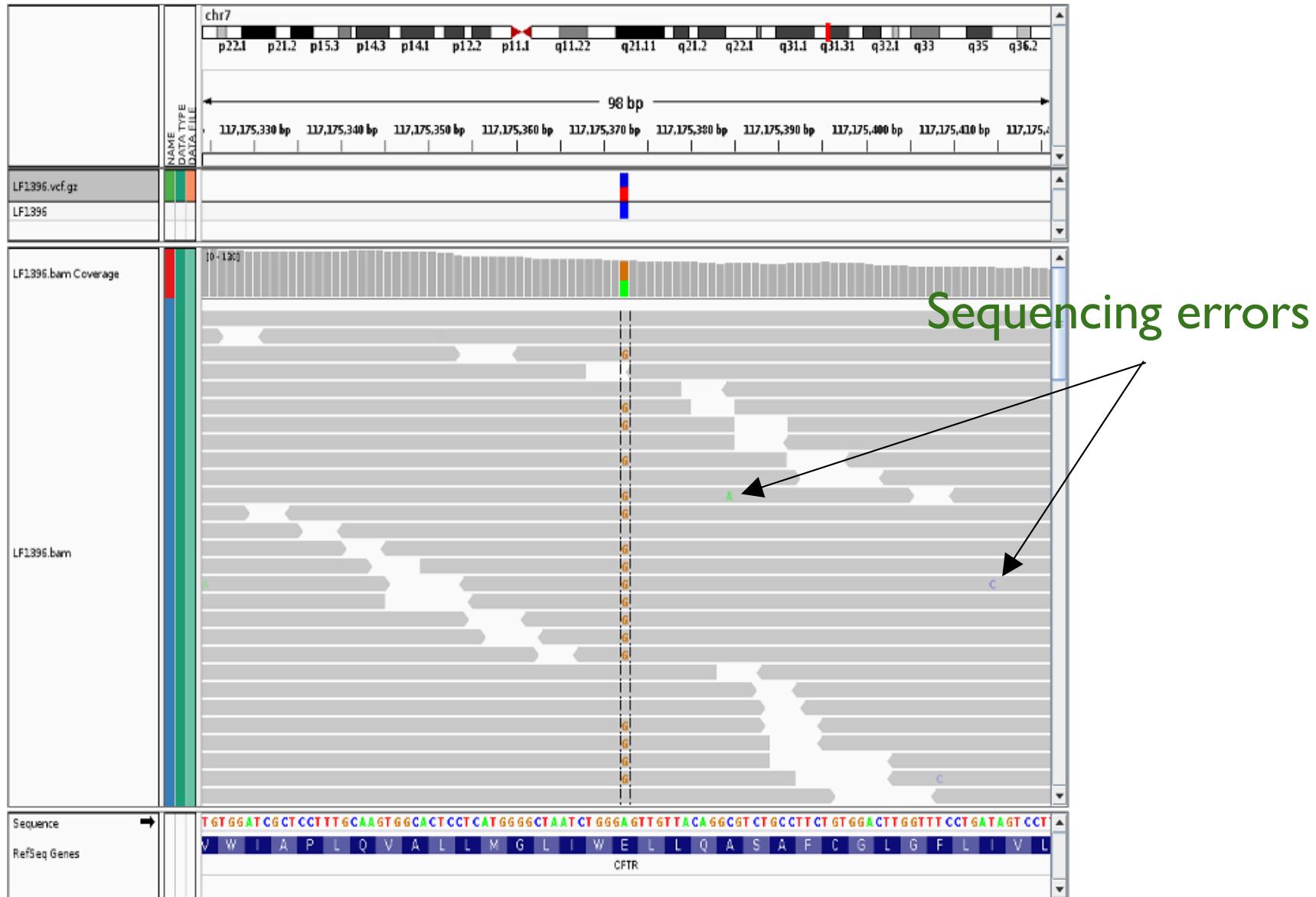
So, with 30 tosses (reads), we are much more likely to see an even mix of alternate and reference alleles at a heterozygous locus in a genome



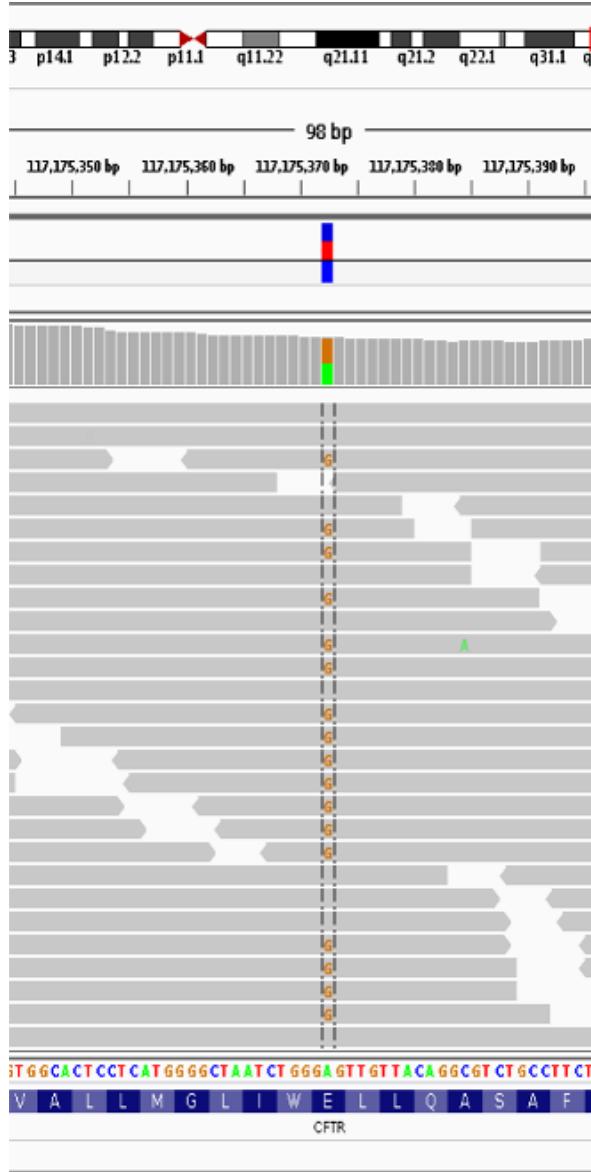
This is why at least a "30X" (30 fold sequence coverage) genome is recommended: it confers sufficient power to distinguish heterozygous alleles and from mere sequencing errors

$P(3/30 \text{ het}) <?> P(3/30 \text{ err})$

Sequencing errors fall out as noise (most of the time)



What information is needed to decide if a variant exists?



- Depth of coverage at the locus
- Bases observed at the locus
- The base qualities of each allele
- The strand composition
- Mapping qualities
- Proper pairs?
- Expected polymorphism rate

PolyBayes: The first statistically rigorous variant detection tool.

letter

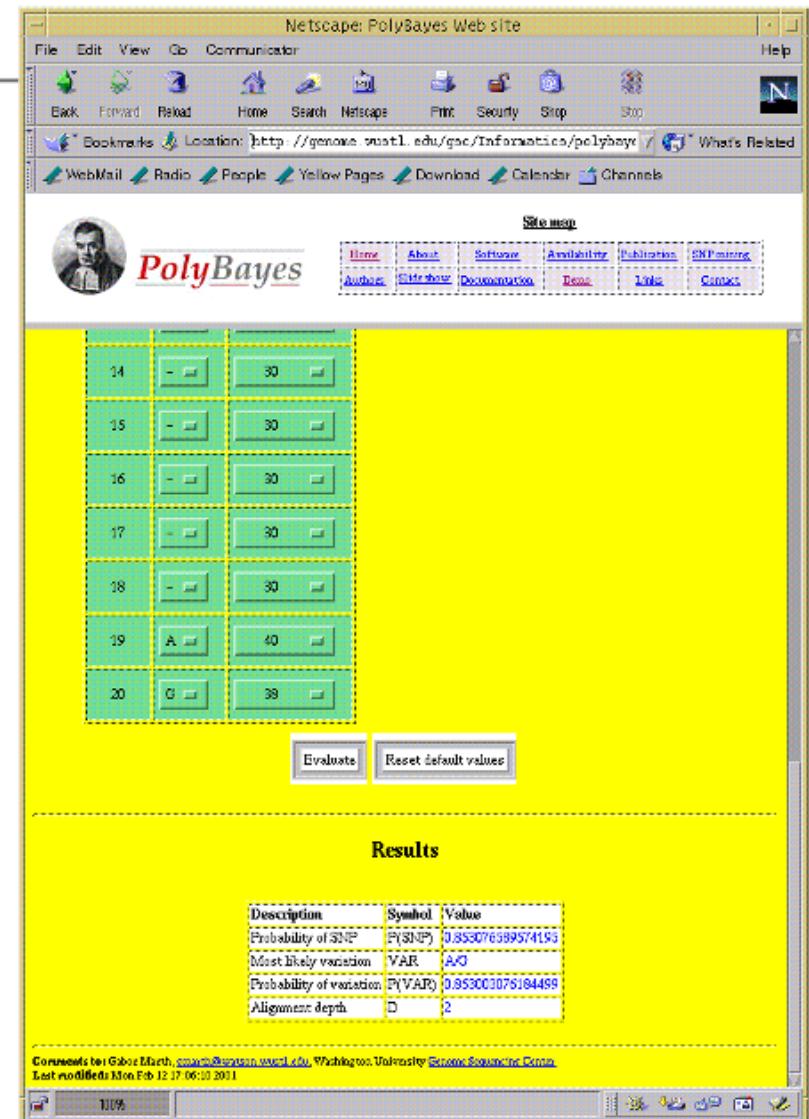


© 1999 Nature America Inc. • <http://genetics.nature.com>

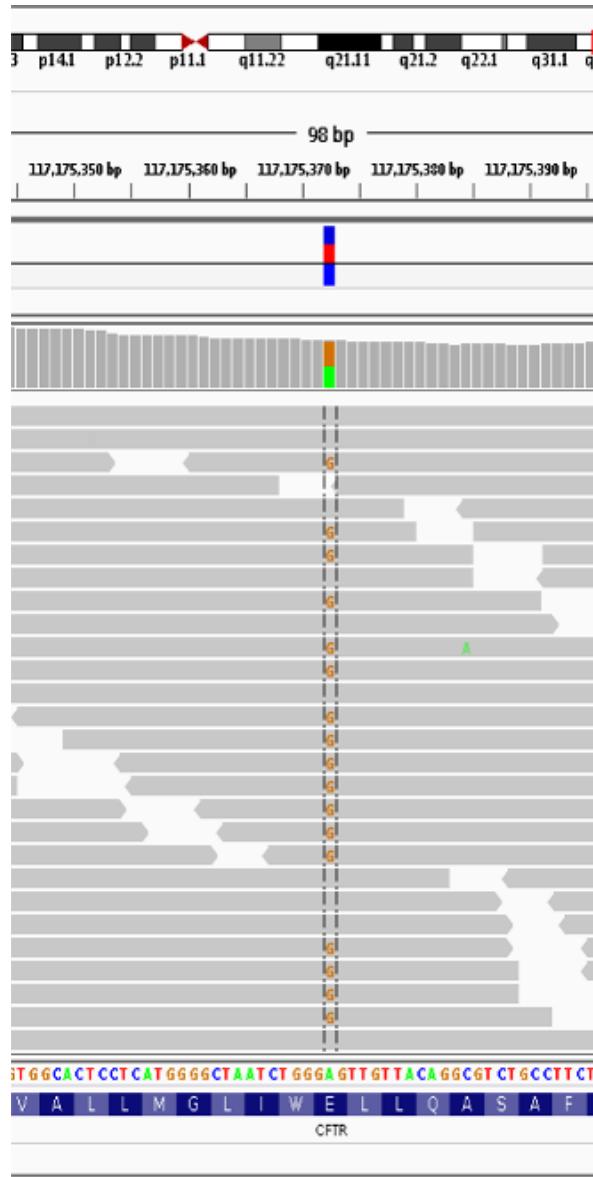
A general approach to single-nucleotide polymorphism discovery

Gabor T. Marth¹, Ian Korf¹, Mark D. Yandell¹, Raymond T. Yeh¹, Zhijie Gu², Hamideh Zakeri², Nathan O. Stitzel¹, LaDeana Hillier¹, Pui-Yan Kwok² & Warren R. Gish¹

Its main innovation was the use of Bayes's theorem



Bayesian SNP calling



$$P(\text{SNP} | \text{Data}) = \frac{P(\text{Data} | \text{SNP}) * P(\text{SNP})}{P(\text{Data})}$$

PolyBayes: The first statistically rigorous variant detection tool.

letter

© 1999 Nature America Inc. • <http://genetics.nature.com>

A general approach to single-nucleotide polymorphism discovery

Gabor T. Marth¹, Ian Korf¹, Mark D. Yandell¹, Raymond T. Yeh¹, Zhijie Gu², Hamideh Zakeri², Nathan O. Stitzel¹, LaDeana Hillier¹, Pui-Yan Kwok² & Warren R. Gish¹

Bayesian posterior probability

$$P(\text{SNP}) = \sum_{\text{all variable } S} \frac{\frac{P(S_1 | R_1) \dots P(S_N | R_N)}{P_{\text{Prior}}(S_1) \dots P_{\text{Prior}}(S_N)} \cdot P_{\text{Prior}}(S_1, \dots, S_N)}{\sum_{S_{i_1} \in [A,C,G,T]} \dots \sum_{S_{i_N} \in [A,C,G,T]} \frac{P(S_{i_1} | R_1) \dots P(S_{i_N} | R_1)}{P_{\text{Prior}}(S_{i_1}) \dots P_{\text{Prior}}(S_{i_N})} \cdot P_{\text{Prior}}(S_{i_1}, \dots, S_{i_N})}$$

Probability of observed base composition
(should model sequencing error rate)

Base call +
Base quality

Expected (prior)
polymorphism rate

PolyBayes: The first statistically rigorous variant detection tool.

letter

 © 1999 Nature America Inc. • <http://genetics.nature.com>

A general approach to single-nucleotide polymorphism discovery

Gabor T. Marth¹, Ian Korf¹, Mark D. Yandell¹, Raymond T. Yeh¹, Zhijie Gu², Hamideh Zakeri², Nathan O. Stitzel¹, LaDeana Hillier¹, Pui-Yan Kwok² & Warren R. Gish¹

This Bayesian statistical framework has been adopted by other modern SNP/INDEL callers such as FreeBayes, GATK, and samtools

VCF Format

Example

VCF header {

```
##fileformat=VCFv4.0
##fileDate=20100707
##source=VCFtools
##reference=NCBI36
##INFO=<ID=AA,Number=1>Type=String,Description="Ancestral Allele">
##INFO=<ID=H2,Number=0>Type=Flag,Description="HapMap2 membership">
##FORMAT=<ID=GT,Number=1>Type=String,Description="Genotype">
##FORMAT=<ID=GQ,Number=1>Type=Integer,Description="Genotype Quality (phred score)">
##FORMAT=<ID=GL,Number=3>Type=Float,Description="Likelihoods for RR,RA,AA genotypes (R=ref,A=alt)">
##FORMAT=<ID=DP,Number=1>Type=Integer,Description="Read Depth">
##ALT=<ID=DEL,Description="Deletion">
##INFO=<ID=SVTYPE,Number=1>Type=String,Description="Type of structural variant">
##INFO=<ID=END,Number=1>Type=Integer,Description="End position of the variant">
#CHROM POS ID REF ALT QUAL FILTER INFO FORMAT SAMPLE1 SAMPLE2
1 1 .
1 2 rs1 C A,AT PASS .
1 5 . G T,CT PASS H2;AA=T
1 100 <DEL> . PASS .
SVTYPE=DEL;END=300 GT:DP 1/2:13 0/0:29
GT:GQ 0/1:100 2/2:70
GT:GQ 1/0:77 1/1:95
GT:GQ:DP 1/1:12:3 0/0:20
```

Mandatory header lines

Optional header lines (meta-data about the annotations in the VCF body)

Body {

CHROM	POS	ID	REF	ALT	QUAL	FILTER	INFO	FORMAT	SAMPLE1	SAMPLE2
1	1	.	ACG	A,AT	.	PASS	.	GT:DP	1/2:13	0/0:29
1	2	rs1	C	T,CT	.	PASS	H2;AA=T	GT:GQ	0/1:100	2/2:70
1	5	.	G	T	.	PASS	.	GT:GQ	1/0:77	1/1:95
1	100				.	PASS	SVTYPE=DEL;END=300	GT:GQ:DP	1/1:12:3	0/0:20

Deletion

SNP

Large SV

Insertion

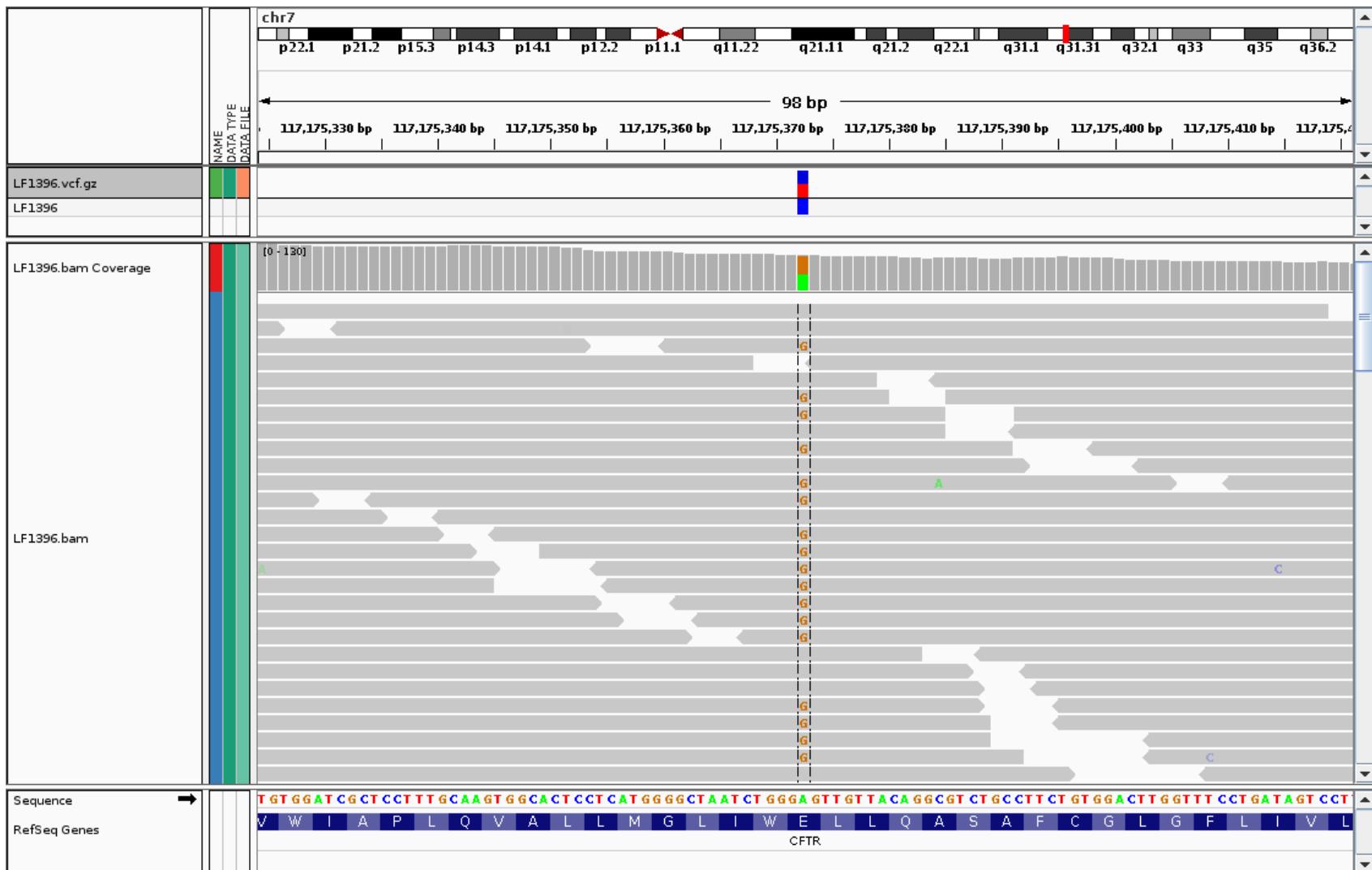
Other event

Reference alleles (GT=0)

Alternate alleles (GT>0 is an index to the ALT column)

Phased data (G and C above are on the same chromosome)

VCF Format



#CHROM	POS	ID	REF	ALT	QUAL	FILTER	INFO	FORMAT	LF1396
chr7	117175373	.	A	G	90	PASS	AF=0.5	GT	0/1