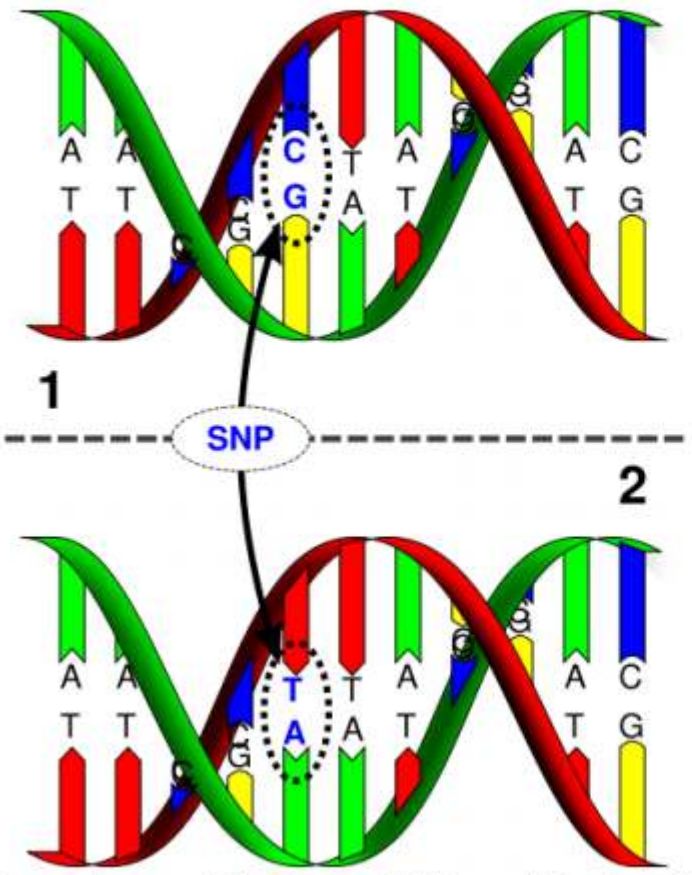
Mutation calling intro I.



http://www.science.marshall.edu/murraye/341/Images/416px-Dna-SNP_svg.png

Individual humans differ approximately at every 1000th bp from the reference genome

The majority of variations are single nucleotide differences

Polymorphisms = inherited differences

Somatic variation = acquired differences

Mutations can affect genes (eg. Cystic fibrosis and mutation in CFTR)

Mutation calling intro II

A nucleotide change that is acquired during life time

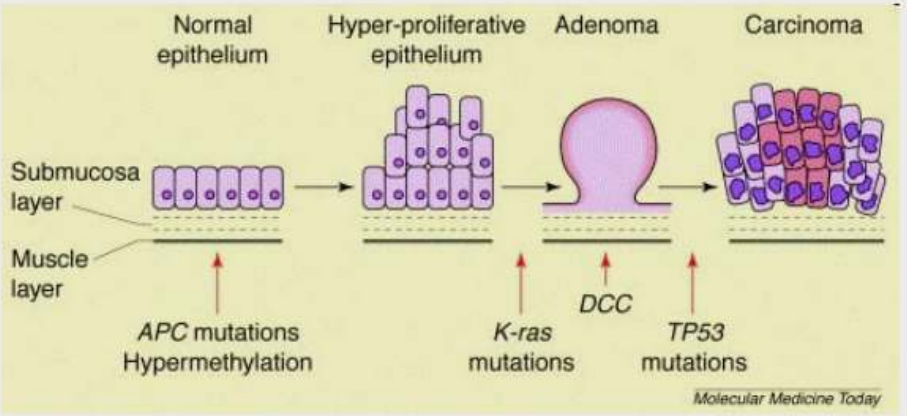
Example: TP53

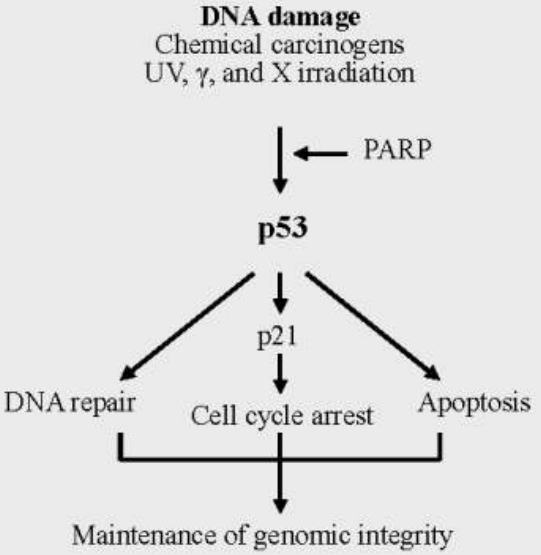
Gene is mutated in half of all human tumors

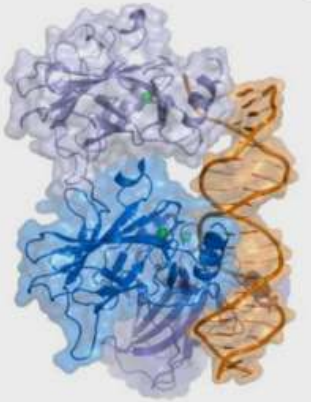
SNVs disrupt tumor suppressor function

27,580 SNVs known

http://www.p53.iarc.fr/Statistics.html







http://ars.sciencedirect.com/content/image/1-s2.0-S1357431099015981-gr1.jpg

http://herkules.oulu.fi/isbn9514270398/html/graphic22.png

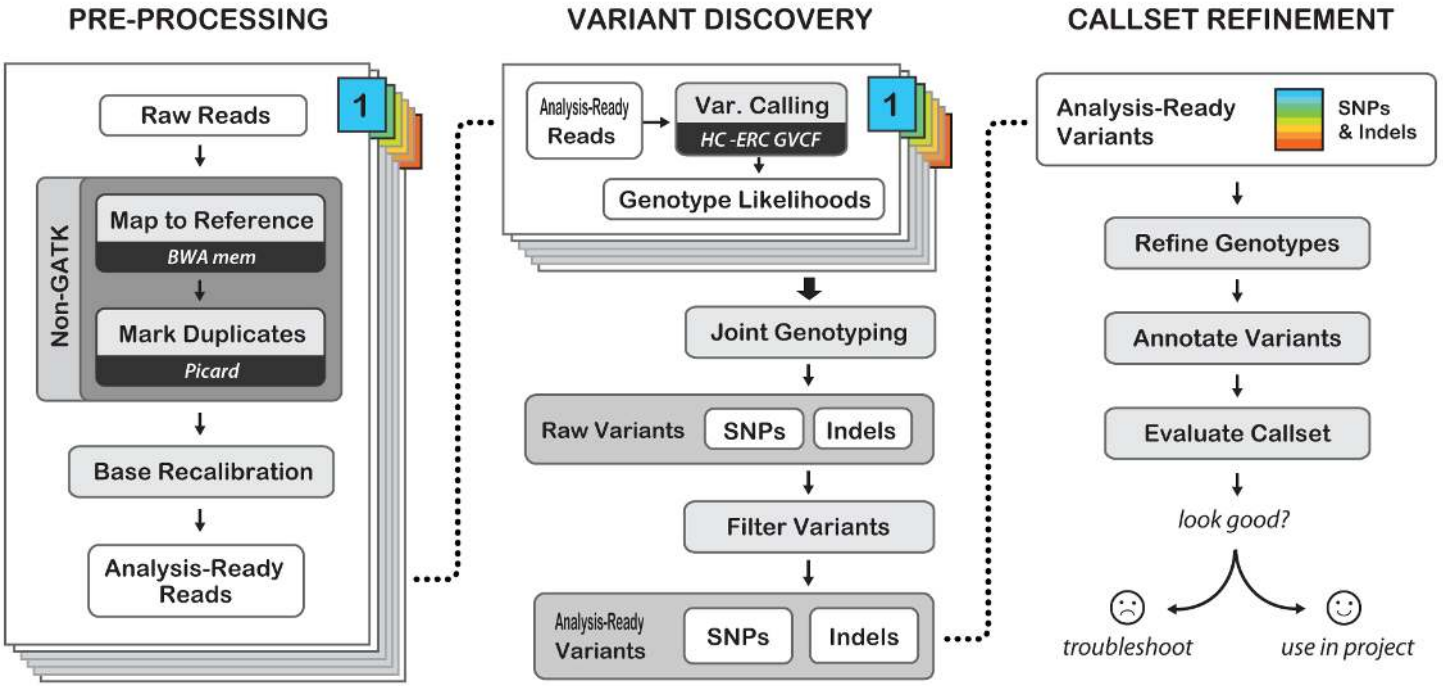
http://wikipedia/p53

Mutation calling steps

- 1) Quality control (FastQC)
  - QC, trimming
- 2) Aligning reads to genome
  - BWA, bowtie2, novoalign
- 3) Identifying mutations
  - GATK (haplotypecaller, mutect2, varscan2…..)
- 4) Annotation mutations
  - SNPeff, Anovar
- 5) Post-analysis steps

Mutation callers used today

- Many programs are available
  - Gold-standard: GATK haplotype caller and Mutect2
    - Haplotype caller: germline mutation
    - Mutect2: somatic mutations (tumor)



Best Practices for Germline SNPs and Indels in Whole Genomes and Exomes - June 2016

https://software.broadinstitute.org/gatk/best-practices/

Today: Breast cancer mutation calling

- Identify
  - Germline mutations
  - Somatic mutations

Methods

- Aligner
  - Burrows-Wheeler aligner (BWA)

Input data

Exome-seq data from tumor and normal (blood) sample

- Mutation callers (both fromGATK):
  - Germline mutation: GATK haplotype caller
  - Somatic mutation: Mutect2

Connect to server (node, not head):
-Ubuntu: ssh -X <username>@130.211.107.222

-Windows (PUTTY): <username>@130.211.107.222
SET X11 in settings!!!!!!

Sequence alignment I.

(We will be working with a subset of reads that will align to the human chromosome 17)

Script name: Align_and_sort.pl

Usage:

" **Align_and_sort.pl** <prefix> <genome.index> <input_fq>"

---

1) Aligning tumor reads

" **Align_and_Sort.pl** tumor \
/gfs/data/mutation/genome/Homo_sapiens.GRCh38.dna.chromosome.17.fa \
/gfs/data/mutation/reads/tumor.fq"

Output: tumor_sorted.bam

---

2) Aligning normal reads

"**Align_and_Sort.pl** normal \
/gfs/data/mutation/genome/Homo_sapiens.GRCh38.dna.chromosome.17.fa \
/gfs/data/mutation/reads/normal.fq"

Output: normal_sorted.bam

Sequence alignment II.

- What happens in the script?

1) Align reads to genome (output is SAM format, simple alignment and mapping format)

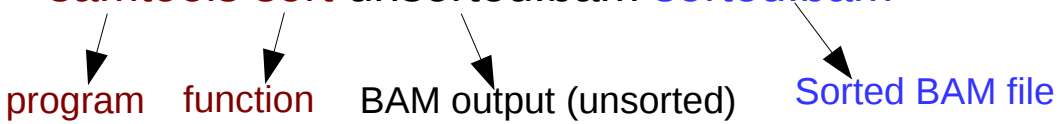
```
bwa mem -t 3 -R '\@RG\tID:gyak\tSM:ubi' <genome> <reads.fq> > unsorted.sam
```

2) Convert SAM to BAM (binary format)

```
samtools import <genome> unsorted.sam unsorted.bam
```

3) Sort reads based on genomic position

```
samtools sort unsorted.bam sorted.bam
```



4) Create a genome index

```
samtools index sorted.bam
```

Creating the Burrows-Wheeler transformed and alignment basics

Small comparison of alignment methods

Running time (sec)			
	Small reference		
	Smith-Waterman	BLAST	Bowtie2
1,000 reads	5	0,2	0,08
10,000 reads	51	5	1
100,000 reads	120	60	6
	Human genome chr 1		
1,000 reads	NA	0,4	0,2
10,000 reads	NA	5	1
100,000 reads	NA	65	9
	Human genome indexing time		
	NA	5 mins.	~1 hour

File size
313 bytes

File size
253 Mb

```
lorinc@TTK:~/bioinfo/test/benchmark$ time bowtie2 -f -x genome/chr1.fa input_100k.fasta > temp
100000 reads; of these:
  100000 (100.00%) were unpaired; of these:
    0 (0.00%) aligned 0 times
    100000 (100.00%) aligned exactly 1 time
    0 (0.00%) aligned >1 times
100.00% overall alignment rate

real    0m8.961s
user    0m8.849s
sys     0m0.114s
lorinc@TTK:~/bioinfo/test/benchmark$
```

Bowtie2 stats
Real alignments!!

BWT indexing

Input sequence

AGCAGCAGACT

Add $ to end

BW matrix

Index	Sequence											
0	A	G	C	A	G	C	A	G	A	C	T	$
1	$	A	G	C	A	G	C	A	G	A	C	T
2	T	$	A	G	C	A	G	C	A	G	A	C
3	C	T	$	A	G	C	A	G	C	A	G	A
4	A	C	T	$	A	G	C	A	G	C	A	G
5	G	A	C	T	$	A	G	C	A	G	C	A
6	A	G	A	C	T	$	A	G	C	A	G	C
7	C	A	G	A	C	T	$	A	G	C	A	G
8	G	C	A	G	A	C	T	$	A	G	C	A
9	A	G	C	A	G	A	C	T	$	A	G	C
10	C	A	G	C	A	G	A	C	T	$	A	G
11	G	C	A	G	C	A	G	A	C	T	$	A

Sort

Sorted BW matrix

Burrows-Wheeler transformed is the last column in the sorted matrix

New Index	Index	Sequence											
0	1	$	A	G	C	A	G	C	A	G	A	C	T
1	4	**A**	**C**	**T**	$	A	G	C	A	G	C	A	G
2	6	**A**	**G**	**A**	**C**	**T**	$	A	G	C	A	G	C
3	9	**A**	**G**	**C**	**A**	**G**	**A**	**C**	**T**	$	A	G	C
4	0	**A**	**G**	**C**	**A**	**G**	**C**	**A**	**G**	**A**	**C**	**T**	$
5	7	**C**	**A**	**G**	**A**	**C**	**T**	$	A	G	C	A	G
6	10	**C**	**A**	**G**	**C**	**A**	**G**	**A**	**C**	**T**	$	A	G
7	3	**C**	**T**	$	A	G	C	A	G	C	A	G	A
8	5	**G**	**A**	**C**	**T**	$	A	G	C	A	G	C	A
9	8	**G**	**C**	**A**	**G**	**A**	**C**	**T**	$	A	G	C	A
10	11	**G**	**C**	**A**	**G**	**C**	**A**	**G**	**A**	**C**	**T**	$	A
11	2	**T**	$	A	G	C	A	G	C	A	G	A	C

Burrows-Wheeler transformed text: TGCC$GGAAAAC

BWT matrix key points: **Rank table**

New Index	Index	Sequence											
0	1	$	A	G	C	A	G	C	A	G	A	C	T
1	4	**A**	**C**	**T**	$	A	G	C	A	G	C	A	G
2	6	**A**	**G**	**A**	**C**	**T**	$	A	G	C	A	G	C
3	9	**A**	**G**	**C**	**A**	**G**	**A**	**C**	**T**	$	A	G	C
4	0	**A**	**G**	**C**	**A**	**G**	**C**	**A**	**G**	**A**	**C**	**T**	$
5	7	**C**	**A**	**G**	**A**	**C**	**T**	$	A	G	C	A	G
6	10	**C**	**A**	**G**	**C**	**A**	**G**	**A**	**C**	**T**	$	A	G
7	3	**C**	**T**	$	A	G	C	A	G	C	A	G	A
8	5	**G**	**A**	**C**	**T**	$	A	G	C	A	G	C	A
9	8	**G**	**C**	**A**	**G**	**A**	**C**	**T**	$	A	G	C	A
10	11	**G**	**C**	**A**	**G**	**C**	**A**	**G**	**A**	**C**	**T**	$	A
11	2	**T**	$	A	G	C	A	G	C	A	G	A	C

Rank table

	A	**C**	**G**	**T**
rank	1	5	8	11

Row index where given char appears in first column

BWT matrix key points: **Occurrence table**

													Occurance			
New Index	Index	Sequence											**A**	**C**	**G**	**T**
0	1	$	A	G	C	A	G	C	A	G	A	C	**T**	**0**	**0**	**0**	**1**
1	4	A	C	T	$	A	G	C	A	G	C	A	**G**	**0**	**0**	**1**	**1**
2	6	A	G	A	C	T	$	A	G	C	A	G	**C**	**0**	**1**	**1**	**1**
3	9	A	G	C	A	G	A	C	T	$	A	G	**C**	**0**	**2**	**1**	**1**
4	0	A	G	C	A	G	C	A	G	A	C	T	**$**	**0**	**2**	**1**	**1**
5	7	C	A	G	A	C	T	$	A	G	C	A	**G**	**0**	**2**	**2**	**1**
6	10	C	A	G	C	A	G	A	C	T	$	A	**G**	**0**	**2**	**3**	**1**
7	3	C	T	$	A	G	C	A	G	C	A	G	**A**	**1**	**2**	**3**	**1**
8	5	G	A	C	T	$	A	G	C	A	G	C	**A**	**2**	**2**	**3**	**1**
9	8	G	C	A	G	A	C	T	$	A	G	C	**A**	**3**	**2**	**3**	**1**
10	11	G	C	A	G	C	A	G	A	C	T	$	**A**	**4**	**2**	**3**	**1**
11	2	T	$	A	G	C	A	G	C	A	G	A	**C**	**4**	**3**	**3**	**1**

BWT

Occurrence is the “rolling sum” of a given char in the BWT

BWT input data needed for exact string search (“alignment”)

			Occurance			
BW matrix			**A**	**C**	**G**	**T**
$	..	**T**	**0**	**0**	**0**	**1**
A	..	**G**	**0**	**0**	**1**	**1**
A	..	**C**	**0**	**1**	**1**	**1**
A	..	**C**	**0**	**2**	**1**	**1**
A	..	**$**	**0**	**2**	**1**	**1**
C	..	**G**	**0**	**2**	**2**	**1**
C	..	**G**	**0**	**2**	**3**	**1**
C	..	**A**	**1**	**2**	**3**	**1**
G	..	**A**	**2**	**2**	**3**	**1**
G	..	**A**	**3**	**2**	**3**	**1**
G	..	**A**	**4**	**2**	**3**	**1**
T	..	**C**	**4**	**3**	**3**	**1**

First column

Last column

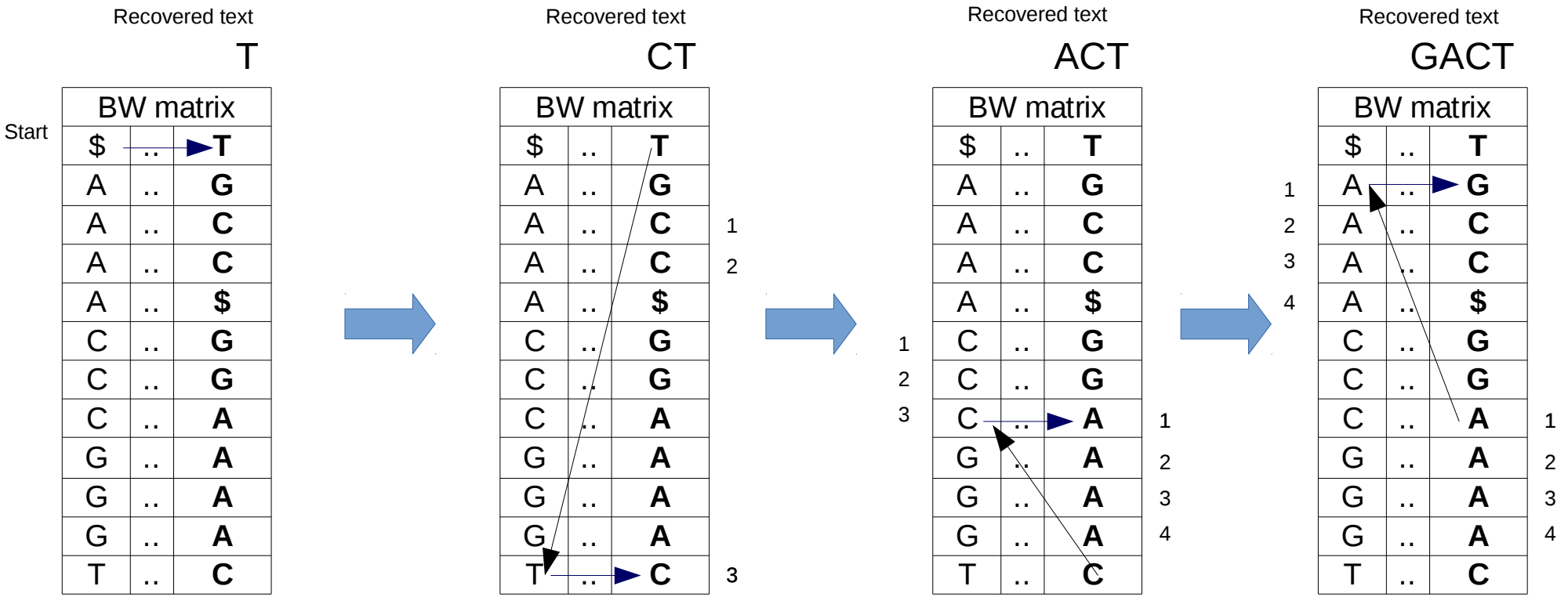
Occurrence table

Rank table

	A	**C**	**G**	**T**
rank	1	5	8	11

Using the “Last to first” mapping property

String recovery: last-to-first mapping

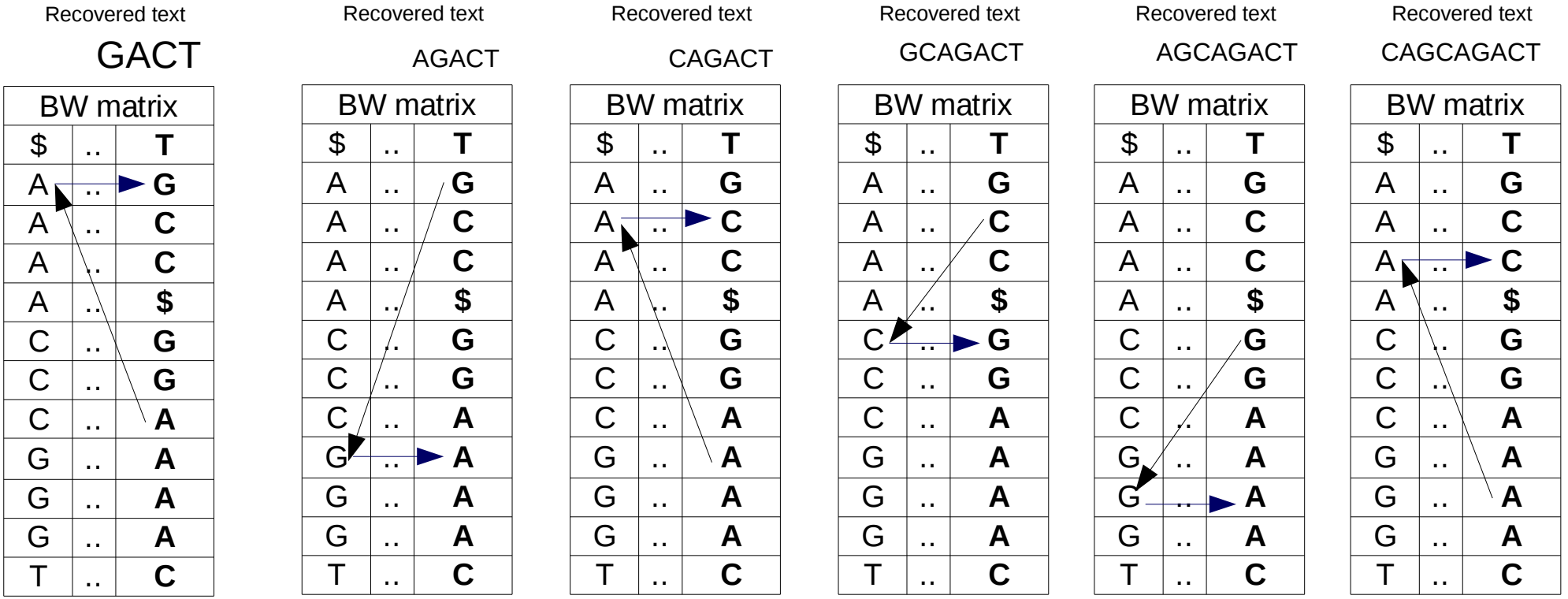


T mapped to 3rd **C** in last column

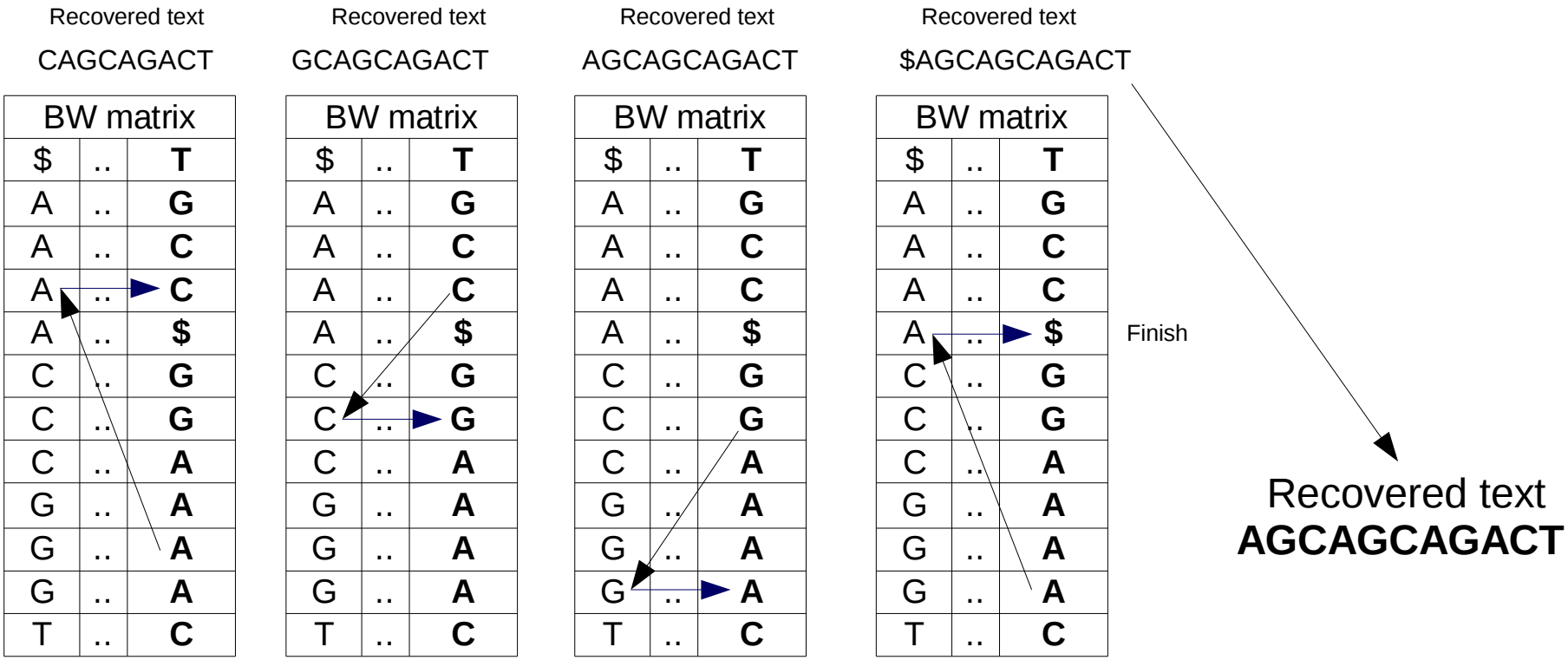
Look for 3rd **C** in the first column
→ **C** maps to first **A**

Look for 1st **A** in the first column
→ **A** maps to first **G**

String recovery: last-to-first mapping cont.



String recovery: last-to-first mapping cont.



Input sequence

AGCAGCAGACT

String searching

	A	C	G	T
rank	1	5	8	11

Source: BWA article, 2009, Bioinformatics

2.4 Exact matching: backward search

Let $C(a)$ be the number of symbols in $X[0,n-2]$ that are lexicographically smaller than $a \in \Sigma$ and $O(a,i)$ the number of occurrences of a in $B[0,i]$. Ferragina and Manzini (2000) proved that if W is a substring of X:

$$\underline{R}(aW) = C(a)+O(a,\underline{R}(W)-1)+1 \quad (3)$$

$$\overline{R}(aW) = C(a)+O(a,\overline{R}(W)) \quad (4)$$

and that $\underline{R}(aW) \leq \overline{R}(aW)$ if and only if aW is a substring of X. This result makes it possible to test whether W is a substring of X and to count the occurrences of W in $O(|W|)$ time by iteratively calculating $\underline{R}$ and $\overline{R}$ from the end of W. This procedure is called *backward search*.

It is important to note that Equations (3) and (4) actually realize the top-down traversal on the prefix trie of X given that we can calculate the SA interval of a child node in constant time if we know the interval of its parent. In this sense, backward search is equivalent to exact string matching on the prefix trie, but without explicitly putting the trie in the memory.

From Rank table

From occurrence table

			Occurance			
BW matrix			A	C	G	T
$	..	T	0	0	0	1
A	..	G	0	0	1	1
A	..	C	0	1	1	1
A	..	C	0	2	1	1
A	..	$	0	2	1	1
C	..	G	0	2	2	1
C	..	G	0	2	3	1
C	..	A	1	2	3	1
G	..	A	2	2	3	1
G	..	A	3	2	3	1
G	..	A	4	2	3	1
T	..	C	4	3	3	1

String searching example

String: GCA → Last to first mapping, so we start with **A**

	A	**C**	**G**	**T**
rank	1	5	8	11

Important: **Occ(“A” -1)**

				Occurance			
New Index	BW matrix			**A**	**C**	**G**	**T**
0	$	..	**T**	**0**	**0**	**0**	**1**
1	A	..	**G**	**0**	**0**	**1**	**1**
2	A	..	**C**	**0**	**1**	**1**	**1**
3	A	..	**C**	**0**	**2**	**1**	**1**
4	A	..	**$**	**0**	**2**	**1**	**1**
5	C	..	**G**	**0**	**2**	**2**	**1**
6	C	..	**G**	**0**	**2**	**3**	**1**
7	C	..	**A**	**1**	**2**	**3**	**1**
8	G	..	**A**	**2**	**2**	**3**	**1**
9	G	..	**A**	**3**	**2**	**3**	**1**
10	G	..	**A**	**4**	**2**	**3**	**1**
11	T	..	**C**	**4**	**3**	**3**	**1**

From: **1** + **0** **= 1**

To: **1** + **4** **= 5**

String searching example

String: GCA → Last to first mapping, so we start with **A**

	A	**C**	**G**	**T**
rank	1	5	8	11

				Occurance			
New Index	BW matrix			**A**	**C**	**G**	**T**
0	$	..	**T**	**0**	**0**	**0**	**1**
1	A	..	**G**	**0**	**0**	**1**	**1**
2	A	..	**C**	**0**	**1**	**1**	**1**
3	A	..	**C**	**0**	**2**	**1**	**1**
4	A	..	**$**	**0**	**2**	**1**	**1**
5	C	..	**G**	**0**	**2**	**2**	**1**
6	C	..	**G**	**0**	**2**	**3**	**1**
7	C	..	**A**	**1**	**2**	**3**	**1**
8	G	..	**A**	**2**	**2**	**3**	**1**
9	G	..	**A**	**3**	**2**	**3**	**1**
10	G	..	**A**	**4**	**2**	**3**	**1**
11	T	..	**C**	**4**	**3**	**3**	**1**

A

From: **1** + **0** **= 1**

To: **1** + **4** **= 5**

String searching example

String: GCA → Last to first mapping, so we start with **A**

	A	**C**	**G**	**T**
rank	1	5	8	11

				Occurance			
New Index	BW matrix			**A**	**C**	**G**	**T**
0	$	..	**T**	**0**	**0**	**0**	**1**
1	A	..	**G**	**0**	**0**	**1**	**1**
2	A	..	**C**	**0**	**1**	**1**	**1**
3	A	..	**C**	**0**	**2**	**1**	**1**
4	A	..	**$**	**0**	**2**	**1**	**1**
5	C	..	**G**	**0**	**2**	**2**	**1**
6	C	..	**G**	**0**	**2**	**3**	**1**
7	C	..	**A**	**1**	**2**	**3**	**1**
8	G	..	**A**	**2**	**2**	**3**	**1**
9	G	..	**A**	**3**	**2**	**3**	**1**
10	G	..	**A**	**4**	**2**	**3**	**1**
11	T	..	**C**	**4**	**3**	**3**	**1**

A

From: **1** + **0** **= 1**

To: **1** + **4** **= 5**

String searching example

String: G**C**A

Next char is **C**

(last to first mapping, so consider rows of **A**)

	A	C	G	T
rank	1	5	8	11

				Occurance			
New Index	BW matrix			A	C	G	T
0	$	..	**T**	**0**	**0**	**0**	**1**
1	A	..	**G**	**0**	**0**	**1**	**1**
2	A	..	**C**	**0**	**1**	**1**	**1**
3	A	..	**C**	**0**	**2**	**1**	**1**
4	A	..	**$**	**0**	**2**	**1**	**1**
5	C	..	**G**	**0**	**2**	**2**	**1**
6	C	..	**G**	**0**	**2**	**3**	**1**
7	C	..	**A**	**1**	**2**	**3**	**1**
8	G	..	**A**	**2**	**2**	**3**	**1**
9	G	..	**A**	**3**	**2**	**3**	**1**
10	G	..	**A**	**4**	**2**	**3**	**1**
11	T	..	**C**	**4**	**3**	**3**	**1**

From: **5** + **0** = **5**

To: **5** + **2** = **7**

String searching example

String: G**C**A

Next char is **C**

(last to first mapping, so consider rows of **A**)

	A	**C**	**G**	**T**
rank	1	5	8	11

				Occurance			
New Index	BW matrix			**A**	**C**	**G**	**T**
0	$	..	**T**	**0**	**0**	**0**	**1**
1	A	..	**G**	**0**	**0**	**1**	**1**
2	A	..	**C**	**0**	**1**	**1**	**1**
3	A	..	**C**	**0**	**2**	**1**	**1**
4	A	..	**$**	**0**	**2**	**1**	**1**
5	C	..	**G**	**0**	**2**	**2**	**1**
6	C	..	**G**	**0**	**2**	**3**	**1**
7	C	..	**A**	**1**	**2**	**3**	**1**
8	G	..	**A**	**2**	**2**	**3**	**1**
9	G	..	**A**	**3**	**2**	**3**	**1**
10	G	..	**A**	**4**	**2**	**3**	**1**
11	T	..	**C**	**4**	**3**	**3**	**1**

From: **5** + **0** = **5**

To: **5** + **2** = **7**

String searching example

String: **G**CA

Next char is **G**

(last to first mapping, so consider rows of **AC**)

				Occurance			
New Index	BW matrix			**A**	**C**	**G**	**T**
0	$	..	**T**	**0**	**0**	**0**	**1**
1	A	..	**G**	**0**	**0**	**1**	**1**
2	A	..	**C**	**0**	**1**	**1**	**1**
3	A	..	**C**	**0**	**2**	**1**	**1**
4	A	..	**$**	**0**	**2**	**1**	**1**
5	C	..	**G**	**0**	**2**	**2**	**1**
6	C	..	**G**	**0**	**2**	**3**	**1**
7	C	..	**A**	**1**	**2**	**3**	**1**
8	G	..	**A**	**2**	**2**	**3**	**1**
9	G	..	**A**	**3**	**2**	**3**	**1**
10	G	..	**A**	**4**	**2**	**3**	**1**
11	T	..	**C**	**4**	**3**	**3**	**1**

	A	**C**	**G**	**T**
rank	1	5	8	11

From: **8** + **1** = **9**

To: **8** + 3 = **11**

String searching example

String: **G**CA

Next char is **G**

(last to first mapping, so consider rows of **AC**)

				Occurance			
New Index	BW matrix			**A**	**C**	**G**	**T**
0	$	..	**T**	**0**	**0**	**0**	**1**
1	A	..	**G**	**0**	**0**	**1**	**1**
2	A	..	**C**	**0**	**1**	**1**	**1**
3	A	..	**C**	**0**	**2**	**1**	**1**
4	A	..	**$**	**0**	**2**	**1**	**1**
5	C	..	**G**	**0**	**2**	**2**	**1**
6	C	..	**G**	**0**	**2**	**3**	**1**
7	C	..	**A**	**1**	**2**	**3**	**1**
8	G	..	**A**	**2**	**2**	**3**	**1**
9	G	..	**A**	**3**	**2**	**3**	**1**
10	G	..	**A**	**4**	**2**	**3**	**1**
11	T	..	**C**	**4**	**3**	**3**	**1**

	A	**C**	**G**	**T**
rank	1	5	8	11

From: **8** + **1** = **9**

To: **8** + 3 = **11**

There are two possible **G** chars
→ **GCA** is present two times in our input string

Input sequence AGCAGCAGACT

Position 1 Position 4

Mutation calling and annotation

1) Germline mutations
   - HaplotypeCaller
2) Somatic mutations
   - Mutect2
3) Annotating variants
   - SNPeff

Script name: GATK_somatic_caller.pl

GATK_somatic_caller.pl <prefix> <genome> <tumor.sorted.bam> <normal.sorted.bam>

Example command:

"GATK_somatic_caller.pl lorinc \
/gfs/data/mutation/genome/Homo_sapiens.GRCh38.dna.chromosome.17.fa \
tumor_sorted.bam \
normal_sorted.bam"

Running the mutation caller

GATK_somatic_caller.pl <prefix> <genome> <tumor.sorted.bam> <normal.sorted.bam>

"GATK_somatic_caller.pl lorinc \
/gfs/data/mutation/genome/Homo_sapiens.GRCh38.dna.chromosome.17.fa \
tumor_sorted.bam \
normal_sorted.bam"

- What happens in the script?

1) Germline mutation calling

java -jar GenomeAnalysisTK.jar -T HaplotypeCaller -R <genome> -I <normal.bam> -o haplotypeCaller.vcf -nct 3

program | algorithm | genome | Normal BAM file | Output | 3 threads

---

2) Somatic mutation calling

java -jar GenomeAnalysisTK.jar -T MuTect2 -R <genome> -I:tumor <tumor.bam> -I:normal <normal.bam> -o mutect2.vcf

program | algorithm | genome | Normal BAM file | Normal BAM file | Output

---

3) Annotating somatic mutations

java -jar snpEff.jar ann -canon GRCh38.86 mutect2.vcf > mutect2.snpeff.vcf

program | algorithm | Canonical transcripts only | Genome version | Input VCF | Annotated VCF output

Post analysis steps

- 1) Identify gene affected by mutation
- 2) Somatic mutation
  - Validate with IGV browser
- 3) Check published articles
- 4) perform survival analysis
  - Kmplot.com, cbioprortal, g-2-o.com