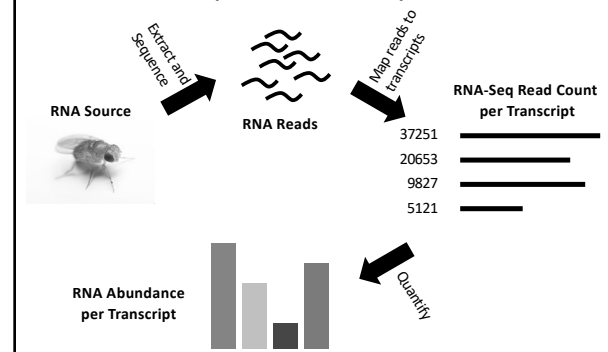


RNA-Seq: a Closer Look at Read Mapping and Quantitation

Jeremy Buhler
for GEP Alumni Workshop

1

RNA-Seq Pipeline for Expression Analysis



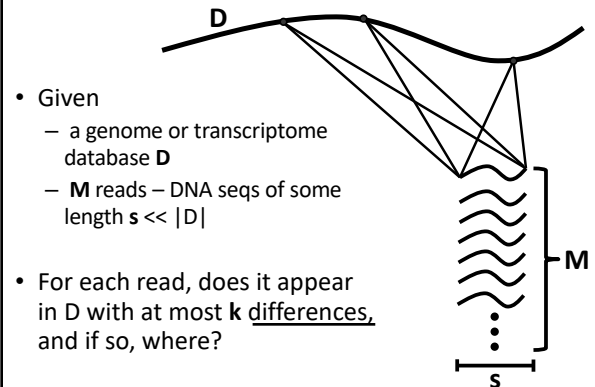
2

Topics for Today

- **Read mapping** and **RNA quantitation** from read counts are two key computational steps in RNA-Seq expression analysis.
- **Mapping**: how do we do it **fast**?
- **Quantitation**: how do we get from mapped reads to transcript abundance?

3

Problem: Short-Read Mapping



4

Typical Parameters

- Database **D** has size $10^7 - 10^{10}$ bases
- Number of reads **M** is $10^6 - 10^8$
- Length **s** is 75-150 (may vary among reads)
- # differences **k** is 0-3 (added, deleted, changed) to account for sequencing errors, polymorphisms

5

Design Pressures

- # reads **M** is huge! → minimal time per read
- Cannot afford to spend $O(|D|)$ time per read as in BLAST → need to **index database** in order to spend less time matching each read
- Index should be small enough to reside in fast memory, so as to maximize mapping speed

6

Basic Idea: Suffix Tree Index

- First, sort all suffixes of database string D to create sorted **suffix array A**.
- Second, merge all common prefixes of adjacent strings in A to create a **suffix tree T**.
- Leaves of tree correspond to suffixes of D.

7

Construct a Suffix Array (A)

D =

0	1	2	3	4	5	6	7	8	9	10
a	c	a	g	a	c	c	a	g	a	\$

Position	Suffix		A	Suffix
10	\$	Sort by suffix ➔	10	\$
9	a\$		9	a\$
8	ga\$		0	acagaccaga\$
7	aga\$		4	accaga\$
6	caga\$		7	aga\$
5	ccaga\$		2	agaccaga\$
4	accaga\$		6	caga\$
3	gaccaga\$		1	cagaccaga\$
2	agaccaga\$		5	ccaga\$
1	cagaccaga\$		8	ga\$
0	acagaccaga\$		3	gaccaga\$

8

Suffix Tree Example

D =

0	1	2	3	4	5	6	7	8	9	10
a	c	a	g	a	c	c	a	g	a	\$

A	Suffix
10	\$
9	a\$
0	aca...
4	acc...
7	aga\$
2	agac...
6	caga\$
1	cagac...
5	cca...
8	ga\$
3	gac...

9

Rapid Matching vs Suffix Tree

Where is cag?

- Can find all matches to a read in D using time proportional to read length s.

10

Rapid Matching vs Suffix Tree

Where is cag?

- Can find all matches to a read in D using time proportional to read length s.

11

Rapid Matching vs Suffix Tree

Where is cag?

- Can find all matches to a read in D using time proportional to read length s.

12

Rapid Matching vs Suffix Tree

Where is cag?

- Can find all matches to a read in D using time proportional to read length s.

D = acagaccaga\$

13

Extension to Inexact Matching

- To permit matches with k **substitutions**, try multiple paths, but charge for each mismatch. (Try searching for "aca" with one mismatch.)
- To permit matches with k **differences**, can do dynamic programming or heuristic search to compute edit distance of read against many paths in tree.
- Descent stops for a read when we hit bottom of tree or find that path requires > k differences.

14

Spliced read mapping (TopHat)

Map unspliced reads

Initially Unmapped (IUM) reads

IUM

Split IUM into segments

S1 S2 S3

S2 part 1 S2 part 2

Junction flanking index

15

Burrows-Wheeler Transform (BWT)

D = acagaccaga\$

Suffix Array	Burrows-Wheeler Matrix
10 \$	10 \$acagaccaga
9 a\$	9 a\$acagaccag
0 acagaccaga\$	0 acagaccaga\$
4 accaga\$	4 accaga\$acag
7 aga\$	7 aga\$acagacc
2 agaccaga\$	2 agaccaga\$ac
6 caga\$	6 caga\$acagac
1 cagaccaga\$	1 cagaccaga\$a
5 ccaga\$	5 ccaga\$acaga
8 ga\$	8 ga\$acagacca
3 gaccaga\$	3 gaccaga\$aca

BWT: aaaaccg\$ga

16

"Virtual Tree": FM-Index

- Suffix trees need 10-100 bytes memory/base indexed
- Theorem** [Ferragina & Manzini, 2000]: we can build an $O(1)$ -time oracle that, given "suffix tree posn" corresponding to a string z, returns posn corresponding to one-char extension z.c
- Hence, we can simulate suffix tree matching without explicitly storing the tree!
- Space cost: ~1 byte/base

17

Commonly Used Mapping Software

- Bowtie** [Langmead *et al.* 2009, 2012]
- BWA** [Li & Durbin 2009, 2010]
- SOAP2** [Li *et al.* 2009]

All these tools use variants of FM-index approach.

18

On to Quantitation...

- See the "[RNA Quantitation from RNA-Seq Data](#)" PowerPoint presentation and lecture notes