Peter Edge
12/23/13
CSCI 5481
Final Project

BaseballCap: RNAseq junction read alignment using the Burrow-Wheeler Transform

Introduction to RNAseq technology

The set of RNA transcripts expressed at a given time represent the set of cellular commands being transmitted; RNA can specify proteins for the cell to produce (mRNA), as well as provide structure and function to the ribosomes (rRNA) as well as regulate existing transcript functionality (miRNA). As a result, the ability to monitor and quantify the transcriptome is of the utmost importance to basic research as well as for prediction of clinical outcomes of cancer.[1] For many years, microarrays were the dominant technology for transcriptome quantification, but in recent years RNAseq technology has rapidly become the new standard. There are numerous advantages to RNAseq over microarrays including the elimination of the task of probe design , higher resolution quantification,[2] and also determination of RNA splice patterns. RNAseq can be used to verify previously identified alternative gene transcripts as well as to discover entirely new splice patterns.

However, this application of RNAseq technology poses a new computational problem: previously known sequence alignment programs are not prepared to handle the presence of *junction reads*, or sequencing reads that do not map to the genome in a single piece. If a given piece of read results from two separate exons, then the read will not map to the genome by any traditional means. For this reason, alternative strategies are employed that rely on sequencing separate halves of the read to separate exons. In this project, one such strategy is used to map RNAseq junction reads.

The Burrow-Wheeler Transform

The Burrow-Wheeler Transform, introduced in 1994, has been used for a number of purposes such as data compression.[3] The transform is performed by placing a lexically minimal character at the end of the input string, and lexically sorting the set of permutations S where the x (x == 1..len) last characters are moved of the string are appended to the front of the string:

```
PETER$              $PETER      last
$PETER    sort      ER$PET      col
R$PETE    --->      ETER$P      --->   BWT('PETER') = 'RTP$EE'
ER$PET              PETER$
TER$PE              R$PETE
ETER$P              TER$PE
```

(note that the last column and the indices of letter-ranges in first column are the only data kept)
The transform has a Last-First (LF) property that states the *i*th occurrence of a character in the first column corresponds to the *i*th occurrence of the same character in the last column (meaning they are the same character occurrence in the original sequence).[4] As a result, any given substring can be

1 Vant Veer, L. J. et al. Gene expression profiling predicts clinical outcome of breast cancer. Nature 415, 530–536 (2002).
2 Wang, Z., Gerstein, M. & Snyder, M. RNA-Seq: a revolutionary tool for transcriptomics. Nat. Rev. Genet. 10, 57–63 (2009).
3 Burrows, M., Wheeler, D. J., Burrows, M. & Wheeler, D. J. A block-sorting lossless data compression algorithm. (1994).
4 Burrows et al

searched for in right-to-left fashion by repeatedly mapping “last” indices to “first” indices and progressively narrowing the possible rows that could carry the substring.

As a trivial example, it is possible locate the substring 'TER' in the string 'PETER' in the above example by noting the presence of one 'R' in the first column at index 4, which has the first-occuring 'E' in the last column. The LF property tells us that now we need only check the first-occuring 'E' in the first column, since the second-occuring 'E' fell outside of the range (since it is not in the set of 'E's with an 'R' in the first column, we already know it cannot be followed by an 'R'). Therefore, we check the first-occuring 'E' in the first column, to discover that it is indeed proceeded by a 'T' because it has a T in the last column.

This fashion of “LF Mapping” is incredibly powerful for rapid mapping of substrings to biological sequence, and appropriately has become commonplace in next-gen sequencing analysis tools. However, as stated before, traditional mapping strategies fail to map RNAseq junction reads, and BWT is no exception. As a result, BWT-based algorithms must be modified in order to successfully identify junction reads. The gold standard of RNAseq junction read alignment tools is the TopHat utility,[5] which uses a long and complicated complicated series of steps to quickly and accurately map junction reads to genomic sequence.

The BaseballCap Tool

BaseballCap (so named for being a “poor man's TopHat”) is a command line tool meant to perform a simple approximation of the renowned TopHat junction read alignment tool. BaseballCap is implemented in Java 7, and uses the Burrow-Wheeler Transform (BWT) along with the LF mapping property of BWT to perform alignments from either side of a set of fasta format short read to a set of fasta format exons. The algorithm is comprised of the following steps:

1. Scan in RNAseq reads and exon sequences
2. For every exon sequence e:
    - compute BWT(e) and BWT(reverse(e))
    - for every read sequence r, attempt to align r to e and reverse(r) to reverse(e) using the LF mapping property of BWT
    - save the partial alignments between ¼ read length and ¾ read length
3. Match reversed partial alignments (which were matched on the left side of a read, and therefore are candidates for the further 5' exon of a junction read) to forward partial alignments that matched to further 3' exons (which were matched on the right side of a read and are therefore candidates for the 3' exon of a junction read)
    - if the lengths of the partial reads sum to the length of the total read, then a junction read has been identified
4. Enumerate the number of reads that map between any given pair of exons and return this statistic for each pair of exons for which junction reads were identified, along with the average amount of overhang that the reads had on the ends of each exon.

Figure 1 provides a graphical view of the BaseballCap workflow and program design. Note that the algorithm works differently from the simple BWT example given earlier. Rather than expecting to map the entire read (thus terminating with a non-zero range of rows in which matches occur), it is expected that the range of possible exon-permutations will close, leading to a “partial alignment” that only aligned to a certain point before closure of the range. The tool limits the length of these partial alignments to between ¼ read length to ¾ read length to avoid saving an abundance of short partial alignments that are more likely to map incorrectly by chance. Mismatches and gaps are also ignored in order to keep the implementation relatively simple.

5 Trapnell, C., Pachter, L. & Salzberg, S. L. TopHat: discovering splice junctions with RNA-Seq. Bioinformatics 25, 1105–1111 (2009).

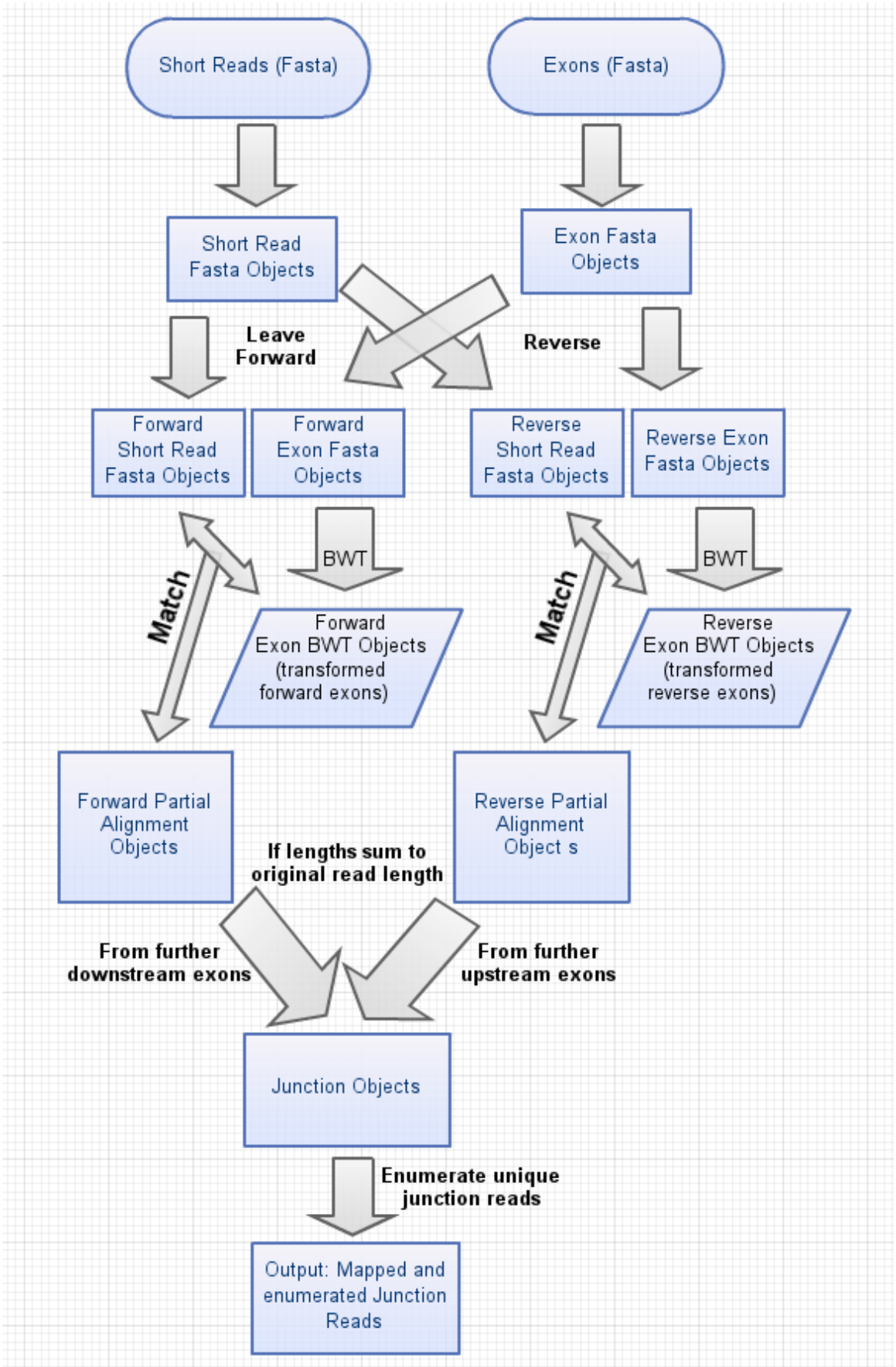

Figure 2: Diagram of the BaseballCap Junction Read Aligner. In the top three rows of boxes, data is handled with the Fasta class. The slanted boxes represent data transformed by the BWT constructor and held as a BWT object. Data in row 5 is held by the PartialAlignment class, and data in row 6 is held by the Junction class.

Methods

A set of exons were obtained from SDSC table browser[6] spanning the region from human chrX:146993469-147032647 . This is the locus of the FMR1(Fragile X mental retardation 1) gene, an RNA-binding-protein coding gene that is widely known to be associated with severe problems in brain development. The expansion of a trinucleotide repeat in the gene leads to a loss of function. As a result, the gene's protein product is unable to regulate RNA in developing neurons, which causes Fragile X syndrome.[7] There are a number of transcript variants, but the first transcript variant is used in this study to gauge the results of the BaseballCap tool and compare them to those obtained from TopHat.

An important consideration is that TopHat takes different input than BaseballCap; it maps junction reads to unbroken genomic sequence rather than exons represented as separated exon sequences. As a result, the problem solved by BaseballCap is fundamentally much less complicated as a result of this initial assumption.

First, RNAseq reads were simulated from a gene annotation file for FMR1 obtained from UCSC table browser using the RNASeqReadSimulator tool.[8] Since BaseballCap is not designed to handle mismatches and gaps, errors were set to 0 for simulated reads. Lengths 25, 50, 75, and 100 were obtained, numbered at the default 100000 each (note that these include all transcript variants in the FMR1 region). Then, these reads were uploaded to Galaxy[9] and mapped to the genomic sequence of the FMR1 region (also obtained from UCSC genome) using TopHat. Following mapping, the junctions identified for NM_002024 (transcript variant 1) were isolated. Then, BaseballCap was used to map the simulated NM_002024 reads to the NM_002024 exon set.[10] The junction reads identified by TopHat were compared to those identified by BaseballCap.

Obtaining FMR1 region data

Region of interest: FMRA1 region
chrX:146993469-147032647
Associated with Fragile X syndrome
Homo sapiens fragile X mental retardation 1 (FMR1), transcript variant ISO1, mRNA
NM_002024

Chromosome X reference retreived from UCSC:
`http://hgdownload.cse.ucsc.edu/goldenPath/hg19/chromosomes/chrX.fa.gz`

FMR1 gene annotation track retreived from UCSC table browser:

```
clade: Mammal
genome: Human
assembly: Feb.2009 GRCh37/hg19
group: Genes and Gene Prediction tracks
position: chrX:146993469-147032647
```

6 Karolchik D, Hinrichs AS, Furey TS, Roskin KM, Sugnet CW, Haussler D, Kent WJ. The UCSC Table Browser data retrieval tool. Nucleic Acids Res. 2004 Jan 1;32(Database issue):D493-6.

7 Hagerman, R. & Hagerman, P. Advances in clinical and molecular understanding of the FMR1 premutation and fragile X-associated tremor/ataxia syndrome. Lancet Neurol 12, 786–798 (2013).

8 RNASeqReadSimulator. http://www.cs.ucr.edu/~liw/rnaseqreadsimulator.html

9 Goecks, J, Nekrutenko, A, Taylor, J and The Galaxy Team. Galaxy: a comprehensive approach for supporting accessible, reproducible, and transparent computational research in the life sciences. Genome Biol. 2010 Aug 25;11(8):R86.

10 Karolchik et al

```
output : BED - browser extensible data
Create one bed entry per: whole gene
```

FMR1 exon fasta file obtained by:

```
clade: Mammal
genome: Human
assembly: Feb.2009 GRCh37/hg19
group: Genes and Gene Prediction tracks
position: chrX:146993469-147032647
output : sequence > genomic
5' UTR exons
CDS exons
3' UTR exons
NO introns
One Fasta record per region with 0 base upstream, 0 base
downstream
```

contiguousFMR1.fa was obtained similarly to above exon Fasta file, except:

- include introns
- one fasta record per gene

Generation of simulated RNAseq reads with RNAseqReadSimulator

Tool used to generate reads:

```
https://github.com/davidliwei/RNASeqReadSimulator
```

Commands to generate read bed files:

```
gensimreads.py -l 25 FMR1GeneAnnotationFile.bed > FMR1ReadsL25.bed
gensimreads.py -l 50 FMR1GeneAnnotationFile.bed > FMR1ReadsL50.bed
gensimreads.py -l 75 FMR1GeneAnnotationFile.bed > FMR1ReadsL75.bed
gensimreads.py -l 100 FMR1GeneAnnotationFile.bed >
FMR1ReadsL100.bed
```

Commands to generate read fasta files from bed files:

```
getseqfrombed.py FMR1ReadsL25.bed chrX.fa > FMR1ReadsL25.fasta
getseqfrombed.py FMR1ReadsL50.bed chrX.fa > FMR1ReadsL50.fasta
getseqfrombed.py FMR1ReadsL75.bed chrX.fa > FMR1ReadsL75.fasta
getseqfrombed.py FMR1ReadsL100.bed chrX.fa > FMR1ReadsL100.fasta
```

TopHat Junction Read Mapping in Galaxy:

Link to Galaxy History:
https://galaxy.msi.umn.edu/u/cs548111/h/final-project-1

TopHat requires fastq input, which is similar to fasta except with quality scores. First, it is necessary convert the fasta files generated with RNASeqReadSimulator into Fastq, which can be accomplished by “Combine FASTA and QUAL”, which assumes max quality (~) when Qual input is left out:

Combine FASTA and QUAL (version 1.0.1)

input in Galaxy History:

```
4: NM_002024SimulatedReadsLen100.fasta
```

```
3: NM_002024SimulatedReadsLen75.fasta
2: NM_002024SimulatedReadsLen50.fasta
1: NM_002024SimulatedReadsLen25.fasta
```

output in Galaxy History:

```
8: Combine FASTA and QUAL on data 4
7: Combine FASTA and QUAL on data 3
6: Combine FASTA and QUAL on data 2
5: Combine FASTA and QUAL on data 1
```

Tophat for Illumina (version 1.5.0)

Difficulty encountered:
Tophat wasn't able to accept a FASTA of seperate exons, because it recognizes each as different chromosomes. Instead, mapped against FMRA region with introns included (contiguousFMR1.fa):
input: 6-9 above

9: contiguousFMR1.fa (use a genome from history)
output:

```
25: Tophat for Illumina on data 8 and data 9: accepted_hits
24: Tophat for Illumina on data 8 and data 9: splice junctions
23: Tophat for Illumina on data 8 and data 9: deletions
22: Tophat for Illumina on data 8 and data 9: insertions
21: Tophat for Illumina on data 7 and data 9: accepted_hits
20: Tophat for Illumina on data 7 and data 9: splice junctions
19: Tophat for Illumina on data 7 and data 9: deletions
18: Tophat for Illumina on data 7 and data 9: insertions
17: Tophat for Illumina on data 6 and data 9: accepted_hits
16: Tophat for Illumina on data 6 and data 9: splice junctions
15: Tophat for Illumina on data 6 and data 9: deletions
14: Tophat for Illumina on data 6 and data 9: insertions
13: Tophat for Illumina on data 5 and data 9: accepted_hits
12: Tophat for Illumina on data 5 and data 9: splice junctions
11: Tophat for Illumina on data 5 and data 9: deletions
10: Tophat for Illumina on data 5 and data 9: insertions
```

Running BaseballCap on NM_002024 Exons

Running BaseballCap, included with this report, is accomplished using the following commands:

To compile the Java source (in main project directory):

```
make
```

Then, to run on a toy sample:

```
make toy
```

To run on the real NM_002024 data (replace XX with 25, 50, 75, 100 to specify read length of simulated FMR1 reads) :

```
make realXX
```

If make is not installed, then the program can be compiled manually using javac.
To run without make, simply copy the java commands specified by the desired make target in makefile.

The template for the makefile is based on a sample makefile.[11]

Results

When run on a toy dataset of exons and reads that form junctions between pairings of exons at various distance, TopHat idenfied the known junction reads aptly. For this reason, there can be confidence in the correctness of BaseballCap, insofar as computing what it was designed to. Performance and sensitivity on real world data, on the other hand, is different matter. Runtime is on the order of a few seconds for any of the NM_02024 datasets on a typical intel core i5 machine. Part of the reason that this analysis was chosen in favor of running on all FMR1 region exons (and parsing through exon headers to differentiate different exon sets) was that running on the entire set of 100000 reads on all FMR1 region exons did not finish in a reasonable amount of time. The size of the data sets is 8284 reads of length 25, 8267 reads of length 50, 12444 reads of length 75, and 12411 reads of length 100.

The results of BaseballCap on simulated NM_002024 RNAseq reads of lengths 25, 50, 75, and 100 are available in Supplementary Tables 1 - 4 (see pages following end of report). Supplementary tables 5 – 8 contain results from TopHat on the same reads. Junction boundaries were converted into pairs of exon numbers to make the results directly comparable to those of BaseballCap. One thing noticeable upon first inspection of the TopHat output compared to the BaseballCap output is that TopHat achieves a sensitivity that is impossible for BaseballCap simply by design; the TopHat results mapped Junction reads with overhang as large as ~95% of the read length. At the maximum, BaseballCap can detect reads with overhang 75% of read length (which means a minimum overhang of 25%) and anything above this is discarded. However, widening this range would result in a much larger number of partial alignments to be stored in memory.

Since the analysis is limited to the 17 exons belonging to one splice isoform, it is expected that each program should locate 16 splice junctions, between each successive exon. At length 50+, both BaseballCap and TopHat identified all 16 unique junctions and no incorrect junctions. However, both programs struggled to identify correct unique exons at length 25: TopHat identified too few (6) unique junctions (*Supp. Table 5*). BaseballCap, on the other hand, identified 15 of the 16 junctions (missing the junction from 11 to 12) *(Supp. Table 1)*. It also, however, identified 10 incorrect unique junctions (22 incorrect in all). These incorrect junctions were not identified at read length 50+. It is possible that these result from the fact that the algorithm does not strictly require reads to map to the edges of exons (it is presumed to be unlikely that two compatible partial reads with lengths that sum to exactly the read length would be a result of random chance. However, at read length 25 with an average of 12 bases mapping to either side (and a minimum overhang of 3/4*12 = 3 bases) this is entirely possible.

	Read len 25	Read len 50	Read len 75	Read len 100
BaseballCap	7.24	20.80	34.87	44.40
TopHat	29.33	73.13	124.06	185.88

Table 1: Average number of reads identified per unique junction. Note that because of the differences in read count between lengths, this is for comparison between programs only.

At all read lengths there is a drastic difference in the average number of reads identified per junction. In the absence of mismatches and gaps, BaseballCap should identify many of the same junctions as TopHat. One clear reason for this difference is the difference in sensitivity, mentioned above. Since only partial alignments in the range of ¼ read length to ½ read length (and their

11 http://www.cs.swarthmore.edu/~newhall/unixhelp/javamakefiles.html

corresponding partial alignments in the range ½ read length to ¾ read length) are identified, this means that reads in range of (0.05 * read length) to ¼ read length (and corresponding long partial alignments) are ignored. Since RNAseq reads are expected to be uniformly distributed along the length of the RNA, it is expected therefore that TopHat will identify almost twice as many junction reads as BaseballCap. The reality is that TopHat identified nearly four times as many reads per unique junction. The reason for this is unclear. TopHat has an immense number of optimizations, which is why it is such a powerful tool compared to the naïve approach that BaseballCap uses.

Conclusion

BaseballCap successfully maps RNAseq junction reads in fashion consistent with known splice patterns. The sensitivity suffers from the limitation of its limited partial alignment read length range, resulting in less reads identified per unique junction. There are number of ways in which the algorithm/program could be improved to be more sophisticated.

1. First, the algorithm could be modified to allow gaps and mismatches. This would allow the algorithm to be more applicable to real-life applications.

3. The algorithm could be generalized to take genomic sequence as input, map non-junction reads to their appropriate locations, and keep non-mappable reads as the partial alignments. Then these remaining partials could be paired to one another and used to hypothesize the presence of previously unseen splice patterns. The fact that the exons are a given is a significant assumption.

4. On the implementation level, the algorithm could acheive better time and space efficiency by transitioning to a language like C/C++ and using more arrays for data handling rather than this Java implementation's generous use of ArrayLists.

5. The algorithm does not differentiate between exons of different genes/transcript variants. The algorithm could be made to only predict junctions between exons of of the same gene/transcript, thereby allowing the massive analysis of pools of RNAseq data (assuming the necessary exon data is given).

The goal of this report was to demonstrate the power of BWT for mapping biological sequences to a reference, as well as its flexibility to be modified tackle new biological problems (in this case, junction reads). BaseballCap is just one example of this, but it is very likely that in coming years, BWT will find its way into implementations of algorithms designed to tackle altogether new problems that arise with advances in biotechnology.

Supplementary Figures

LeftExon	RightExon	AvgLeftOverhang	AvgRightOverhang	Count
1	2	11.333333	13.666667	12
1	7	8	17	1
1	8	17	8	1
1	11	8	17	1
1	16	8	17	1
2	3	11.909091	13.090909	11
3	4	12.5	12.5	10
4	5	12.428572	12.571428	7
4	8	17	8	1
4	17	17	8	4
5	6	12.933333	12.066667	15
6	7	12.692307	12.307693	13
7	8	11.727273	13.272727	11
8	9	14.428572	10.571428	7
8	17	16	9	3
9	10	12.833333	12.166667	6
9	17	17	8	1
10	11	12.666667	12.333333	12
12	13	12.444445	12.555555	18
13	14	12.166667	12.833333	6
13	17	16.5	8.5	6
14	15	12.666667	12.333333	15
14	17	16	9	3
15	16	12.5	12.5	8
16	17	11.875	13.125	8

Supp. Table 1: BaseballCap NM_002024 Junctions, length 25

LeftExon	RightExon	AvgLeftOverhang	AvgRightOverhang	Count
1	2	24.904762	25.095238	21
2	3	23.96875	26.03125	32
3	4	24.428572	25.571428	21
4	5	27.73913	22.26087	23
5	6	24.0625	25.9375	16
6	7	23.62069	26.37931	29
7	8	22.68421	27.31579	19
8	9	23.954546	26.045454	22
9	10	22.461538	27.538462	13
10	11	24.52381	25.47619	21
12	13	27.333334	22.666666	15
13	14	22.576923	27.423077	26
14	15	24.619047	25.380953	21
15	16	23.9	26.1	10
16	17	25.782608	24.217392	23

Supp. Table 2: BaseballCap NM_002024 Junctions, length 50

LeftExon	RightExon	AvgLeftOverhang	AvgRightOverhang	Count
1	2	39.720932	35.279068	43
2	3	37.848484	37.151516	33
3	4	35.51282	39.48718	39
4	5	36	39	31
5	6	37.60606	37.39394	33
6	7	36.833332	38.166668	30
7	8	35.775	39.225	40
8	9	41.26829	33.73171	41
9	10	35.54286	39.45714	35
10	11	38.228573	36.771427	35
12	13	39.465115	35.534885	43
13	14	38.62963	36.37037	27
14	15	37.37037	37.62963	27
15	16	42.347828	32.652172	23
16	17	37.744186	37.255814	43

Supp. Table 3: BaseballCap NM_002024 Junctions, length 75

LeftExon	RightExon	AvgLeftOverhang	AvgRightOverhang	Count
1	2	57.095238	42.904762	21
2	3	42.1	57.9	30
3	4	51.262295	48.737705	61
4	5	48.785713	51.214287	42
5	6	49.116665	50.883335	60
6	7	49.901962	50.098038	51
7	8	51.104168	48.895832	48
8	9	48.037037	51.962963	54
9	10	50.923077	49.076923	39
10	11	48.2807	51.7193	57
12	13	42.97059	57.02941	34
13	14	48.642857	51.357143	56
14	15	48.47619	51.52381	42
15	16	37.6875	62.3125	16
16	17	45.890907	54.109093	55

Supp. Table 4: BaseballCap NM_002024 Junctions, length 100

junction	LeftExon	RightExon	MaxLeftOverhang	MaxRightOverhang	count
JUNC00000001	2	3	21	19	33
JUNC00000002	4	5	21	19	24
JUNC00000003	8	9	21	20	24
JUNC00000004	10	11	20	20	30
JUNC00000005	11	12	20	21	35
JUNC00000006	13	14	21	19	30

Supp. Table 5: TopHat NM_002024 Junctions, length 25
Adapted from output .bed file. Left corresponds to furthest 5' mapped base that spanned junction. Right corresponds to furthest right 3' mapped base that spanned junction.

junction	LeftExon	RightExon	MaxLeftOverhang	MaxRightOverhang	count
JUNC00000001	1	2	42	44	66
JUNC00000002	2	3	46	44	82
JUNC00000003	3	4	42	43	76
JUNC00000004	4	5	46	43	82
JUNC00000005	5	6	44	46	81
JUNC00000006	6	7	47	40	74
JUNC00000007	7	8	47	43	74
JUNC00000008	8	9	46	45	77
JUNC00000009	9	10	43	42	44
JUNC00000010	10	11	45	45	73
JUNC00000011	11	12	45	46	81
JUNC00000012	12	13	46	42	69
JUNC00000013	13	14	46	44	87
JUNC00000014	14	15	42	41	70
JUNC00000015	15	16	45	42	63
JUNC00000016	16	17	45	43	71

Supp. Table 6: TopHat NM_002024 Junctions, length 50
Adapted from output .bed file. Left corresponds to furthest 5' mapped base that spanned junction. Right corresponds to furthest right 3' mapped base that spanned junction.

junction	LeftExon	RightExon	MaxLeftOverhang	MaxRightOverhang	count
JUNC00000001	1	2	67	53	128
JUNC00000002	2	3	55	69	139
JUNC00000003	3	4	67	68	126
JUNC00000004	4	5	70	69	122
JUNC00000005	5	6	69	71	115
JUNC00000006	6	7	72	65	115
JUNC00000007	7	8	72	68	129
JUNC00000008	8	9	71	69	131
JUNC00000009	9	10	69	67	112
JUNC00000010	10	11	70	70	124
JUNC00000011	11	12	70	63	125
JUNC00000012	12	13	64	67	152
JUNC00000013	13	14	71	69	117
JUNC00000014	14	15	65	70	104
JUNC00000015	15	16	69	65	98
JUNC00000016	16	17	70	68	148

Supp. Table 5: TopHat NM_002024 Junctions, length 75
Adapted from output .bed file. Left corresponds to furthest 5' mapped base that spanned junction. Right corresponds to furthest right 3' mapped base that spanned junction.

junction	LeftExon	RightExon	MaxLeftOverhang	MaxRightOverhang	count
JUNC00000001	1	2	92	53	177
JUNC00000002	2	3	55	94	185
JUNC00000003	3	4	94	72	192
JUNC00000004	4	5	74	94	177
JUNC00000005	5	6	94	94	202
JUNC00000006	6	7	94	90	179
JUNC00000007	7	8	97	93	178
JUNC00000008	8	9	96	80	190
JUNC00000009	9	10	81	92	184
JUNC00000010	10	11	94	95	197
JUNC00000011	11	12	95	64	192
JUNC00000012	12	13	64	88	203
JUNC00000013	13	14	90	94	188
JUNC00000014	14	15	92	95	180
JUNC00000015	15	16	95	83	162
JUNC00000016	16	17	85	93	188

Supp. Table 8: TopHat NM_002024 Junctions, length 100
Adapted from output .bed file. Left corresponds to furthest 5' mapped base that spanned junction. Right corresponds to furthest right 3' mapped base that spanned junction.

Works Cited

1. Vant Veer, L. J. et al. Gene expression profiling predicts clinical outcome of breast cancer. Nature 415, 530–536 (2002).

2. Wang, Z., Gerstein, M. & Snyder, M. RNA-Seq: a revolutionary tool for transcriptomics. Nat. Rev. Genet. 10, 57–63 (2009).

3. Wang, Z., Gerstein, M. & Snyder, M. RNA-Seq: a revolutionary tool for transcriptomics. Nat. Rev. Genet. 10, 57–63 (2009).

4. Trapnell, C., Pachter, L. & Salzberg, S. L. TopHat: discovering splice junctions with RNA-Seq. Bioinformatics 25, 1105–1111 (2009).

5. Karolchik D, Hinrichs AS, Furey TS, Roskin KM, Sugnet CW, Haussler D, Kent WJ. The UCSC Table Browser data retrieval tool. Nucleic Acids Res. 2004 Jan 1;32(Database issue):D493-6.

6. Hagerman, R. & Hagerman, P. Advances in clinical and molecular understanding of the FMR1 premutation and fragile X-associated tremor/ataxia syndrome. Lancet Neurol 12, 786–798 (2013).

7. RNASeqReadSimulator. http://www.cs.ucr.edu/~liw/rnaseqreadsimulator.html

8. Goecks, J, Nekrutenko, A, Taylor, J and The Galaxy Team. Galaxy: a comprehensive approach for supporting accessible, reproducible, and transparent computational research in the life sciences. Genome Biol. 2010 Aug 25;11(8):R86.

9. http://www.cs.swarthmore.edu/~newhall/unixhelp/javamakefiles.html