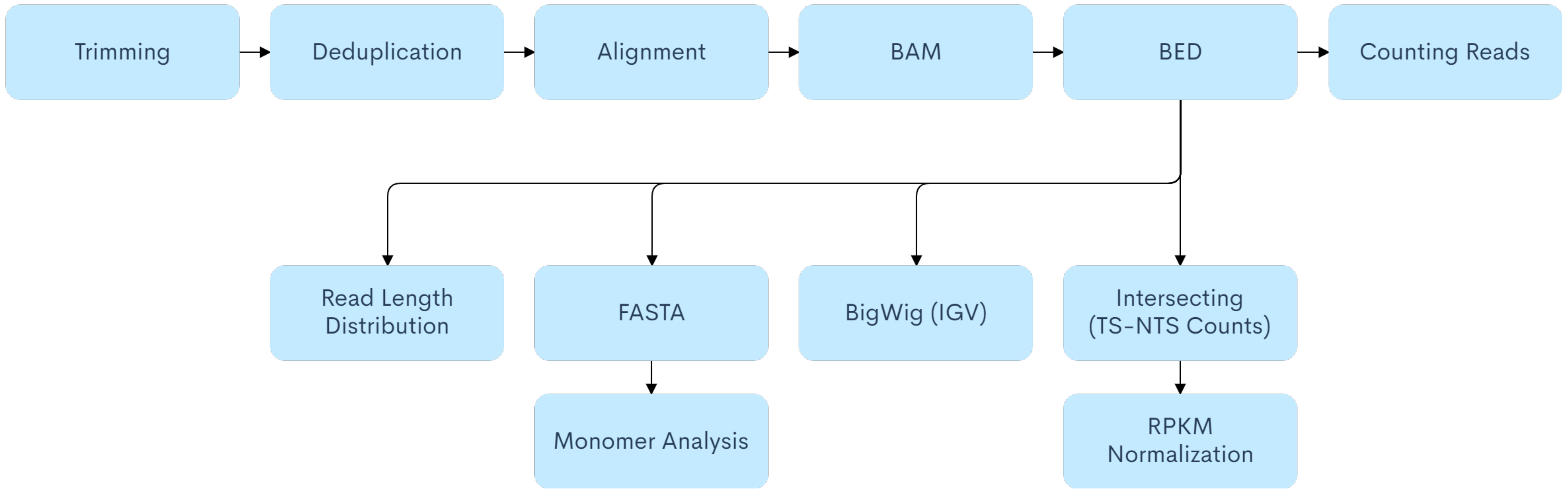
Bioinformatic Analysis of XR-Seq

Saygin Gulec

Overview

- Explanation of the steps
- Automating this process
- Future implementations

Workflow


Trimming
Deduplication
Alignment
BAM
BED
Counting Reads
Read Length Distribution
FASTA
BigWig (IGV)
Intersecting (TS-NTS Counts)
Monomer Analysis
RPKM Normalization

Adaptor Trimming

The samples are unzipped and the adaptor sequences are trimmed with Cutadapt.

```
cutadapt -a [adaptor] --discard-untrimmed -m 13 –M 27 –o [output].fastq
[input].fastq.gz
```

-a	Adaptor sequence.
-m	Minimum allowed read length.
-M	Maximum allowed read length.
-o	Name of the output fastq file.
	Input fastq.gz file.

Deduplication

Duplicate reads are removed with FASTX-Toolkit and the file format is converted from FASTQ to FASTA.

```
fastx_collapser -v -i [input].fastq -o [output].fasta -Q33
```

• -v	Verbose.
• -i	Input fastq.
• -o	Output fasta.
• -Q33	To tell it that you're using Illumina encoded quality scores, not Sanger encoding.

Genome Alignment

The reads are aligned to the genome with Bowtie2.

```
bowtie2 -x [Bowtie2 Index] -f [input].fasta -S [output].sam
```

• -x	Bowtie2 Index i.e. Bowtie2Index/WBcel235
• -f	Input FASTA file.
• -S	Output SAM file.

Converting from SAM to BAM

The SAM file is converted to BAM format with Samtools and sorted.

```
samtools sort -o [output].bam [input].sam
```

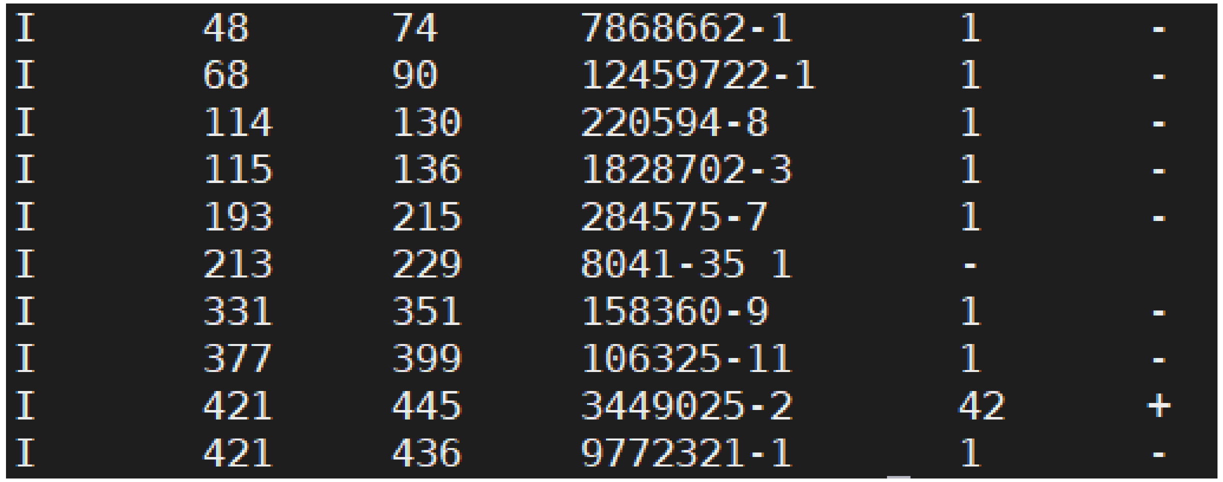
- -o	Output BAM file.
- 	Input SAM file.

- BAM is smaller than SAM and is faster to process because it is in binary format.

Converting from BAM to BED

The BAM file is converted to BED for downstream analysis.

bedtools bamtobed -i [input].bam > [output].bed



```
I       48      74      7868662-1       1       -
I       68      90      12459722-1      1       -
I       114     130     220594-8        1       -
I       115     136     1828702-3       1       -
I       193     215     284575-7        1       -
I       213     229     8041-35 1       -
I       331     351     158360-9        1       -
I       377     399     106325-11       1       -
I       421     445     3449025-2       42      +
I       421     436     9772321-1       1       -
```

Counting Total Mapped Reads

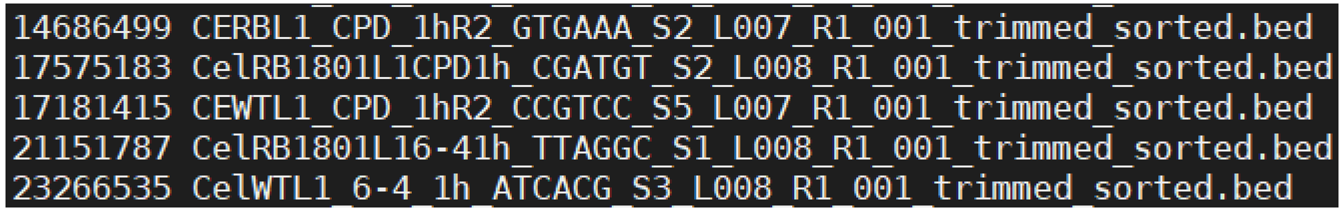
The total number of mapped reads is counted for RPKM normalization with built-in Linux commands grep and wc.

For calculation:

grep -c \"^\" [input].bed > [sample name]_readCount.txt

For displaying in results:

wc -l [input].bed >> results/total_mapped_reads.txt

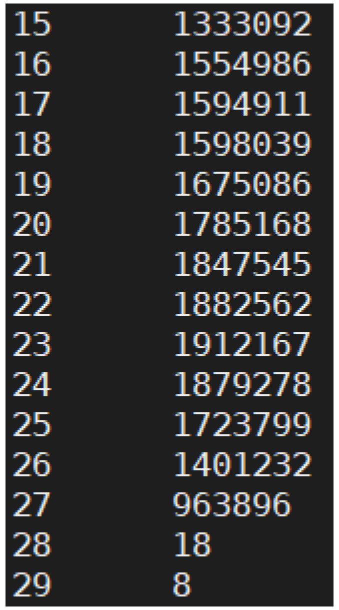


```
14686499 CERBL1_CPD_1hR2_GTGAAA_S2_L007_R1_001_trimmed_sorted.bed
17575183 CelRB1801L1CPD1h_CGATGT_S2_L008_R1_001_trimmed_sorted.bed
17181415 CEWTL1_CPD_1hR2_CCGTCC_S5_L007_R1_001_trimmed_sorted.bed
21151787 CelRB1801L16-41h_TTAGGC_S1_L008_R1_001_trimmed_sorted.bed
23266535 CelWTL1_6-4_1h_ATCACG_S3_L008_R1_001_trimmed_sorted.bed
```

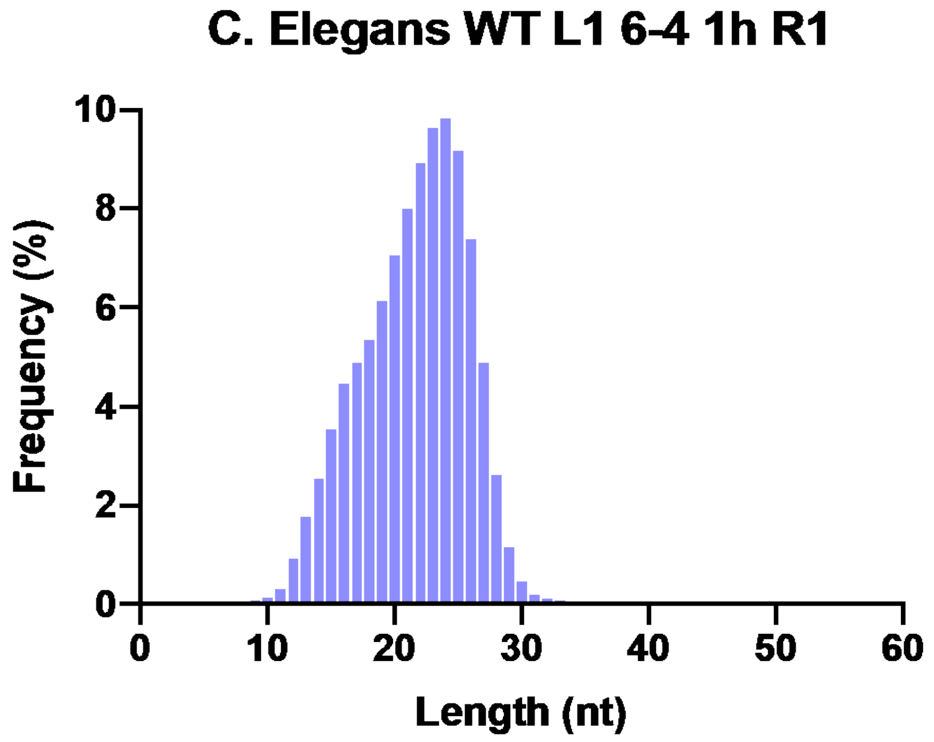
Read Length Distribution

A table containing the numbers of reads with certain lengths is generated with built-in Linux commands awk, sort, uniq and sed.

```
awk '{print \$3-\$2}’ [sample].bed | sort -k1,1n | uniq -c | sed 's/\s\s*/ /g' | awk '{print \$2\"\t\"\$1}’ >
[sample]_read_length_distribution.txt
```

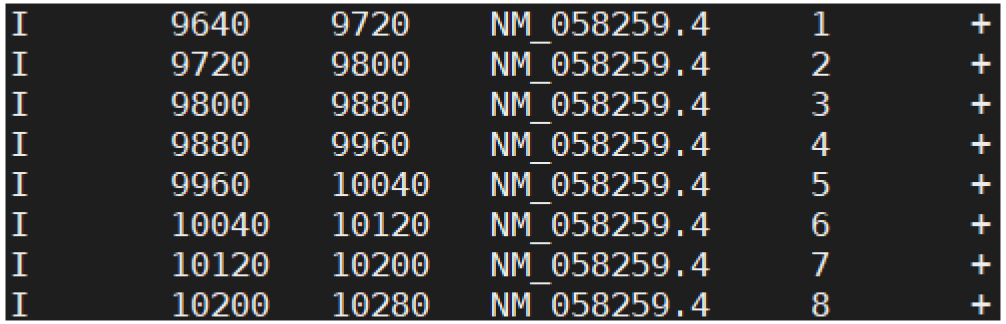


```
15      1333092
16      1554986
17      1594911
18      1598039
19      1675086
20      1785168
21      1847545
22      1882562
23      1912167
24      1879278
25      1723799
26      1401232
27      963896
28      18
29      8
```



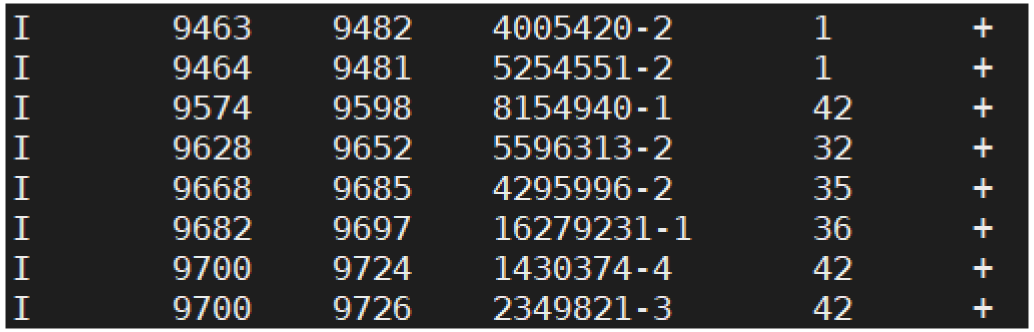
Counting the Number of Reads in Genes

Gene List



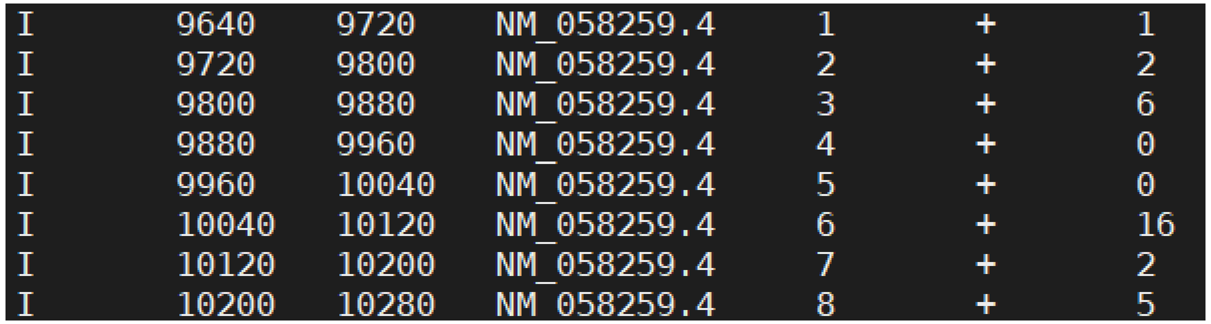
```
I       9640    9720    NM_058259.4     1       +
I       9720    9800    NM_058259.4     2       +
I       9800    9880    NM_058259.4     3       +
I       9880    9960    NM_058259.4     4       +
I       9960    10040   NM_058259.4     5       +
I       10040   10120   NM_058259.4     6       +
I       10120   10200   NM_058259.4     7       +
I       10200   10280   NM_058259.4     8       +
```

Excision Products



```
I       9463    9482    4005420-2       1       +
I       9464    9481    5254551-2       1       +
I       9574    9598    8154940-1       42      +
I       9628    9652    5596313-2       32      +
I       9668    9685    4295996-2       35      +
I       9682    9697    16279231-1      36      +
I       9700    9724    1430374-4       42      +
I       9700    9726    2349821-3       42      +
```

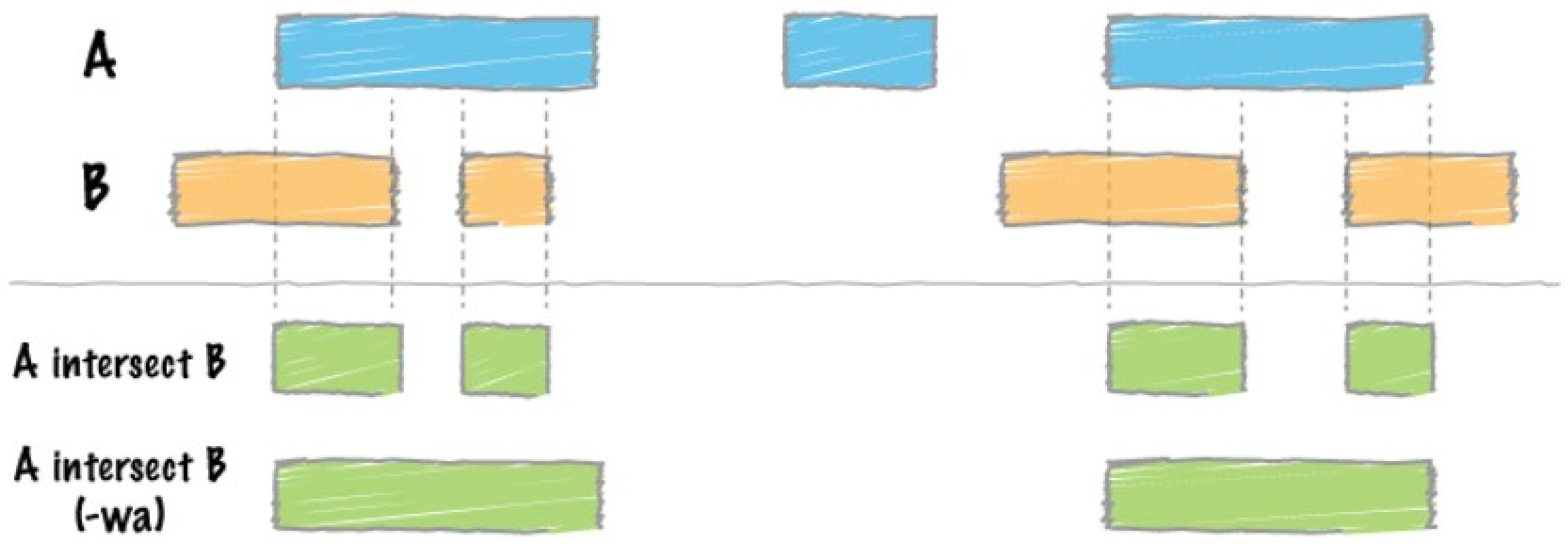
Number of Reads in Each Gene/Region



```
I       9640    9720    NM_058259.4     1       +       1
I       9720    9800    NM_058259.4     2       +       2
I       9800    9880    NM_058259.4     3       +       6
I       9880    9960    NM_058259.4     4       +       0
I       9960    10040   NM_058259.4     5       +       0
I       10040   10120   NM_058259.4     6       +       16
I       10120   10200   NM_058259.4     7       +       2
I       10200   10280   NM_058259.4     8       +       5
```

Counting the Number of Reads in Genes

The reads are intersected with the given gene list using Bedtools.

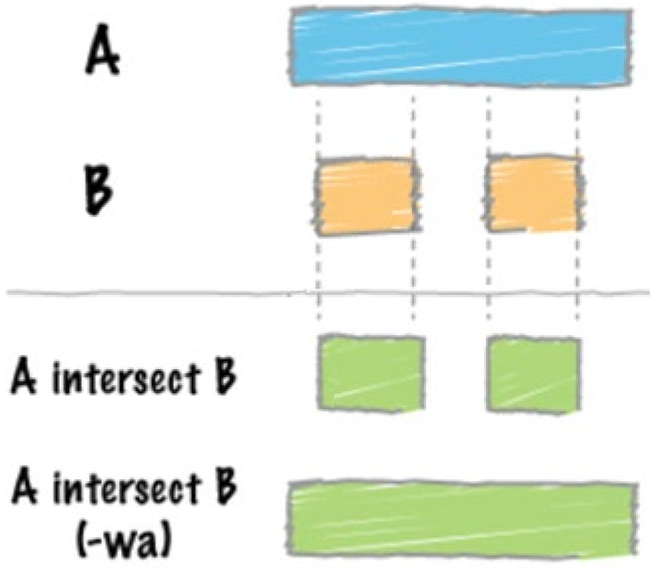


Counting the Number of Reads in Genes

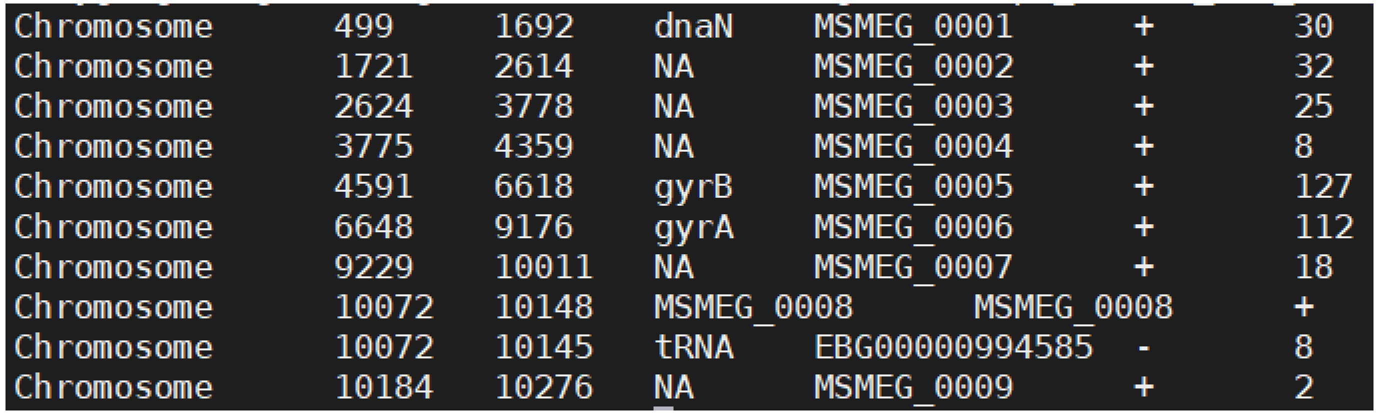
The reads are intersected with the given gene list using Bedtools.

```
bedtools intersect -c -a [Gene List] -b [input].bam -wa -s -F 0.5 > [output]_NTS.bed
bedtools intersect -c -a [Gene List] -b [input].bam -wa -S -F 0.5 > [output]_TS.bed
```

-c	For each entry in A, report the number of hits in B.
-a	File A. Each feature in A is compared to B in search of overlaps.
-b	File B.
-wa	Write the original entry in A for each overlap.
-s	Only report hits in B that overlap A on the same strand.
-S	Only report hits in B that overlap A on the opposite strand.
-F	Minimum overlap required as a fraction of B.

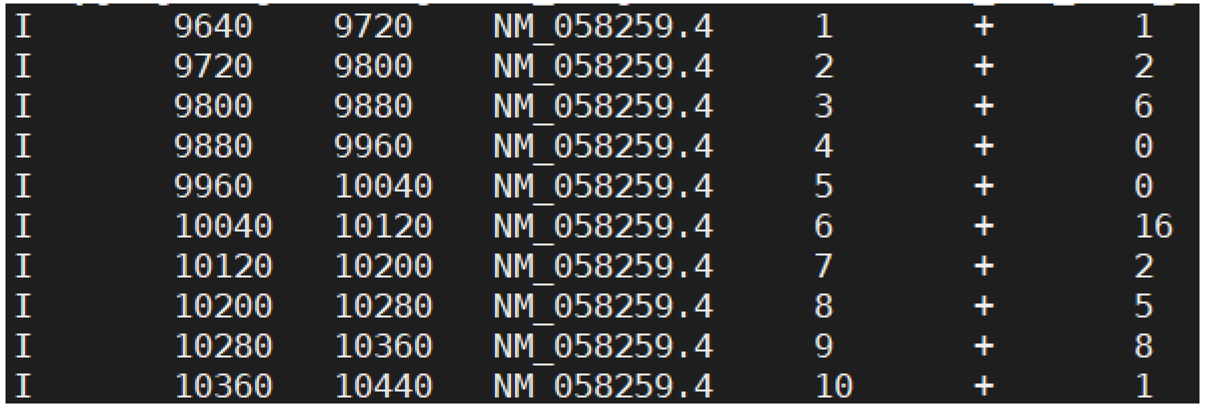


Whole Genes

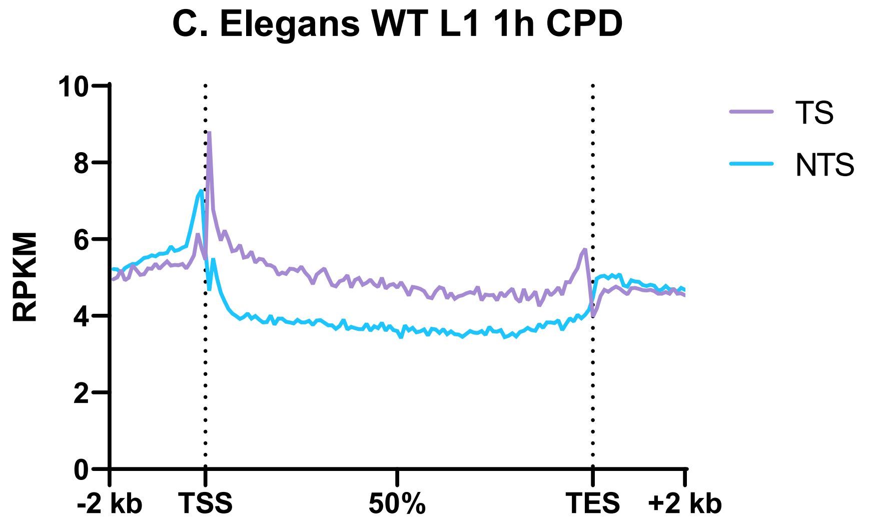


```
Chromosome      499     1692    dnaN    MSMEG_0001      +       30
Chromosome      1721    2614    NA      MSMEG_0002      +       32
Chromosome      2624    3778    NA      MSMEG_0003      +       25
Chromosome      3775    4359    NA      MSMEG_0004      +       8
Chromosome      4591    6618    gyrB    MSMEG_0005      +       127
Chromosome      6648    9176    gyrA    MSMEG_0006      +       112
Chromosome      9229    10011   NA      MSMEG_0007      +       18
Chromosome      10072   10148   MSMEG_0008      MSMEG_0008      +
Chromosome      10072   10145   tRNA    EBG00000994585  -       8
Chromosome      10184   10276   NA      MSMEG_0009      +       2
```

or Gene Regions



```
I       9640    9720    NM_058259.4     1       +       1
I       9720    9800    NM_058259.4     2       +       2
I       9800    9880    NM_058259.4     3       +       6
I       9880    9960    NM_058259.4     4       +       0
I       9960    10040   NM_058259.4     5       +       0
I       10040   10120   NM_058259.4     6       +       16
I       10120   10200   NM_058259.4     7       +       2
I       10200   10280   NM_058259.4     8       +       5
I       10280   10360   NM_058259.4     9       +       8
I       10360   10440   NM_058259.4     10      +       1
```



Generating BigWig Files

The BedGraph files are generated from BED files to make BigWigs with Bedtools.

```
bedtools genomecov -i [input].bed -g [genome].fai -bg -strand + -scale [total mapped reads / 10^7] > [output]_plus.bedGraph
bedtools genomecov -i [input].bed -g [genome].fai -bg -strand - -scale [total mapped reads / 10^7] > [output]_minus.bedGraph
```

-i	Input BED file.
-g	Chromosome sizes.
-bg	Output as BedGraph file.
-strand	Specify strand.
-scale	Scale by some number.

Generating BigWig Files

The BedGraph files are sorted to make BigWigs with built-in Linux command sort.

```
sort -k1,1 -k2,2n [sample]_plus.bedGraph > [sample]_plus_sorted.bedGraph
sort -k1,1 -k2,2n [sample]_minus.bedGraph > [sample]_minus_sorted.bedGraph
```

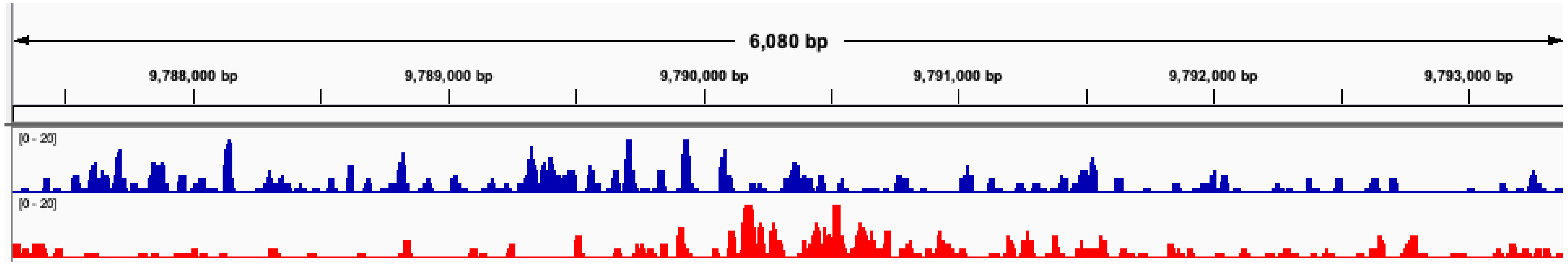
This command sorts the file by chromosome and then by position.

Generating BigWig Files

BigWig files are generated from sorted BedGraph files using UCSCTools.

```
bedGraphToBigWig [sample]_plus_sorted.bedGraph [genome].fai [sample]_plus.bw
bedGraphToBigWig [sample]_plus_sorted.bedGraph [genome].fai [sample]_plus.bw
```

These are then visualized using IGV.

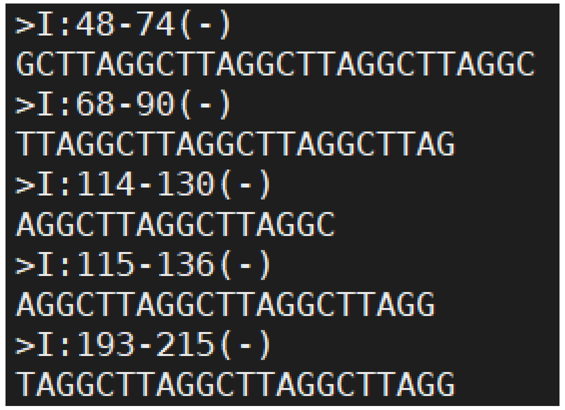


Generating FASTA of Aligned and Filtered Reads for Monomer Analysis

The sequences of the reads are obtained by generating a FASTA file from their BED file with Bedtools.

bedtools getfasta -s -fi [genome].fa -bed [reads].bed -fo [sequences].fa

-s	Gives the reverse complement for reads on the - strand.
-fi	Reference FASTA file.
-bed	Output BED file.
-fo	Output FASTA file.

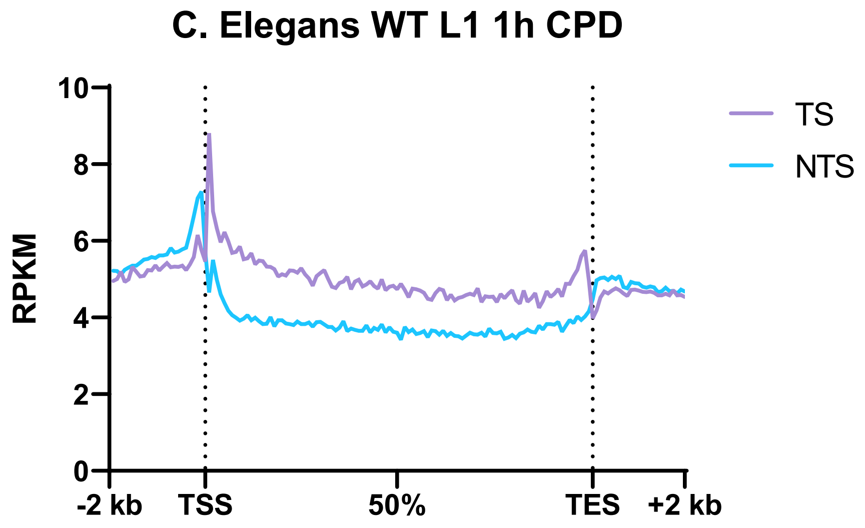


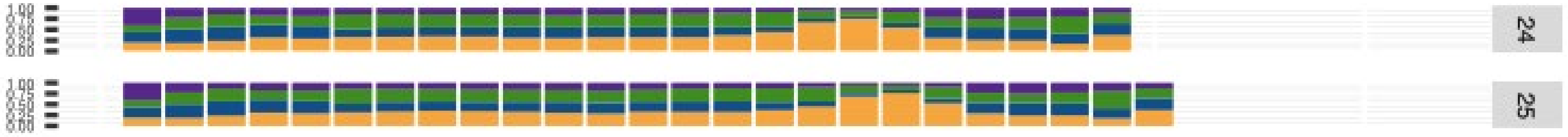
```
>I:48-74(-)
GCTTAGGCTTAGGCTTAGGCTTAGGC
>I:68-90(-)
TTAGGCTTAGGCTTAGGCTTAG
>I:114-130(-)
AGGCTTAGGCTTAGGC
>I:115-136(-)
AGGCTTAGGCTTAGGCTTAGG
>I:193-215(-)
TAGGCTTAGGCTTAGGCTTAGG
```

Custom Scripts

Custom Python scripts are used for:

- Pinpointing damage sites
- RPKM normalization
- Calculating average read counts for the TCR graph
- Monomer analysis

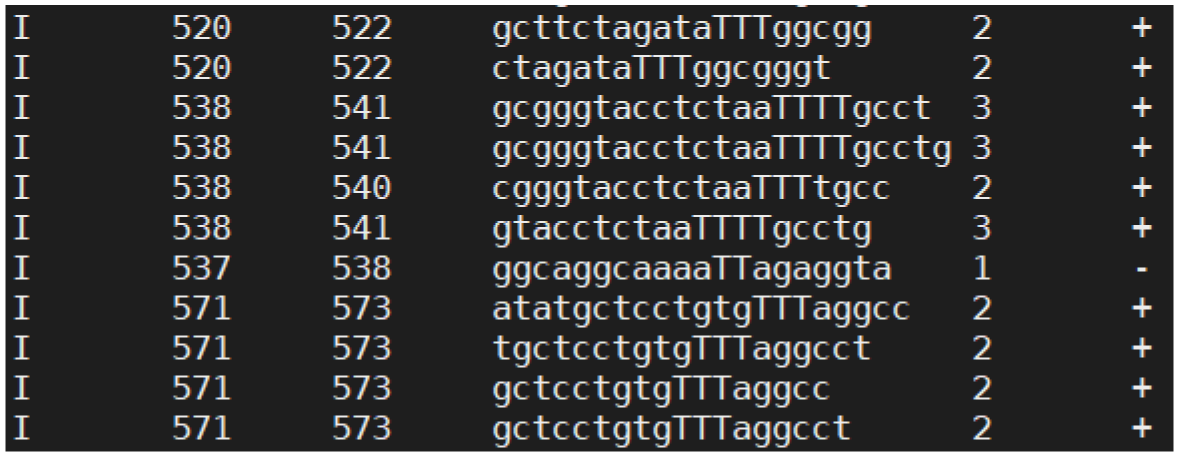




Pinpointing Damage Sites

Instead of using the whole excision product as the damage site, we can pinpoint where the damaged dimers themselves are.

This also allows us to filter the reads for higher stringency.



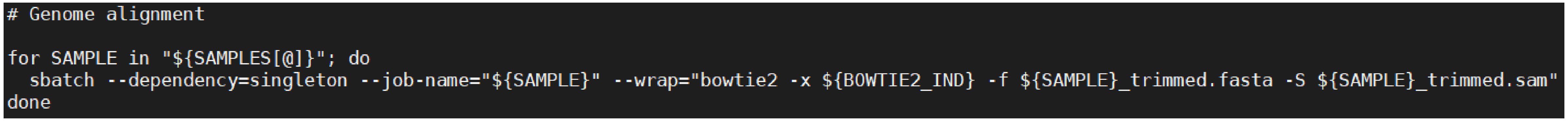
```
I       520     522     gcttctagataTTTggcgg     2       +
I       520     522     ctagataTTTggcgggt       2       +
I       538     541     gcgggtacctctaaTTTTgcct  3       +
I       538     541     gcgggtacctctaaTTTTgcctg 3       +
I       538     540     cgggtacctctaaTTTtgcc    2       +
I       538     541     gtacctctaaTTTTgcctg     3       +
I       537     538     ggcaggcaaaaTTagaggta    1       -
I       571     573     atatgctcctgtgTTTaggcc   2       +
I       571     573     tgctcctgtgTTTaggcct     2       +
I       571     573     gctcctgtgTTTaggcc       2       +
I       571     573     gctcctgtgTTTaggcct      2       +
```

Automating This Process

We don’t have to fill in all these brackets by hand.

Shell scripting can be used for these commands to be chained.

SLURM allows us to put the steps in line.



```
# Genome alignment

for SAMPLE in "${SAMPLES[@]}"; do
  sbatch --dependency=singleton --job-name="${SAMPLE}" --wrap="bowtie2 -x ${BOWTIE2_IND} -f ${SAMPLE}_trimmed.fasta -S ${SAMPLE}_trimmed.sam"
done
```

XR_Seq.sh

```
bash XR_Seq.sh -d ./ -b [Bowtie2 Index] -l [Gene List] –g [Genome]
```

What you need to provide:

• Samples from HTSF	CEWTL1_CPD_1hR2_CCGTCC_S5_L007.fastq.gz
• Bowtie2Index	Bowtie2Index/WBcel235
• Genome	Caenorhabditis_elegans.WBcel235.fa
• Gene List	ce11_150bin.bed
• Scripts	XR_Seq.sh
	XR_Seq.py

Filter by Read Length

```
bash XR_Seq.sh -d ./ -b [Bowtie2 Index] -l [Gene List] -g [Genome] -m 13 -M 27
```

-m	Minimum allowed read length.
-M	Maximum allowed read length.

Default: At least 1 nucleotide long.

Lengths Used in Monomer Analysis

```
bash XR_Seq.sh -d ./ -b [Bowtie2 Index] -l [Gene List] -g [Genome] -m 23 -M 29
--mon_min 26 --mon_max 28
```

• --mon_min	Minimum length to be used in monomer analysis.
• --mon_max	Maximum length to be used in monomer analysis.

Default: 10 – 30 nucleotides long.

If -m or -M is specified, this number is used instead.

This can be overridden by giving --mon_min and --mon_max after -m and –M.

Pinpointing Damage

```
bash XR_Seq.sh -d ./ -b [Bowtie2 Index] -l [Gene List] -g [Genome] -m 13 -M
27 --mon_min 10 --mon_max 30 -p --dimers TC,CT --lower 9 --upper 8
```

• -p	Pinpoint damage sites.
• --dimers	Dimers that get damaged.
• --lower	Maximum distance of the damage site from 3’ end.
• --upper	Minimum distance of the damage site from 3’ end.

Default: TT dimers 8-9 bp from 3’ end.

Consider Filtered Out Reads

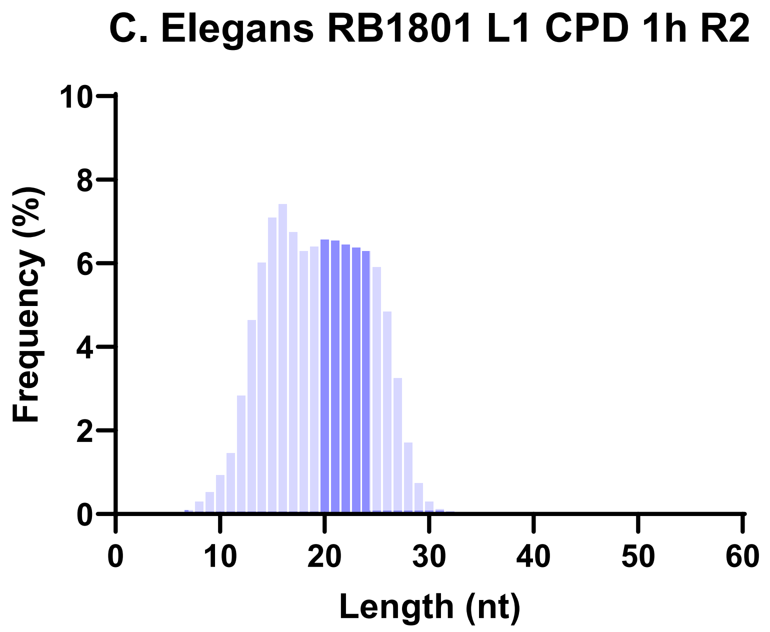
- -w Use the total number of oligomers mapped to the genome as the scaling factor instead of what is left after filtering.

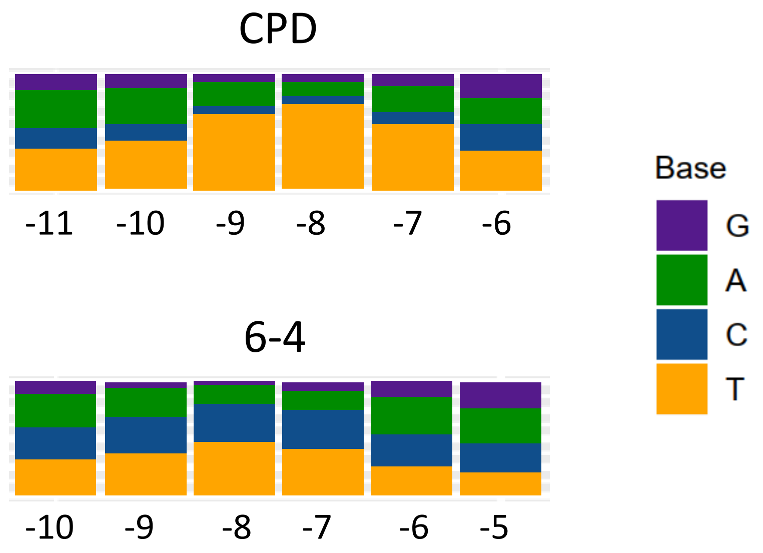
$$RPKM = \frac{number\ of\ reads\ of\ the\ region}{\frac{total\ reads}{1{,}000{,}000}\ x\ \frac{region\ length}{1{,}000}}$$

Read Count	Sample 1	Sample 2
Total Mapped	30,000,000	60,000,000
After Filtering	15,000,000	30,000,000
After Pinpointing	10,000,000	5,000,000

Quality Check

- The script always reports these for all mapped reads:
  - Read length distribution
  - Monomer analysis





Acknowledgements

- Sancar, Aziz
- Lindsey-Boltz, Laura
- Yanyan Yang
- Cansu Kose