Interval representation

- binning \( \rightarrow \) redundantly storing data at different zoom levels - originally implemented in UCSC genome browser (also used in BAM and BedTools)
- A different option \( \rightarrow \) interval tree, usually supported by programming languages
- For intervals that are very similar in size or have other constraints: a sort + binary search may also work well

BedTools

- High performance software package that operates on multiple interval oriented data formats: BED, GFF, SAM, BAM and VCF
- Download and install bedtools

\[
\text{http://code.google.com/p/bedtools/}
\]

BedTools concepts

- There are many (25 and growing) tools with different names
- Most tools write to the standard output
- The \(-\) (minus) character specifies the standard input
- Can be chained with \textit{pipes} like all UNIX commands
- Most tools write their help when invoked, others need \(-h\) flag
- Flag options can substantially change the output format
BedTools has an excellent user manual

1. OVERVIEW
   1.1 BACKGROUND
   1.2 SUMMARY OF AVAILABLE TOOLS
   1.3 FUNDAMENTAL CONCEPTS REGARDING BEDTOOLS USAGE
      1.3.1 What are genome features and how are they represented?
      1.3.2 Overlapping / intersecting features
      1.3.3 Comparing features in file “A” and file “B”
      1.3.4 BED starts are zero-based and BED ends are one-based
      1.3.5 GFF starts and ends are one-based
      1.3.6 VCF coordinates are one-based
      1.3.7 File B is loaded into memory
      1.3.8 Feature files must be tab-delimited
      1.3.9 All BEDTools allow features to be “piped” via standard input
      1.3.10 Most BEDTools write their results to standard output
      1.3.11 What is a “genome” file?

Two identical ways to invoke

- The old style mode contains a different tool for each task (the manual covers these tools):
  - intersectBed
  - windowBed
  - closestBed

- A new style mode that contains only one tool that takes commands like samtools:
  - bedtools intersect
  - bedtools window
  - bedtools closest

Basic concepts

- For any operation that requires two files the tools asks for file A and file B
- Each element in file A is matched against each element in file B
- File B is loaded into memory – try to make that the smaller file
  (if one of the files is a BAM file then use -abam as flag and that usually changes the output format)

BedTools main runner
Transform from BAM to bedfile

invoke the help

```
$ /bin/bedtools bamtobed -b
```

Tool: bedtools bamtobed (aka bamToBed)

Version: v2.16.2

Summary: Converts BAM alignments to BED6 or BEDPE format.

Usage: bedtools bamtobed [OPTIONS] < bam

Options:
- `bedpe` Write BEDPE format.
  - Requires BAM to be grouped or sorted by query.
- `bed12` Write "blocked" BED format (aka "BED12").

http://genome-test.ucsc.edu/FAQ/FAQformat#format1

-spli Report "split" BAM alignments as separate BED entries.

Extra options can further customize the output

options that alter the output format

Feature coverage

```
$ cat features.gff | awk '{ print $0 }' > simple.gff
```

Make a simpler feature file

Replace column 9 with a dot

The tool help describes the results

```
-spli Report "split" BAM alignments as separate BED entries.
```

Default Output:
After each entry in B, reports:
1) The number of features in A that overlapped the B interval.
2) The number of bases in B that had non-zero coverage.
3) The length of the entry in B.
4) The fraction of bases in B that had non-zero coverage.

-spli Report "split" BAM alignments as separate BED entries.

Extra options can further customize the output

Invoke the help

Transform to BED, produces the 5’ end

```
$ /bin/bedtools bamtobed -i bwa.bam | more
```

chrM 24 62 chrM_1_492_0:0:0:0:0:1:2:25:0:1:0:2_2bf/1 60
chrM 24 62 chrM_2_547_1:0:0:0:0:2:28:0:1:0:2_5bf/1 60
chrM 24 62 chrM_3_535_0:0:0:0:0:2:32:0:1:0:2_3bf/1 60
chrM 10 80 chrM_11_492_0:0:0:0:0:2:33:0:1:0:2_3bf/1 60
chrM 24 62 chrM_12_495_0:0:0:0:0:2:38:0:1:0:2_3bf/1 60
chrM 11 81 chrM_13_513_0:0:0:0:0:2:38:0:1:0:2_3bf/1 60
chrM 11 81 chrM_14_511_0:0:0:0:0:2:38:0:1:0:2_3bf/1 60
chrM 16 86 chrM_17_451_0:0:0:0:0:2:38:0:1:0:2_3bf/1 60
chrM 20 90 chrM_21_512_0:0:0:0:0:2:38:0:1:0:2_3bf/1 60
chrM 27 99 chrM_28_457_0:0:0:0:0:2:33:0:1:0:2_3bf/1 60
chrM 29 99 chrM_30_505_0:0:0:0:0:2:33:0:1:0:2_3bf/1 60
chrM 41 111 chrM_42_553_0:0:0:0:0:2:33:0:1:0:2_3bf/1 60
chrM 42 112 chrM_43_527_0:0:0:0:0:2:33:0:1:0:2_3bf/1 60
chrM 59 125 chrM_56_517_0:0:0:0:0:2:33:0:1:0:2_3bf/1 60
chrM 75 145 chrM_76_554_0:0:0:0:0:2:33:0:1:0:2_3bf/1 60
chrM 83 153 chrM_84_552_0:0:0:0:0:2:33:0:1:0:2_3bf/1 60
chrM 85 155 chrM_86_572_0:0:0:0:0:2:33:0:1:0:2_3bf/1 60
chrM 124 194 chrM_125_454_0:0:0:0:0:2:33:0:1:0:2_3bf/1 60

Replace column 9 with a dot
BedTools operators

- slop (extend)
- flank
- merge
- subtract
- complement

Essential feature: Strand Awareness

- Some tools take a -l (left), -r (right) parameter that will have a different effect if the “stranded” mode is turned on

1. **default mode**: left, right are interpreted on the default coordinate system (screen)

2. **stranded mode**: left, right are interpreted in the transcriptional direction 5’to 3’

Strategy: generate a simple file then study what happens

- Some tools require a genome file, tab delimited list of chromosome sizes

```
A simple file

Create this by hand or from the SAM headers

It is very important to understand what happens here. It can be occasionally counterintuitive
```
BedTools is format aware for input

```bash
ialbert@porthos ~/work/lec19
$ cat demo.gff
chrI : one 101 200 0 + . . .
chrI : two 1001 2000 0 + . .

ialbert@porthos ~/work/lec19
$ ~/bin/bedtools slop -l 10 -r 0 -s -i demo.gff -g genome.txt
chrI : one 91 200 0 + . .
chrI : two 1001 2010 0 + .

ialbert@porthos ~/work/lec19
$ ~/bin/bedtools slop -l 10 -r 0 -s -i demo.gff -g genome.txt
chrI : one 91 200 0 + .
chrI : two 1001 2010 0 + .
```

But some tools may produce output that is in different format!

This changed the output format!

```bash
ialbert@porthos ~/work/lec19
$ cat demo.gff
chrI : one 101 200 0 + . . .
chrI : two 1001 2000 0 + . .

ialbert@porthos ~/work/lec19
$ ~/bin/bedtools compliment -i demo.gff -g genome.txt
chrI 0 100
chrI 200 1000
chrI 2000 230218
chrII 0 813184

ialbert@porthos ~/work/lec19
$ ~/bin/bedtools slop -l 10 -r 0 -s -i demo.gff -g genome.txt
chrI : one 91 200 0 + . .
chrI : two 1001 2010 0 + .
```

Note that the output is in BED format!

Slop vs Flank

```bash
ialbert@porthos ~/work/lec19
$ cat demo.gff
chrI : one 101 200 0 + . . .
chrI : two 1001 2000 0 + . .

ialbert@porthos ~/work/lec19
$ ~/bin/bedtools slop -l 10 -r 0 -s -i demo.gff -g genome.txt
chrI : one 91 200 0 + . .
chrI : two 1001 2010 0 + . .

ialbert@porthos ~/work/lec19
$ ~/bin/bedtools slop -l 10 -r 0 -s -i demo.gff -g genome.txt
chrI : one 91 100 0 + . .
chrI : two 991 1000 0 + . .
```

The best is to draw the intervals and track what each tool does

Homework 19

Prepare the data: From features.gff file keep only genes that lie on chrI and chrII. Using this file:

1. Produce a file that contains the regions that are not covered by genes (complement)
2. Produce a file that contains only the 100bp long regions that are upstream of each gene (flank).