Week 11, Lecture 22

István Albert

Biochemistry and Molecular Biology and Bioinformatics Consulting Center

Penn State
Interval related tasks

An intervals are not one-dimensional points! – make sure to specify more precisely

- For each feature find the intervals from another dataset that are close/overlapping with it
- For each interval on one strand find the closest on the other strand

This is may not be sufficiently well defined.
• What are the anchor points (the locations that represent the intervals)

• Which direction does the comparison proceed – upstream, downstream?

• What gets reported?

Often we need to create another transformed interval data that conforms to what we actually need
Two features are said to overlap or intersect if they share at least one base in common.

Feature A
Feature B
Feature C

genome
Computing Interval Overlaps

- Unexpectedly complex task as it needs to account for various types of positioning:
  - full containment of either interval
  - partial overlaps

Neat and useful formulas (X,Y is the query interval):

- \textit{midpoint} = (\textit{start} + \textit{end}) // 2 \quad \text{(with integer division)}
- overlap condition: (\textit{start} < Y) \text{ and } (\textit{end} > X)
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

- **Programming tip**: for intervals that are not radically different in size a sort by start coordinate followed by a binary search will be efficient
BedTools

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

• Download and install bedtools

http://code.google.com/p/bedtools/

Quinlan AR and Hall IM, 
*BEDTools: a flexible suite of utilities for comparing genomic features.* 
*Bioinformatics.* 26, 6, (2010)
BedTools concepts

• There are many (25 and growing) tools/actions with different names

• Most tools write to the standard output

• The – (minus) character specifies the standard input

• Can be chained with 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?

Technically this is obsolete – instead of standalone programs we have commands
But the content in this old manual is still valid and often superior to the official documentation
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

  (make file –A the reads file and file –B the feature file)
Bedtools concepts

- 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
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 forward strand’s coordinate system

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

```
ialbert@porthos ~/work/lec22
$ cat demo.bed
chrI  100  200   one  0   +
chrI  300  400   two  0   -

ialbert@porthos ~/work/lec22
$ cat genome.txt
chrI   3000
chrII  813184

ialbert@porthos ~/work/lec22
$ ~/bin/bedtools slop -i demo.bed -g genome.txt -b 25
chrI    75  225   one  0   +
chrI    275  425   two  0   -

IALBERT@PORTHOS ~/WORK/LEC22
$```

A simple file

For this example we claim the chromosome is short
Stranded mode

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

```
ialbert@porthos ~/work/lec22
$ cat demo.gff
chrI . one 101 200 0 + : :
chrI . two 301 400 0 - : :

ialbert@porthos ~/work/lec22
$ ~/bin/bedtools slop -i demo.gff -g genome.txt -l 10 -r 0 -s
chrI . one 91 200 0 + : :
chrI . two 301 410 0 - : :

ialbert@porthos ~/work/lec22
$ 
```

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

Note that the output is in BED format! Moreover it is a 3 column BED format!
Slop vs Flank

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

Prepare toy examples and explore what the tool does.

Pay close attention to the directionality

Think in terms of “interval operations” as they were “mathematical operations”
Homework 22

Filter the yeast feature file (that we have been using since lecture 2) to keep only genes that lie on chrI and chrII. Then using this file:

1. Create a new interval file that contains only the genomic regions that are NOT covered by genes (complement)

2. Create an interval file that contains only the 100bp long regions that are upstream of each gene (flank).

3. Create a fasta file that contains the sequences for the 100bp flanking regions that you extracted in step 2 (getfasta).