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.

Important details

- 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

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 target interval, start, end refer to the query):

- midpoint = (start + end) // 2 (with integer division)
- overlap condition: (start < Y ) and (end > X)
Overlap/intersect

- 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 target interval, start, end refer to the query):
  - midpoint = (start + end) // 2 (with integer division)
  - overlap condition: (start < Y) and (end > X)

Interval representation

- binning → redundantly storing data at different zoom levels - originally implemented in UCSC genome browser (also used in BAM and BedTools)
- A different option → 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://bedtools.readthedocs.org/en/latest/

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

Basic concepts

- For any operation that requires **two files** the tools will require a 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
  
  (for example the A file contains the reads –B file contains the features)

Excellent documentation

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
A few 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’

Important details

- 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

Interval intersection (find overlaps)

- The most important functionality of the toolset
- Other functionality of bedtools could probably be implemented by your programs
- Efficiently intersecting intervals is an algorithmically more complex problem
Basic concepts

• For any operation that requires **two files** the tools will require a 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

(For example the \(A\) file contains the reads – \(B\) file contains the features)

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 intersect

• Different flags can produce richer outputs

• There are variants such as closest/window that are similar in functionality to intersect

• Sometime the solution to getting what you want is to create intervals of length 1 around the feature of interest

Next: Bedtools Tutorial by Aaron Quinlan

Material taught at Cold Spring Harbor summer workshops

http://quinlanlab.org/tutorials/cshl2014/bedtools.html
Regions not covered by intervals

Merging overlapping intervals

Genome wide coverage

Homework 26

Create an ebola feature file that has only the features annotated as genes. 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 250bp long regions that are upstream of each gene (flank). Call these promoter regions.

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

In your homework show the commands and a screenshot of IGV that shows the intervals.