A few BedTools operators

- extend (slop)
- flank
- merge
- subtract
- complement
• 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 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**
Or any other lecture where we aligned reads against a genome (15 to 23).

Question: **How many reads cover a certain region?**

(how many reads overlap with a certain interval)
bedtools intersect bam file with region

```
$ cat region.bed
KM034562 1000 2000
$
$ bedtools intersect -a bam/SRR1553593.bam -b region.bed > output.bam
$
$ samtools index output.bam
$`

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
Additional parameters change behavior

Require a minimal fraction of overlap.

Recall that the default is to report overlaps between features in A and B so long as at least one basepair of overlap exists. However, the \texttt{-r} option allows you to specify what fraction of each feature in A should be overlapped by a feature in B before it is reported.

Let’s be more strict and require 50\% of overlap.

\begin{verbatim}
bedtools intersect -a cpg.bed -b exons.bed \
    -wo -f 0.50 \
    | head
\end{verbatim}

\begin{verbatim}
chr1 135124 135563 Cpg:30 chr1 134772 139696 NR_039983_exon_0_0_chr1_134773_r 0 439
chr1 327790 328229 Cpg:29 chr1 324438 328581 NR_028322_exon_2_0_chr1_324439_f 0 439
chr1 327790 328229 Cpg:29 chr1 324438 328581 NR_028325_exon_2_0_chr1_324439_f 0 439
chr1 327790 328229 Cpg:29 chr1 327035 328581 NR_028327_exon_3_0_chr1_327036_f 0 439
chr1 788663 789211 Cpg:28 chr1 788770 794826 NR_047519_exon_5_0_chr1_788771_f 0 348
chr1 788663 789211 Cpg:28 chr1 788770 794826 NR_047521_exon_4_0_chr1_788771_f 0 348
chr1 788663 789211 Cpg:28 chr1 788770 794826 NR_047523_exon_3_0_chr1_788771_f 0 348
chr1 788663 789211 Cpg:28 chr1 788770 794826 NR_047524_exon_3_0_chr1_788771_f 0 348
chr1 788663 789211 Cpg:28 chr1 788770 794826 NR_047525_exon_4_0_chr1_788771_f 0 348
chr1 788663 789211 Cpg:28 chr1 788858 794826 NR_047520_exon_6_0_chr1_788859_f 0 348
\end{verbatim}

Faster analysis via sorted data.

So far the examples presented have used the traditional algorithm in bedtools for finding intersections. It turns out, however, that bedtools is much faster when using presorted data.

For example, compare the difference in speed between the two approaches when finding intersections between \texttt{exons.bed} and \texttt{hesc.chromHMM.bed}:
Regions not covered by intervals

bedtools “complement”

We often want to know which intervals of the genome are NOT “covered” by intervals in a given feature file. For example, if you have a set of ChIP-seq peaks, you may also want to know which regions of the genome are not bound by the factor you assayed. The `complement` addresses this task.

Input (1)

As an example, let’s find all of the non-exonic (i.e., intronic or intergenic) regions of the genome. Note, to do this you need a “genome” file, which tells `bedtools` the length of each chromosome in your file. Consider why the tool would need this information...

```
betools complement -i exons.bed -g genome.txt
> non-exonic.bed
head non-exonic.bed
chr1 0 11873
chr1 12227 12612
chr1 12721 13228
chr1 14829 14969
chr1 15831 15795
chr1 15947 16686
chr1 16765 16857
chr1 17055 17232
chr1 17368 17685
chr1 17742 17914
```
Merging overlapping intervals

**bedtools “merge”**

Many datasets of genomic features have many individual features that overlap one another (e.g. alignments from a ChIP seq experiment). It is often useful to combine the overlapping into a single, contiguous interval. The bedtools `merge` command will do this for you.

**Input (I)**

![Input Intervals](image)

**merge I**

![Merged Intervals](image)

**merge I (-d 10)**

![Merged Intervals with -d 10](image)

**merge I (-n)**

![Merged Intervals with -n](image)
Genome wide coverage

bedtools "genomcov"

For many analyses, one wants to measure the genome wide coverage of a feature file. For example, we often want to know what fraction of the genome is covered by 1 feature, 2 features, 3 features, etc. This is frequently crucial when assessing the "uniformity" of coverage from whole-genome sequencing. This is done with the versatile `genomcov` tool.

As an example, let's produce a histogram of coverage of the exons throughout the genome. Like the `merge` tool, `genomcov` requires pre-sorted data. It also needs a genome file as above.
Homework 26

• Get the data for the Bedtools tutorial

http://quinlanlab.org/tutorials/cshl2014/bedtools.html

• Answer the first three questions from the Puzzle List at the bottom.