Week 4, Lecture 7

István Albert
Bioinformatics Consulting Center
Penn State
Unix command of the day

For any analysis use a small dataset initially, work out the details then run it on the large datasets.

Keep the first or last 1000 lines:

```
head -1000 data.fastq > small.fastq
tail -1000 data.fastq > small.fastq
```

Keep the structure fastq has 4 lines per entry so make the number divisible by 4 → 1000/4=250
Topic: Working with genomic Features

Requirements:

1. Get the test datasets from the website (week-4-data.zip)

2. Install BEDtools (see website for information)

Goals:

• We will learn to handle genomic intervals
What is a genomic feature?

• Feature: a genomic region (interval) associated with a certain annotation (description).

Typical attributes to describe a feature

1. chromosome
2. start
3. end
4. strand
5. name
DNA two stranded and directional → But there is only one coordinate system

Most formats use **start < end** even on the reverse strand

The **upstream region** – before the 5’ end relative to the direction of transcription
Coordinate systems

• 0 based $\rightarrow$ first 10 $\rightarrow$ 0, 1, 2, ... 9
• 1 based $\rightarrow$ first 10 $\rightarrow$ 1, 2, 3, 4, ... 10

Typically

• 0 based are non-inclusive 10:20 $\rightarrow$ [ 10, 20 )
• 1 based include both ends 10:20 $\rightarrow$ [ 10, 20 ]
Why do different indexing systems even exist?

<table>
<thead>
<tr>
<th>1 based indexing</th>
<th>0 based indexing</th>
</tr>
</thead>
<tbody>
<tr>
<td>Third element</td>
<td>Third element</td>
</tr>
<tr>
<td>First ten</td>
<td>First ten</td>
</tr>
<tr>
<td>data[1:10]</td>
<td>data[0 :10]</td>
</tr>
<tr>
<td>Second ten:</td>
<td>Second ten:</td>
</tr>
<tr>
<td>Third ten:</td>
<td>Third ten:</td>
</tr>
<tr>
<td>Size of the slice = 10 → end-start + 1</td>
<td>Size of the slice = 10 → end – start</td>
</tr>
<tr>
<td>Empty slice: data[?] – not sure</td>
<td>Empty slice data[10:10] → size=0</td>
</tr>
<tr>
<td>Get a five element long segment starting</td>
<td>Get a five element long segment starting</td>
</tr>
<tr>
<td>at 1000</td>
<td>at 1000</td>
</tr>
<tr>
<td>data[1000: 10000 + 4 ]</td>
<td>data[1000: 10000 + 5 ]</td>
</tr>
</tbody>
</table>
http://genome.ucsc.edu/FAQ/FAQformat.html
Two commonly used formats

• **BED** – UCSC genome browser → 0 based non inclusive → also used to display tracks in the genome browser (US “standard”)

• **GFF** – Sanger institute in Great Britain → 1 based inclusive indexing system (“European standard”)

• Neither of them includes column names
The first three required BED fields are:

1. **chrom** - The name of the chromosome (e.g. chr3, chrY, chr2_random) or scaffold (e.g. scaffold10671).
2. **chromStart** - The starting position of the feature in the chromosome or scaffold. The first base in a chromosome is numbered 0.
3. **chromEnd** - The ending position of the feature in the chromosome or scaffold. The chromEnd base is not included in the display of the feature. For example, the first 100 bases of a chromosome are defined as chromStart=0, chromEnd=100, and span the bases numbered 0-99.

The 9 additional optional BED fields are:

4. **name** - Defines the name of the BED line. This label is displayed to the left of the BED line in the Genome Browser window when the track is open to full display mode or directly to the left of the item in pack mode.
5. **score** - A score between 0 and 1000. If the track line useScore attribute is set to 1 for this annotation data set, the score value will determine the level of gray in which this feature is displayed (higher numbers = darker gray). This table shows the Genome Browser’s translation of BED score values into shades of gray:

<table>
<thead>
<tr>
<th>shade</th>
<th>score in range</th>
</tr>
</thead>
<tbody>
<tr>
<td>white</td>
<td>0 - 166</td>
</tr>
<tr>
<td>light gray</td>
<td>167-277</td>
</tr>
<tr>
<td>gray</td>
<td>278-388</td>
</tr>
<tr>
<td>dark gray</td>
<td>389-499</td>
</tr>
<tr>
<td>very dark gray</td>
<td>500-611</td>
</tr>
<tr>
<td>black</td>
<td>612-722</td>
</tr>
<tr>
<td>darkest gray</td>
<td>723-833</td>
</tr>
<tr>
<td>black</td>
<td>834-944</td>
</tr>
<tr>
<td>black</td>
<td>≥ 945</td>
</tr>
</tbody>
</table>

6. **strand** - Defines the strand - either '+' or '-'.
7. **thickStart** - The starting position at which the feature is drawn thickly (for example, the start codon in gene displays).
8. **thickEnd** - The ending position at which the feature is drawn thickly (for example, the stop codon in gene displays).
9. **itemRgb** - An RGB value of the form R,G,B (e.g. 255,0,0). If the track line itemRgb attribute is set to "On", this RGB value will determine the display color of the data contained in this BED line. NOTE: It is recommended that a simple color scheme (eight colors or less) be used with this attribute to avoid overwhelming the color resources of the Genome Browser and your Internet browser.
10. **blockCount** - The number of blocks (exons) in the BED line.
11. **blockSizes** - A comma-separated list of the block sizes. The number of items in this list should correspond to blockCount.
12. **blockStarts** - A comma-separated list of block starts. All of the blockStart positions should be calculated relative to chromStart. The number of items in this list should correspond to blockCount.

**Example:**
Here’s an example of an annotation track that uses a complete BED definition:

```
track name=pairedReads description="Clone Paired Reads" useScore=1
chr22 1000 5000 cloneA 960 + 1000 5000 0 2 567,488, 0,3512
chr22 2000 6000 cloneB 900 - 2000 6000 0 2 433,399, 0,3601
```
GFF format

Search for GFF3 → http://www.sequenceontology.org/gff3.shtml

Tab separated with 9 columns. Missing attributes may be replaced with a dot → .

1. **Seqid** (usually chromosome)
2. **Source** (where is the data coming from)
3. **Type** (usually a term from the sequence ontology)
4. **Start** (interval start relative to the seqid)
5. **End** (interval end relative to the seqid)
6. **Score** (the score of the feature, a floating point number)
7. **Strand** (+/-/.)
8. **Phase** (used to indicate reading frame for coding sequences)
9. **Attributes** (semicolon separated attributes → Name=ABC;ID=1)
We may have data in different coordinate systems!

Being “one off” is one of the most common errors in bioinformatics.

Conversion from GFF to BED

(start, end) \rightarrow (start - 1, end)

Conversion from BED to GFF

(start, end) \rightarrow (start + 1, end)
Typical interval related tasks

Finding intervals relative to one another, for example:

• for each interval on one strand find the closest on the other strand

An interval is not a point - these type of problems need to be better specified
• What are the anchor points (the locations that represent the intervals)

• Which direction does the comparison proceed – upstream, downstream?
Two features are said to overlap or intersect if they share at least one base in common.
Further complications ...

• An interval file may or may not be sorted

• Usually that means sorted by the start coordinates

• Some operations require that the files be sorted!
Where to get genomic features

• UCSC: http://genome.ucsc.edu/

• Ensembl/Biomart: http://www.ensembl.org/

Numerous custom databases tuned for a certain organism or disease.
Exporting data from the UCSC genome browser
Exporting data from BioMart

<table>
<thead>
<tr>
<th>Chromosome Name</th>
<th>Gene Start (bp)</th>
<th>Gene End (bp)</th>
<th>Strand</th>
<th>Associated Gene Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>17</td>
<td>74720132</td>
<td>7473413</td>
<td>1</td>
<td>AC005837</td>
</tr>
<tr>
<td>17</td>
<td>75446413</td>
<td>75468652</td>
<td>-1</td>
<td>AC111170</td>
</tr>
<tr>
<td>17</td>
<td>75543823</td>
<td>75560925</td>
<td>1</td>
<td>AC021683</td>
</tr>
<tr>
<td>17</td>
<td>75718934</td>
<td>75748643</td>
<td>-1</td>
<td>AC104985</td>
</tr>
<tr>
<td>17</td>
<td>77889984</td>
<td>77900024</td>
<td>-1</td>
<td>AC100791</td>
</tr>
<tr>
<td>17</td>
<td>78313698</td>
<td>79209690</td>
<td>-1</td>
<td>AC027601</td>
</tr>
<tr>
<td>17</td>
<td>78775440</td>
<td>78794620</td>
<td>1</td>
<td>AC019245</td>
</tr>
<tr>
<td>17</td>
<td>79084137</td>
<td>79069203</td>
<td>1</td>
<td>AC139530</td>
</tr>
<tr>
<td>17</td>
<td>79885705</td>
<td>79888623</td>
<td>1</td>
<td>AC145207</td>
</tr>
<tr>
<td>17</td>
<td>86172103</td>
<td>86175526</td>
<td>-1</td>
<td>AC132872</td>
</tr>
</tbody>
</table>
Sometime we cannot get the data in the exact form we needed. This is not a BED file although it is similar to it – you may not be able to transform it yourself. It is a lot easier to find help to transform this file into a BED format than finding it in BED format in the first place.

<table>
<thead>
<tr>
<th>Chromosome</th>
<th>Name</th>
<th>Gene Start (bp)</th>
<th>Gene End (bp)</th>
<th>Strand</th>
<th>Asso</th>
</tr>
</thead>
<tbody>
<tr>
<td>17</td>
<td></td>
<td>74730199</td>
<td>74733413</td>
<td>1</td>
<td>AC005837.1</td>
</tr>
<tr>
<td>17</td>
<td></td>
<td>75464643</td>
<td>75468852</td>
<td>-1</td>
<td>AC111170.1</td>
</tr>
<tr>
<td>17</td>
<td></td>
<td>75543023</td>
<td>75559325</td>
<td>1</td>
<td>AC021683.1</td>
</tr>
<tr>
<td>17</td>
<td></td>
<td>75718954</td>
<td>75724641</td>
<td>-1</td>
<td>AC104981.1</td>
</tr>
<tr>
<td>17</td>
<td></td>
<td>77889984</td>
<td>77900524</td>
<td>-1</td>
<td>AC100791.3</td>
</tr>
<tr>
<td>17</td>
<td></td>
<td>78313698</td>
<td>79329659</td>
<td>-1</td>
<td>AC027601.1</td>
</tr>
<tr>
<td>17</td>
<td></td>
<td>78775440</td>
<td>78779420</td>
<td>-1</td>
<td>AC016245.1</td>
</tr>
<tr>
<td>17</td>
<td></td>
<td>79604197</td>
<td>79606203</td>
<td>1</td>
<td>AC139530.1</td>
</tr>
<tr>
<td>17</td>
<td></td>
<td>79885705</td>
<td>79888628</td>
<td>1</td>
<td>AC145207.1</td>
</tr>
<tr>
<td>17</td>
<td></td>
<td>80172103</td>
<td>80175228</td>
<td>-1</td>
<td>AC132872.4</td>
</tr>
<tr>
<td>17</td>
<td></td>
<td>80200543</td>
<td>80203317</td>
<td>1</td>
<td>AC132872.3</td>
</tr>
<tr>
<td>17</td>
<td></td>
<td>80247922</td>
<td>80250690</td>
<td>-1</td>
<td>AC132872.2</td>
</tr>
<tr>
<td>2</td>
<td></td>
<td>243173849</td>
<td>243175523</td>
<td>-1</td>
<td>AC215220.1</td>
</tr>
</tbody>
</table>
A possible solution

```bash
1. cat biomart-export.txt | awk -f prog.txt > biomart-export.bed
```

where someone else gives you the content of `prog.txt`
This is what a prog.txt could look like

```plaintext
1 { 
2     if ( $4=="1") $4 = "+"; else $4 = "-";
3     printf("%s\t%s\t%s\t%s\t%0\t%s\n", $1, $2, $3, $5, $4); 
4 } 
```

You don’t need to know how to make this. This is (should be) very easy for someone that hired to do programming.
The easiest is to filter your own data

- `sc-features.gff` → **ALL** the genomic features for the *Saccharomyces Cerevisiae*
Select all ARS elements

# ARS = cis-acting sequences required for the initiation of chromosomal DNA replication
#
# count how many ARS elements we have
#
grep ARS sc-features.gff | wc -l

# now select all the ARS elements and put them into a file
grep ARS sc-features.gff > ars.gff
VCF – Variant Call Format

- http://www.1000genomes.org/node/101

encoding structural variations
Visualize BED files in the UCSC genome browser

- Add a browser line to the top of the bed file
Upload into the UCSC genome browser

Note that there are many other visualizers that you can use. We will cover other options later.
The UCSC genome browser was first to offer custom visualizations.
Exercises

• Find and download genomic data from UCSC and Ensembl

• Redo the examples for filtering from the features in the yeast genome
Week 4, Lecture 8

István Albert
Bioinformatics Consulting Center
Penn State
Workflow so far

• The instrument gave us data.fastq (week 1)

• We mapped it with bowtie against the yeast genome (week 3) → this gave us a SAM file

• We transform the SAM file to a BAM file (we did not cover this yet, it is command line tool)

• Now we have BAM file called mapped-reads.bam

In the archive there is a file called all-commands.txt that contains all the commands needed to generate today’s presentation (some of them we have not covered yet)
BedTools

• High performance software package that deals with interval data: BED, GFF, BAM and VCF

• SAM is the Sequence Alignment Map format (see lecture 6)

• BAM is a binary version of SAM
BedTools concepts

• There are 25 tools with different names

• Most tools write to the standard output

• Can be chained with *pipes* like all other tools

• Most tools write their help when invoked

• Some tools need the –h flag to print their help message

• Keep the manual open
Tips for running the bedtools programs

• There are multiple programs

• sc.genome is the file with the genome size (check its content)

• Keep the manual open

• Pick various output formats
Transform from BAM to bedfile

```
jalbert@borg ~/course/week4/demo
$ /home/code/bedtools/bamToBed.exe -h

Program: bamToBed (v2.11.2)
Author: Aaron Quinlan (aaronquinlan@gmail.com)
Summary: Converts BAM alignments to BED6 or BEDPE format.
Usage: bamToBed [OPTIONS] -i <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.cse.ucsc.edu/FAQ/FAQformat#format1
  -split Report "split" BAM alignments as separate BED entries.
  -ed Use BAM edit distance (NM tag) for BED score.
    - Default for BED is to use mapping quality.
    - Default for BEDPE is to use the minimum of the two mapping qualities for the pair.

/path/to/bedtools/bamToBed.exe -i reads.bam > reads.bed
```
Genome Coverage

Program: genomeCoverageBed (v2.11.2)
Authors: Aaron Quinlan (aaronquinlan@gmail.com)
         Assaf Gordon, CSHL

Summary: Compute the coverage of a feature file among a genome.
Usage:   genomeCoverageBed [OPTIONS] -i <bed/gff/vcf> -g <genome>

Options:

-ibam    The input file is in BAM format.
Note: BAM _must_ be sorted by position
-d       Report the depth at each genome position.
Default behavior is to report a histogram.
-bg      Report depth in BedGraph format. For details, see:
genome.ucsc.edu/goldenPath/help/bedgraph.html
-bga     Report depth in BedGraph format, as above (-bg).
However with this option, regions with zero

/path/to/bedtools/genomeCoverageBed.exe
### Default output

```bash
$ /home/code/bedtools/genomeCoverageBed.exe -i reads.bed -g sc.genome | more
```

| 2micron 0 | 6318 | 415 | 0.0656853 |
| 2micron 1 | 6318 | 457 | 0.072333 |
| 2micron 2 | 6318 | 290 | 0.0459006 |
| 2micron 3 | 6318 | 159 | 0.0251662 |
| 2micron 4 | 6318 | 79  | 0.012504  |
| 2micron 5 | 6318 | 151 | 0.0239    |
| 2micron 6 | 6318 | 69  | 0.0109212 |
| 2micron 7 | 6318 | 122 | 0.0193099 |
| 2micron 8 | 6318 | 117 | 0.0185185 |
| 2micron 9 | 6318 | 93  | 0.0147198 |
| 2micron 10| 6318 | 98  | 0.0155112 |
| 2micron 11| 6318 | 119 | 0.0188351 |
| 2micron 12| 6318 | 143 | 0.0226337 |
| 2micron 13| 6318 | 184 | 0.0291231 |
| 2micron 14| 6318 | 90  | 0.014245  |
| 2micron 15| 6318 | 90  | 0.014245  |
| 2micron 16| 6318 | 50  | 0.0079139 |
| 2micron 17| 6318 | 63  | 0.00997151|
| 2micron 18| 6318 | 125 | 0.0197847 |
| 2micron 19| 6318 | 96  | 0.0151947 |
| 2micron 20| 6318 | 76  | 0.0120291 |
| 2micron 21| 6318 | 114 | 0.0180437 |
Change the output format

```
ialbert@borg ~/course/week4/work
$ /home/code/bedtools.genomeCoverageBed.exe -d -i reads.bed -g sc.genome | more
2micron 1 0
2micron 2 0
2micron 3 0
2micron 4 0
2micron 5 0
2micron 6 0
2micron 7 0
2micron 8 0
2micron 9 0
2micron 10 0
2micron 11 0
2micron 12 0
2micron 13 0
2micron 14 0
2micron 15 0
2micron 16 0
2micron 17 0
2micron 18 0
2micron 19 0
2micron 20 0
2micron 21 0
```

```
genomeCoverageBed.exe -d -i reads.bed -g sc.genome | more
```
Commands involving two interval files

• There is a file A and a file B

• Each line in file A is checked against a **interval tree** built from file B

• Interval trees are resource intensive, put the smaller of the two files to be file B
Intersect/Overlap

First reduce your features to genes only

```
grep genes sc-features.gff > genes.gff
```

/path/to/bedtools/intersectBed.exe
### Default output

```bash
intersectBed.exe -a reads.bed -b genes.gff | more
```

<table>
<thead>
<tr>
<th>chr</th>
<th>start</th>
<th>end</th>
<th>name</th>
<th>score</th>
<th>strand</th>
</tr>
</thead>
<tbody>
<tr>
<td>chrI</td>
<td>48563</td>
<td>48565</td>
<td>HWI-ST407_110218_0088_B81H3VABXX:1:1:9445:25761#0/1</td>
<td>255</td>
<td>-</td>
</tr>
<tr>
<td>chrI</td>
<td>48563</td>
<td>48602</td>
<td>HWI-ST407_110218_0088_B81H3VABXX:1:1:19064:12475#0/1</td>
<td>255</td>
<td>+</td>
</tr>
<tr>
<td>chrI</td>
<td>48611</td>
<td>48611</td>
<td>HWI-ST407_110218_0088_B81H3VABXX:1:1:10668:3353#0/1</td>
<td>255</td>
<td>-</td>
</tr>
<tr>
<td>chrI</td>
<td>48611</td>
<td>48659</td>
<td>HWI-ST407_110218_0088_B81H3VABXX:1:1:2253:9868#0/1</td>
<td>255</td>
<td>-</td>
</tr>
<tr>
<td>chrI</td>
<td>48641</td>
<td>48689</td>
<td>HWI-ST407_110218_0088_B81H3VABXX:1:1:4878:3554#0/1</td>
<td>255</td>
<td>-</td>
</tr>
<tr>
<td>chrI</td>
<td>48698</td>
<td>48746</td>
<td>HWI-ST407_110218_0088_B81H3VABXX:1:1:2552:7764#0/1</td>
<td>255</td>
<td>+</td>
</tr>
<tr>
<td>chrI</td>
<td>48783</td>
<td>48831</td>
<td>HWI-ST407_110218_0088_B81H3VABXX:1:1:20531:24160#0/1</td>
<td>255</td>
<td>-</td>
</tr>
<tr>
<td>chrI</td>
<td>48788</td>
<td>48836</td>
<td>HWI-ST407_110218_0088_B81H3VABXX:1:1:13112:13178#0/1</td>
<td>255</td>
<td>-</td>
</tr>
<tr>
<td>chrI</td>
<td>48830</td>
<td>48878</td>
<td>HWI-ST407_110218_0088_B81H3VABXX:1:1:20404:13591#0/1</td>
<td>255</td>
<td>+</td>
</tr>
<tr>
<td>chrI</td>
<td>48839</td>
<td>48887</td>
<td>HWI-ST407_110218_0088_B81H3VABXX:1:1:10319:13098#0/1</td>
<td>255</td>
<td>+</td>
</tr>
<tr>
<td>chrI</td>
<td>48847</td>
<td>48895</td>
<td>HWI-ST407_110218_0088_B81H3VABXX:1:1:13635:16043#0/1</td>
<td>255</td>
<td>-</td>
</tr>
<tr>
<td>chrI</td>
<td>48849</td>
<td>48897</td>
<td>HWI-ST407_110218_0088_B81H3VABXX:1:1:11378:3298#0/1</td>
<td>255</td>
<td>-</td>
</tr>
<tr>
<td>chrI</td>
<td>48900</td>
<td>48948</td>
<td>HWI-ST407_110218_0088_B81H3VABXX:1:1:10430:18034#0/1</td>
<td>255</td>
<td>+</td>
</tr>
<tr>
<td>chrI</td>
<td>48951</td>
<td>48999</td>
<td>HWI-ST407_110218_0088_B81H3VABXX:1:1:16063:19613#0/1</td>
<td>255</td>
<td>-</td>
</tr>
<tr>
<td>chrI</td>
<td>48954</td>
<td>49002</td>
<td>HWI-ST407_110218_0088_B81H3VABXX:1:1:1714:18440#0/1</td>
<td>255</td>
<td>+</td>
</tr>
<tr>
<td>chrI</td>
<td>48991</td>
<td>49039</td>
<td>HWI-ST407_110218_0088_B81H3VABXX:1:1:18098:12007#0/1</td>
<td>255</td>
<td>-</td>
</tr>
<tr>
<td>chrI</td>
<td>49040</td>
<td>49088</td>
<td>HWI-ST407_110218_0088_B81H3VABXX:1:1:19945:10078#0/1</td>
<td>255</td>
<td>-</td>
</tr>
<tr>
<td>chrI</td>
<td>49227</td>
<td>49275</td>
<td>HWI-ST407_110218_0088_B81H3VABXX:1:1:14456:23866#0/1</td>
<td>255</td>
<td>+</td>
</tr>
<tr>
<td>chrI</td>
<td>49229</td>
<td>49277</td>
<td>HWI-ST407_110218_0088_B81H3VABXX:1:1:13922:4890#0/1</td>
<td>255</td>
<td>+</td>
</tr>
<tr>
<td>chrI</td>
<td>49274</td>
<td>49322</td>
<td>HWI-ST407_110218_0088_B81H3VABXX:1:1:10140:5392#0/1</td>
<td>255</td>
<td>+</td>
</tr>
<tr>
<td>chrI</td>
<td>49279</td>
<td>49327</td>
<td>HWI-ST407_110218_0088_B81H3VABXX:1:1:9973:9935#0/1</td>
<td>255</td>
<td>-</td>
</tr>
</tbody>
</table>
Customize the output add –wo flag

intersectBed.exe -wo -a reads.bed -b genes.gff | more
windowBed

- Similar to `intersectBed`, `windowBed` searches for overlapping features in A and B. However, `windowBed` adds a specified number (1000, by default) of base pairs upstream and downstream of each feature in A.

- In effect, this allows features in B that are “near” features in A to be detected.
Exercise

• Use windowBed to find reads that map upstream within 1000 bases of each gene

• Investigate three tools not presented in this lecture
Behind the scenes – not always easy to see

• The file called `all-commands.txt` in the data lists all the commands that I had to perform to prepare today’s presentation.

• Shows an series of commands that could be easily rerun
  (some steps like fetching the whole genome of course are usually done only once)