



Analyzing Next Generation Sequencing Data, MSU 2014

# Genomic Intervals

**István Albert**

Bioinformatics Consulting Center

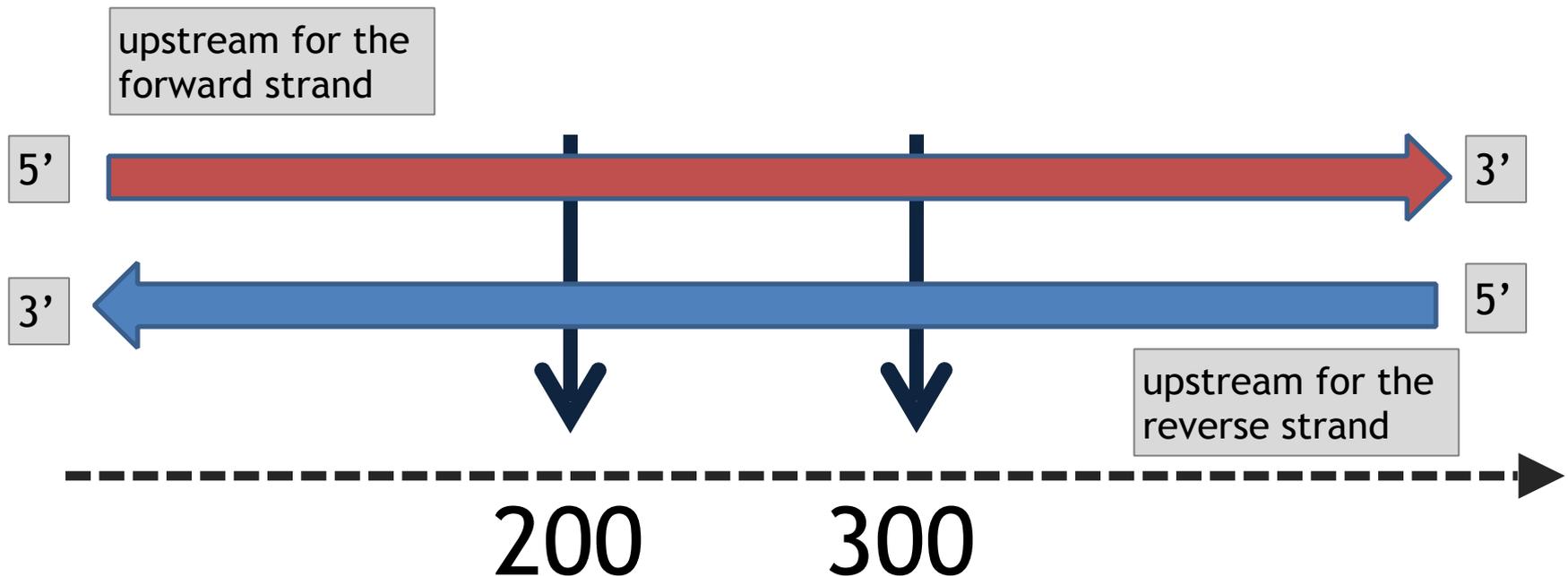
Penn State, 2014

# Genome representation concepts

- At the simplest level of abstraction the genome is represented by a one dimensional “space” (lines)
- Genome is two stranded → a line corresponds to each strand
- Each strand has a polarity → each line has a direction
- The genome has strands (lines) are paired
- The smallest unit is one base → one integer on the number line
- Annotations (features) are segments (coordinates) on each line

# Genomic coordinates - brief overview

DNA two stranded and directional  
But there is only one coordinate system



Standard formats use start < end notation even for the reverse strand

The **upstream region** - before the 5' end relative to the direction of transcription

# Coordinate systems

- 0 based  $\rightarrow 0, 1, 2, \dots 9$
- 1 based  $\rightarrow 1, 2, 3, \dots 10$

## Typically

- 0 based are non-inclusive  $10:20 \rightarrow [ 10, 20 )$
- 1 based include both ends  $10:20 \rightarrow [10, 20 ]$

# Comparing coordinate systems

	1 based indexing	0 based indexing
<i>Third element</i>	<b>3</b>	<b>2</b>
<i>First ten</i>	<b>1, 10</b>	<b>0, 10</b>
<i>Second ten</i>	<b>11, 20</b>	<b>10, 20</b>
	<b>21, 30</b>	<b>20, 30</b>
<i>One base long starting at 10</i>	<b>10, 10</b>	<b>10, 11</b>
<i>Length of interval</i>	<b><math>end - start + 1</math></b>	<b><math>end - start</math></b>
<i>Five elements starting at</i>	<b>1000, 1004</b>	<b>1000, 1005</b>
<i>Empty interval</i>	<b>?</b>	<b><math>start, start (0,0)</math></b>



Vote for what you think is better



# Fundamental interval formats

- **SAM/BAM** - Sequence Alignment Map
- **BED/GFF** → Gene Annotation representation
- **VCF/BCF** → for variant calls

# Pick one coordinate system and stick with it!

Show All

My Tags

News

Questions

Unanswered

Tutorials

Tools

Videos

Jobs

## Question: What are the most common stupid mistakes in bioinformatics?

34

While I of course never have stupid mistakes...ahem...I have many "friends" who:

1. forget to check both strands
2. generate random genomic sites without avoiding masked (NNN) gaps
3. confuse genome freezes **and even species**

but I'm favorite:

42

I truncated many fasta files this way when trying to see which headers it contained:

```
grep > some.fasta
```

I also see a lot of off-by-one errors due to switching between formats

- Bed is 0 based
- GFF/GTF are 1-based

and switching between languages:

- Python and nearly every other modern language are 0-based indexing
- R is 1-based (as is Lua)

1. Pick the standard your group is using and convert every new data to this standard!
2. If you have a choice of what to use pick the one based system! (GFF).

# What is a genomic feature?

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

Typical attributes to describe a feature

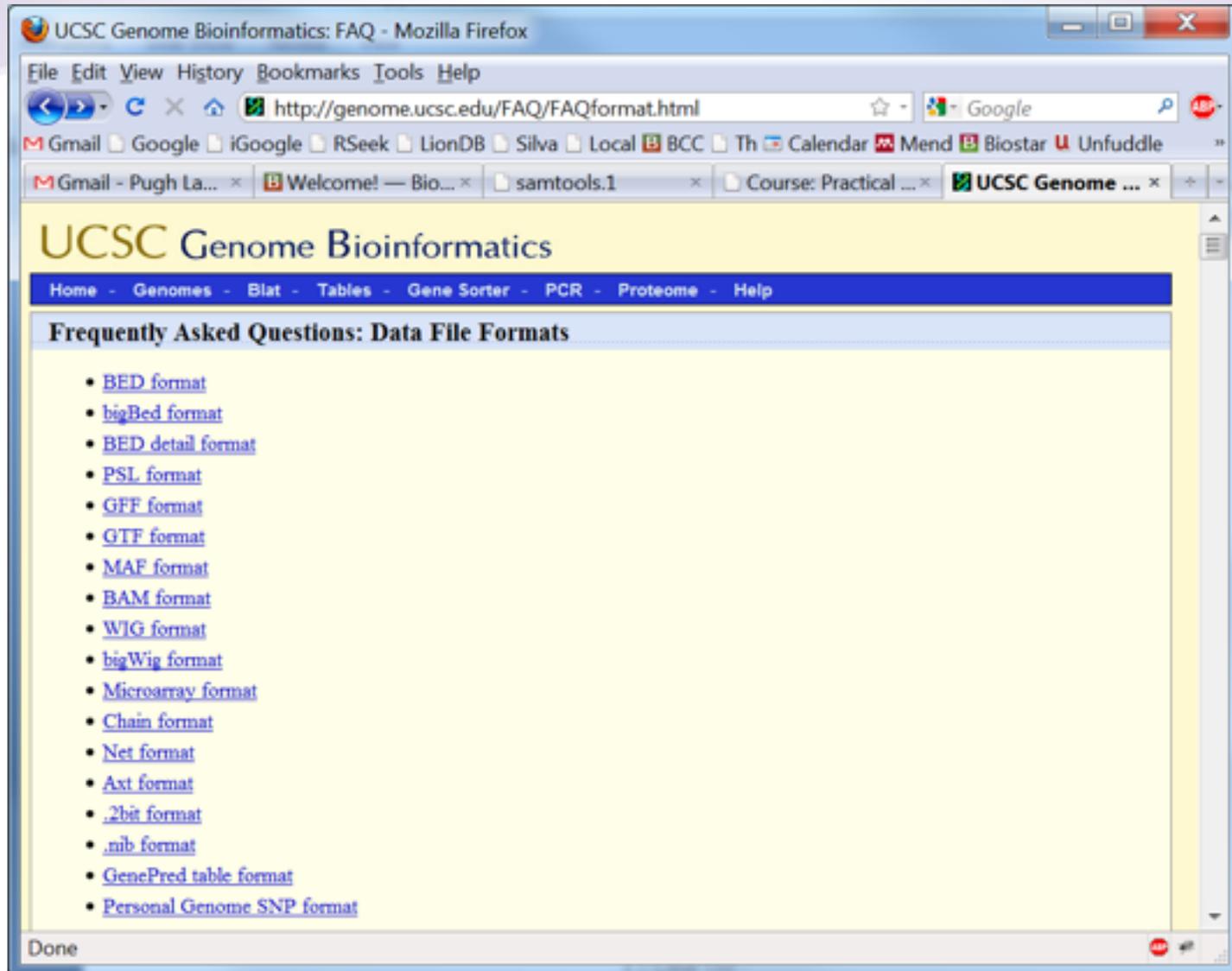
1. chromosome
2. start
3. end
4. strand
5. name

# Values over intervals

Two options:

1. A single value characterizes an entire interval  $\rightarrow$  score (value) for the interval
2. Continuous values  $\rightarrow$  different value for each base of the interval

<http://genome.ucsc.edu/FAQ/FAQformat.html>



The screenshot shows a Mozilla Firefox browser window with the title "UCSC Genome Bioinformatics: FAQ - Mozilla Firefox". The address bar displays the URL "http://genome.ucsc.edu/FAQ/FAQformat.html". The browser's menu bar includes "File", "Edit", "View", "History", "Bookmarks", "Tools", and "Help". The address bar also shows a search engine icon and the text "Google". The browser's tab bar shows several open tabs, including "Gmail", "Google", "iGoogle", "RSeek", "LionDB", "Silva", "Local", "BCC", "Th", "Calendar", "Mend", "Biostar", "Unfuddle", "Gmail - Pugh La...", "Welcome! — Bio...", "samtools.1", "Course: Practical ...", and "UCSC Genome ...".

The main content area of the browser displays the UCSC Genome Bioinformatics website. The page title is "UCSC Genome Bioinformatics". Below the title is a navigation menu with links for "Home", "Genomes", "Blat", "Tables", "Gene Sorter", "PCR", "Proteome", and "Help". The main heading of the page is "Frequently Asked Questions: Data File Formats". Below this heading is a list of links to various data file formats:

- [BED format](#)
- [bigBed format](#)
- [BED detail format](#)
- [PSL format](#)
- [GFF format](#)
- [GTF format](#)
- [MAF format](#)
- [BAM format](#)
- [WIG format](#)
- [bigWig format](#)
- [Microarray format](#)
- [Chain format](#)
- [Net format](#)
- [Axt format](#)
- [.2bit format](#)
- [.nib format](#)
- [GenePred table format](#)
- [Personal Genome SNP format](#)

The browser's status bar at the bottom left shows "Done".

# Two commonly used formats

- **BED** - UCSC genome browser → 0 based non inclusive → also used to display tracks in the genome browser (US “standard”) (variants: **bigBed**, **bedgraph**)
- **GFF** - Sanger institute in Great Britain → 1 based inclusive indexing system (“European standard”), (variants: **GTF**, **GFF 2.0**)

# BED format

## Search for BED format

Tab separated 3 required and 9 optional columns.

1. **chrom** (name of the chromosome, sequence id)
2. **chromStart** (starting position on the chromosome)
3. **chromEnd** (end position of the chromosome, **note** this base is not included)
4. **name** (feature name)
5. **score** (between 0 and 1000)
6. **strand** (+ or -)
7. **thickStart** (the starting position at which the feature is drawn thickly)
8. **thickEnd** (the ending position at which the feature is drawn thickly)
9. **itemRGB** (RGB color → 255, 0, 0 display color of the data contained)
10. **blockCount** (the number of blocks (exons) in the BED line.)
11. **blockSizes** (a comma-separated list of the block sizes)
12. **blockStarts** (a comma-separated list of the block starts)

# GFF format

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

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

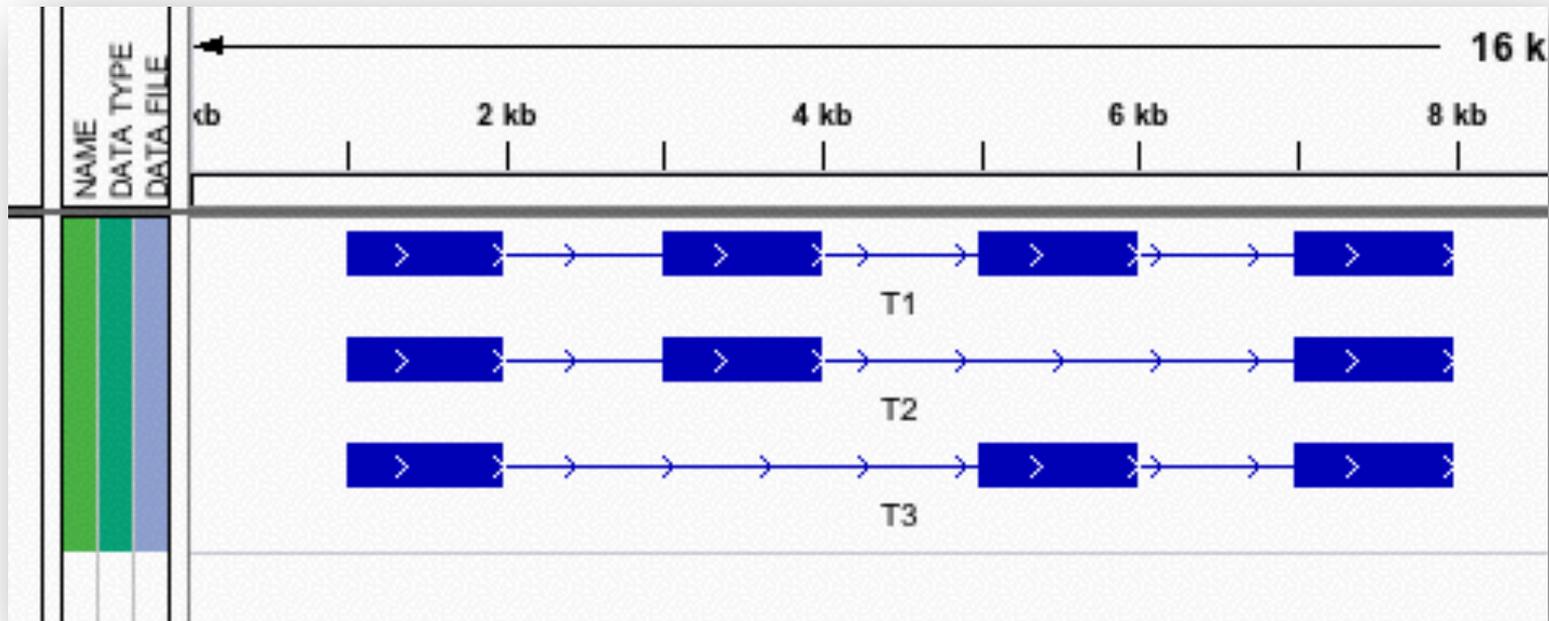
1. **Seqid** (usually chromosome)
2. **Source** (where is the data coming from)
3. **Type** (usually a term from the sequence ontology)
4. **Start** (interval start relative to the seqid)
5. **End** (interval end relative to the seqid)
6. **Score** (the score of the feature, a floating point number)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



people like to stuff a lot of information here

# Representing interval relationships

We have a gene with three splicing variants



Note: each exon is 1kb separated by multiples of 1kb

How to represent this data in a simple text file?

# Data representation

- Both BED and GFF files can represent them
- Two common versions of GFF → **GTF2** and **GFF3**  
*(note: tool documentation can often be wrong and shows a weird combination of these two formats)*
- In GFF the content of the ATTRIBUTE (9<sup>th</sup>) column specifies the relationship between features

# The GTF formats

GTF attributes:

- **gene\_id** value;  
*a globally unique identifier for the genomic source of the transcript*
- **transcript\_id** value  
*a globally unique identifier for the predicted transcript.*

**gene\_id “G1” transcript\_id “T1”**

GFF attributes:

**ID=exon1; Parent=T1**

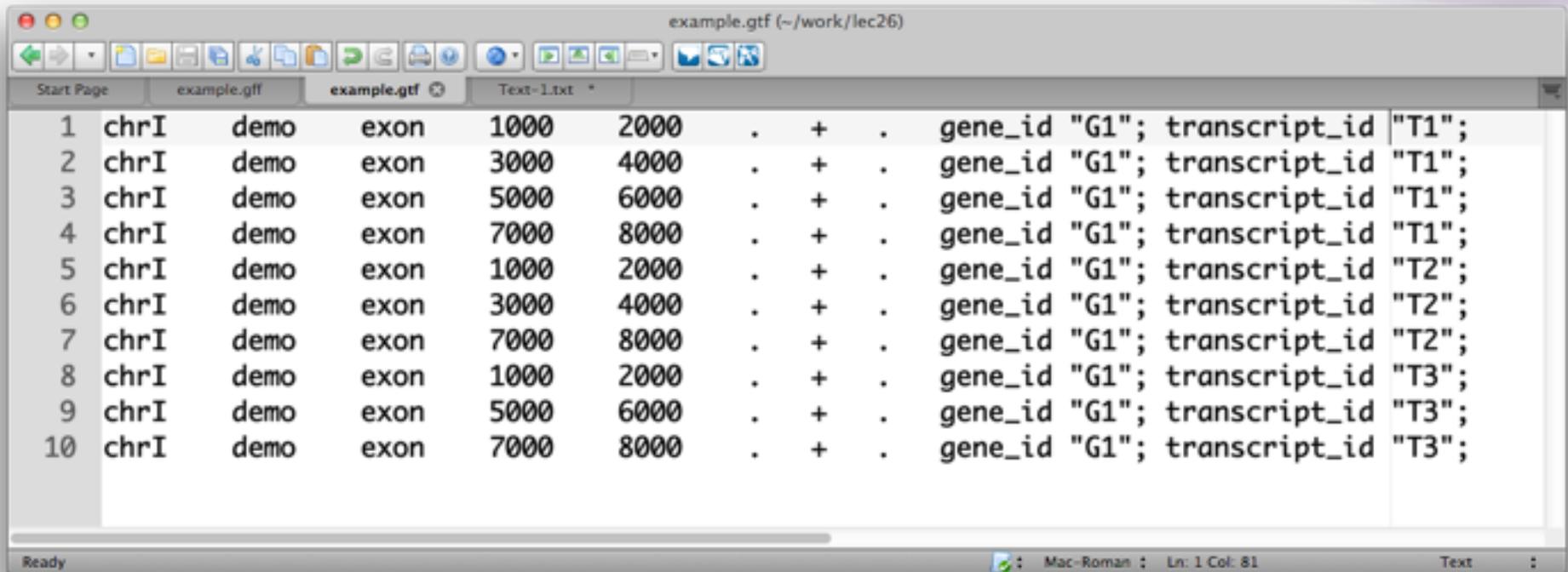
See the GFF3 site for exact specification of the these mean.  
Important: More than one parent may be listed!

# The GFF formats

GFF attributes:

**ID=exon1; Parent=T1**

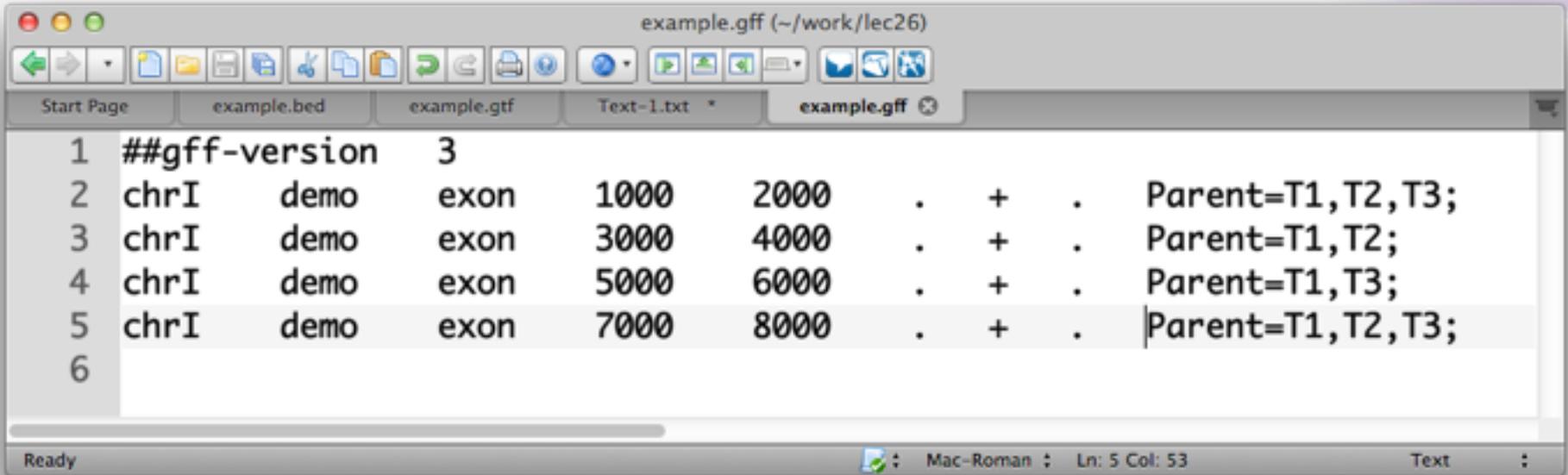
# Example interval as GTF



```
example.gtf (~/work/lec26)
Start Page example.gff example.gtf Text-1.txt
1 chrI demo exon 1000 2000 . + . gene_id "G1"; transcript_id "T1";
2 chrI demo exon 3000 4000 . + . gene_id "G1"; transcript_id "T1";
3 chrI demo exon 5000 6000 . + . gene_id "G1"; transcript_id "T1";
4 chrI demo exon 7000 8000 . + . gene_id "G1"; transcript_id "T1";
5 chrI demo exon 1000 2000 . + . gene_id "G1"; transcript_id "T2";
6 chrI demo exon 3000 4000 . + . gene_id "G1"; transcript_id "T2";
7 chrI demo exon 7000 8000 . + . gene_id "G1"; transcript_id "T2";
8 chrI demo exon 1000 2000 . + . gene_id "G1"; transcript_id "T3";
9 chrI demo exon 5000 6000 . + . gene_id "G1"; transcript_id "T3";
10 chrI demo exon 7000 8000 . + . gene_id "G1"; transcript_id "T3";
Ready Mac-Roman Ln: 1 Col: 81 Text
```

A distinct line is entered for each exon, repeated for each transcript

# Example interval as GFF 3



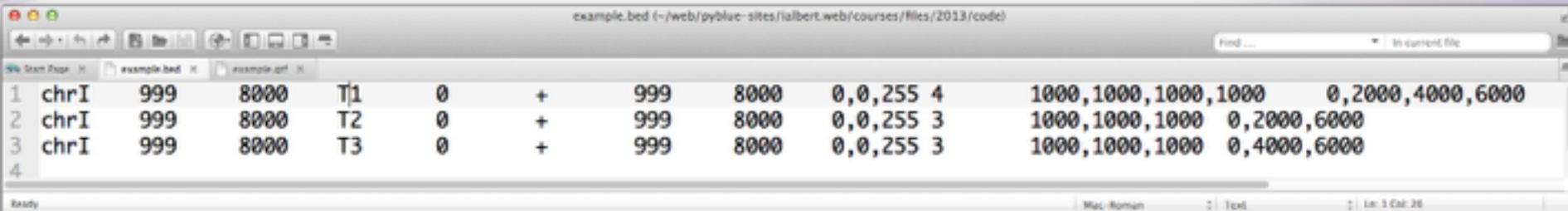
The screenshot shows a text editor window titled "example.gff (~/.work/lec26)". The window contains the following GFF 3 format text:

```
1 ##gff-version 3
2 chrI demo exon 1000 2000 . + . Parent=T1,T2,T3;
3 chrI demo exon 3000 4000 . + . Parent=T1,T2;
4 chrI demo exon 5000 6000 . + . Parent=T1,T3;
5 chrI demo exon 7000 8000 . + . Parent=T1,T2,T3;
6
```

The status bar at the bottom indicates "Ready", "Mac-Roman", "Ln: 5 Col: 53", and "Text".

The same exon may be part of different transcripts (parents)

# Example interval in BED

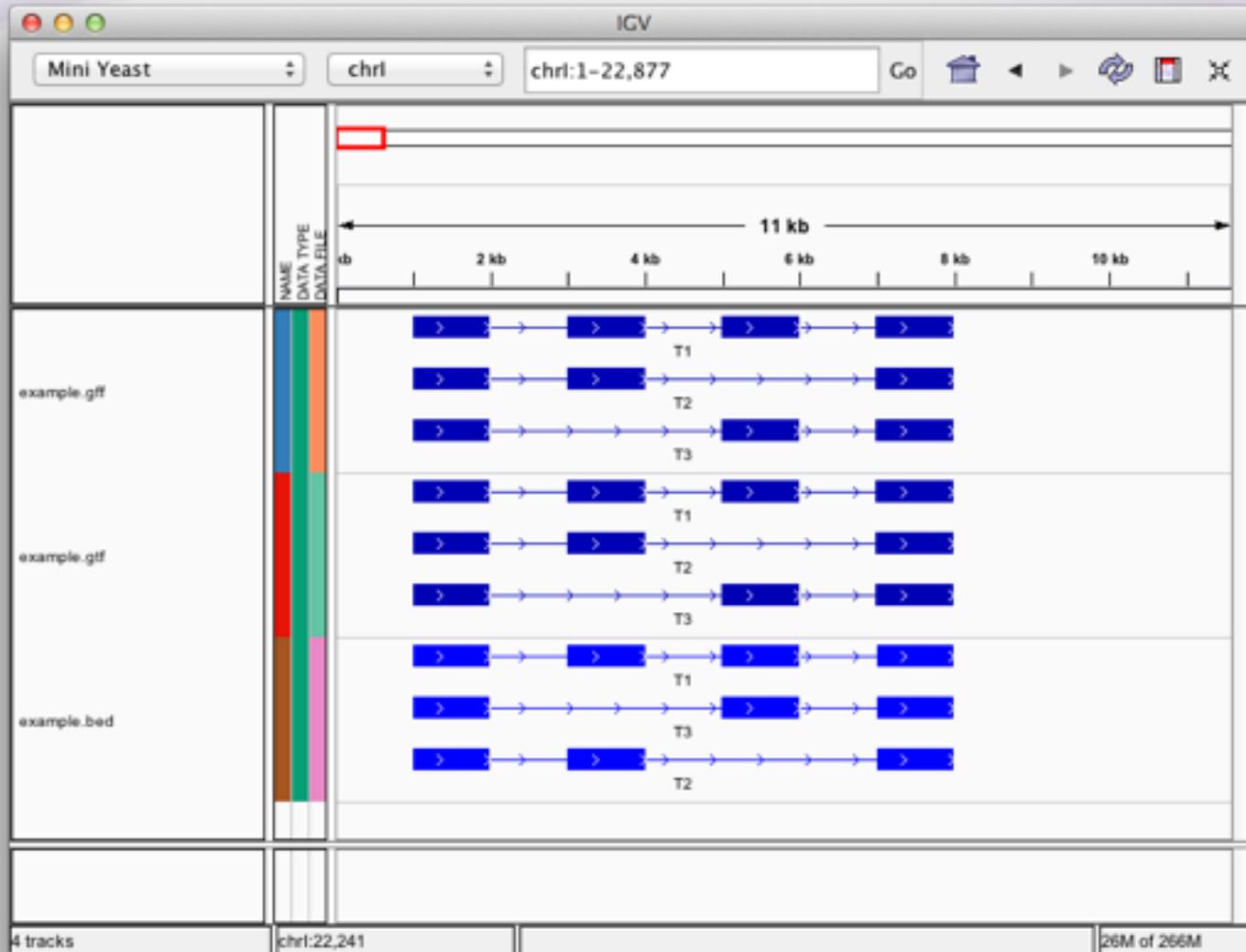


```
example.bed (-/web/pyblue-sites/ialbert.web/courses/files/2013/code)
Find ... In current file
1 chrI 999 8000 T1 0 + 999 8000 0,0,255 4 1000,1000,1000,1000 0,2000,4000,6000
2 chrI 999 8000 T2 0 + 999 8000 0,0,255 3 1000,1000,1000 0,2000,6000
3 chrI 999 8000 T3 0 + 999 8000 0,0,255 3 1000,1000,1000 0,4000,6000
4
```

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*.

From the BED format specification

# Visualizing in IGV



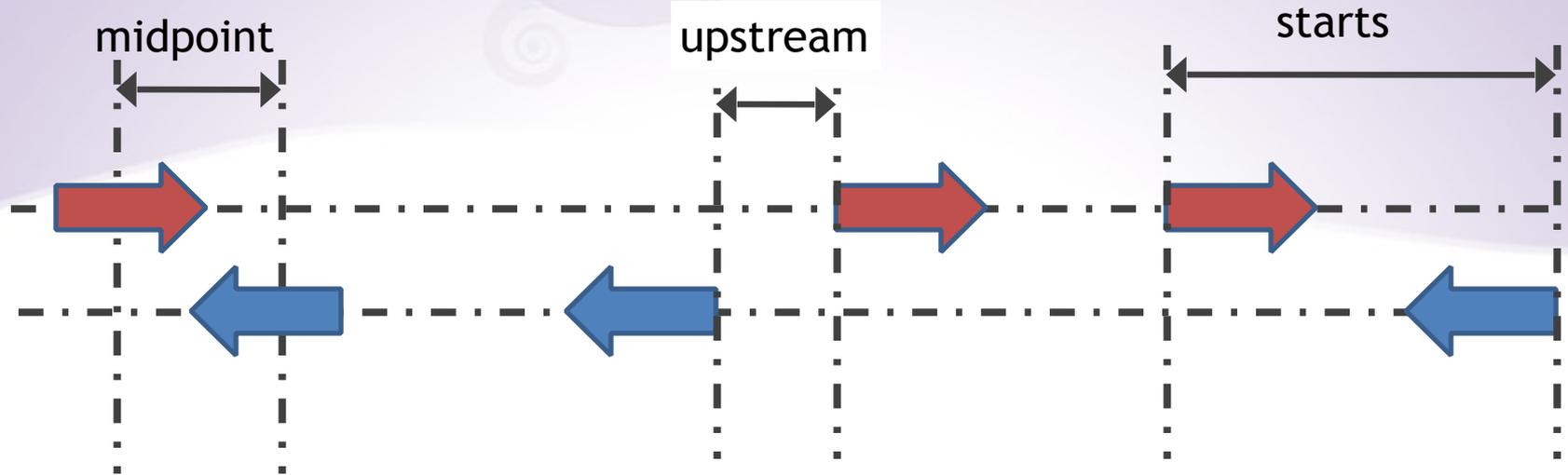
# Interval related tasks

An intervals are not one-dimensional points! -

We need to specify tasks more precisely than for one dimensional points. For example:

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

# Important details

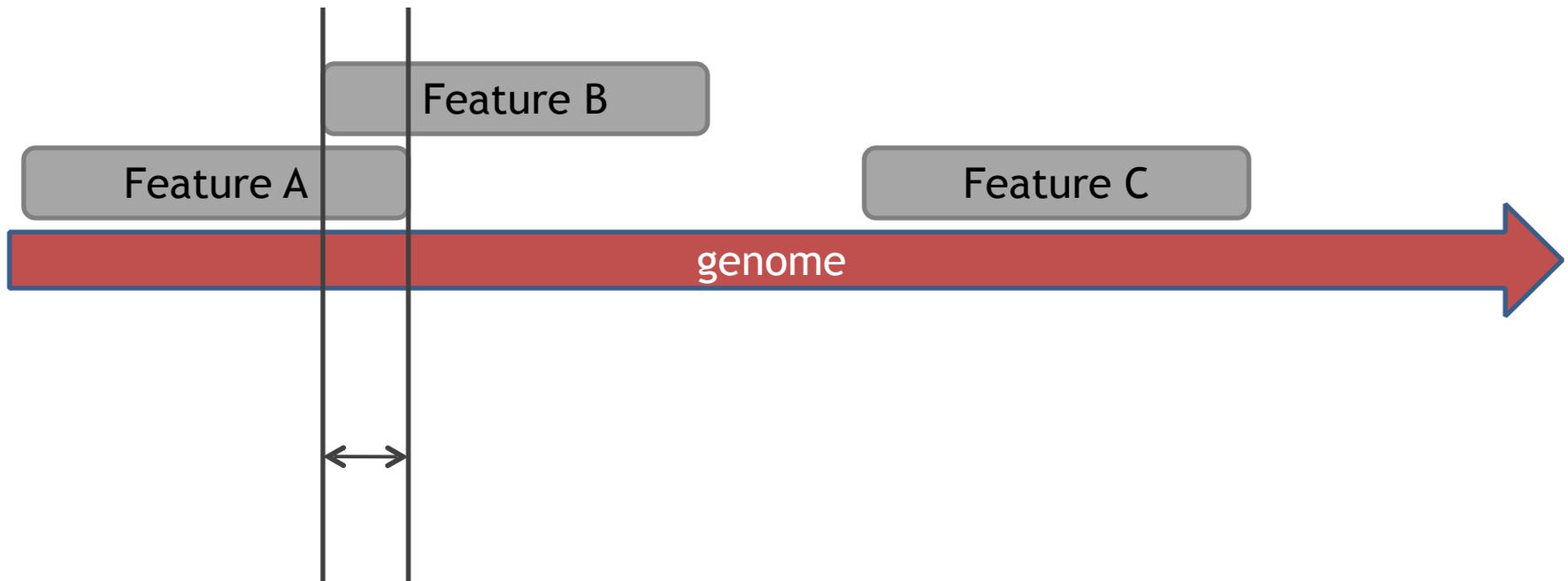


- What are the anchor points (the locations that represent the intervals)
- Which direction does the comparison proceed - upstream, downstream?
- What gets reported?

Often we need to create another transformed interval data that conforms to what we actually need

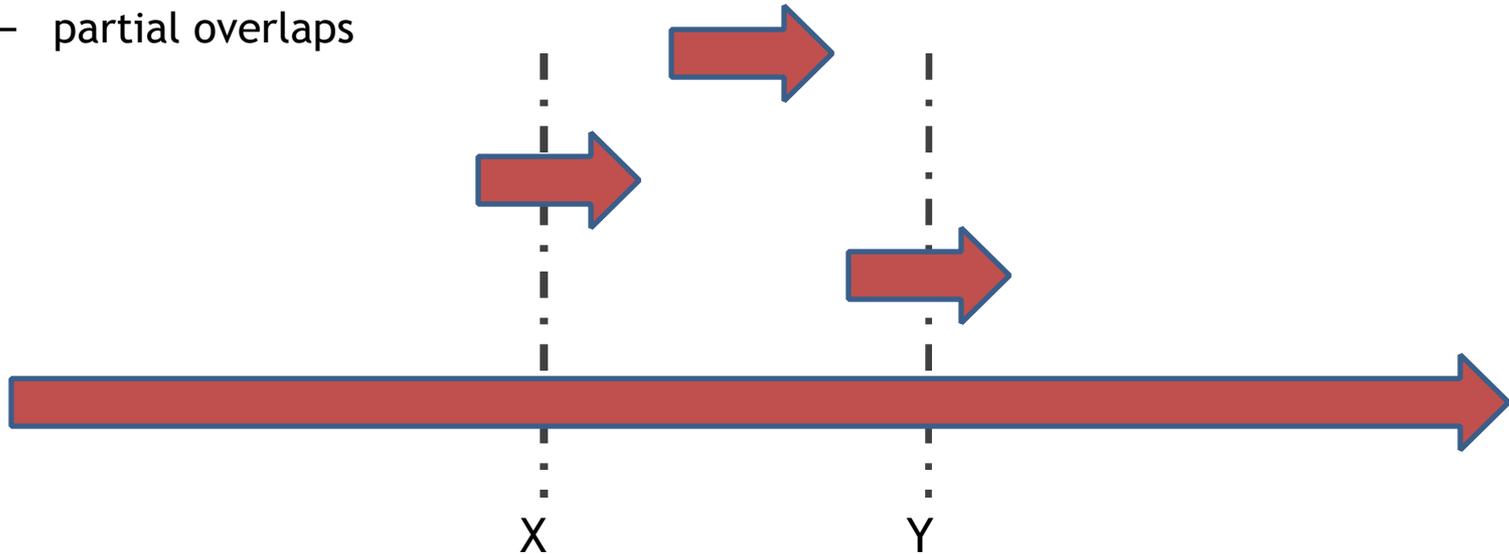
# Overlap/intersect

- Two features are said to overlap or intersect if they share at least one base in common.



# 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):

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

# BedTools: Interval Arithmetics

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

<http://bedtools.readthedocs.org/en/latest/>

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 documentation

## ***bedtools: a powerful toolset for genome arithmetic***

Collectively, the **bedtools** utilities are a swiss-army knife of tools for a wide-range of genomics analysis tasks. The most widely-used tools enable *genome arithmetic*: that is, set theory on the genome. For example, **bedtools** allows one to *intersect, merge, count, complement, and shuffle* genomic intervals from multiple files in widely-used genomic file formats such as BAM, BED, GFF/GTF, VCF. While each individual tool is designed to do a relatively simple task (e.g., *intersect* two interval files), quite sophisticated analyses can be conducted by combining multiple bedtools operations on the UNIX command line.

## **Interesting usage examples**

To whet your appetite, here are a few examples of ways in which bedtools has been used for genome research. If you have interesting examples, please send them our way and we will add them to the list.

- [Coverage analysis for targeted DNA capture. Thanks to Stephen Turner.](#)
- [Measuring similarity of DNase hypersensitivity among many cell types](#)
- [Extracting promoter sequences from a genome](#)
- [Comparing intersections among many genome interval files](#)
- [RNA-seq coverage analysis. Thanks to Erik Minikel.](#)
- [Identifying targeted regions that lack coverage. Thanks to Brent Pedersen.](#)

# 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)

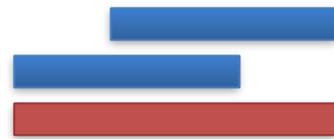


← before  
← after

– 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

```
lec22 -- ~/work/lec22 -- bash -- 74x17
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

```
lec22 — ~/work/lec22 — bash — 68x13

ialbert@porthos ~/work/lec22
$ ~/bin/bedtools slop -i demo.bed -g genome.txt -l 10 -r 0
chrI    90      200     one     0       +
chrI    290     400     two     0       -

ialbert@porthos ~/work/lec22
$ ~/bin/bedtools slop -i demo.bed -g genome.txt -l 10 -r 0 -s
chrI    90      200     one     0       +
chrI    300     410     two     0       -

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



It is very important to understand what happens here.  
It can be occasionally feel counterintuitive

# BedTools is format aware for input

```
lec22 — ~/work/lec22 — bash — 68x13

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!

```
lec22 — ~/work/lec22 — bash — 65x10
ialbert@porthos ~/work/lec22
$ ~/bin/bedtools complement -i demo.gff -g genome.txt
chrI      0         100
chrI      200       300
chrI      400       3000
chrII     0         813184

ialbert@porthos ~/work/lec22
$ █
```

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

# Slop vs Flank

```
lec22 — ~/work/lec22 — bash — 69x13

ialbert@porthos ~/work/lec22
$ ~/bin/bedtools slop -i demo.bed -g genome.txt -l 10 -r 0 -s
chrI    90      200     one     0       +
chrI    300     410     two     0       -

ialbert@porthos ~/work/lec22
$ ~/bin/bedtools flank -i demo.bed -g genome.txt -l 10 -r 0 -s
chrI    90      100     one     0       +
chrI    400     410     two     0       -

ialbert@porthos ~/work/lec22
$ █
```

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”