Unix, Perl, and Python

Session 2: Perl for Bioinformatics

Exercise 1: Converting a SAM short-read alignment file into a BED file

Goal: Read through a SAM file and create a BED file that is much smaller (since it includes less information).

The script you'll write will take as input a sorted SAM alignment file that was created by an unspliced alignment tool like bowtie (such as part of a ChIP-Seq experiment). A SAM file is a tab-delimited text file that shows information such as the short read sequence, the genome region to which it maps, and even details about where any mismatches occur. You may be able to figure out most of the fields in the file, but the complete file format is described here: http://samtools.sourceforge.net/SAM1.pdf (in section 1.4). The file often starts with a header that describes the length of each chromosome in the genome. Some SAM files also include unmapped reads, but this one only includes those mapped to the mouse genome.

See **`http://jura.wi.mit.edu/bio/education/hot_topics/Unix_Perl_Python/`**
for the course page

This exercise will exclusively use Perl commands (rather than some Unix commands such as in exercise 2). When you're finished, you can upload your BED file to the Mouse July 2007 UCSC genome browser (http://genome.ucsc.edu/cgi-bin/hgGateway?db=mm9).

To do:

1	Log onto tak, which should take you to your home directory.
2	Create a directory (if you haven't already) called `perl_class` and enter it.
3	Copy starting script and data file from `/nfs/BaRC_Public/Unix_Perl_Python/Perl/` to your current directory. script = `SAMtoBED.pl` data file = `Mouse_mm9_reads.sam`
4	Check permissions of script and change to executable if necessary. `chmod +x SAMtoBED.pl`
5	Open `SAMtoBED.pl` with pico (command line), nedit (Xwindows), or another text editor. It may be helpful to look at `Mouse_mm9_reads.sam` too (using a text editor or the `more` command). This SAM file is just the top piece of a complete file that could be 5 G or more (so normally you can't open the whole thing).

6	Under the line “# 0”: Print an error message if the user forgets the name of the sam file, which should be the first argument (**`$ARGV[0]`**).
7	Run the script, check the output, and debug: `./SAMtoBED.pl`
8	Under line “# 1” is an “if” command. Do you have any idea of what this expression means? Hints: It has something to do with the SAM header, and **`^`** is a special character that, in this context, refers to the beginning of a line. Answer: It’s a regular expression that says to enter the loop only if the line does not begin with a @ (which means to skip the header lines)
9	Under line “# 2”: Use the tabs "\t" to split the line into fields, and place these fields into an array called **`@data`** A command like this appears in the class slides.
10	Under line “# 3”: Get the starting position of the mapped read in Field 4 (**`$data[3]`**) is the start position of the mapped read (where the first base of a chromosome is 1). We want to create a BED file but BED format counts the first base of a chromosome as 0, so we need to subtract 1 from the SAM coordinate.
11	Under line “# 4”: You have the starting coordinate of the read. To get the “end” coordinate, you need to know the read length, which you can get from the **`length`** command applied to the short read (field 10). Call the length, **`$length`**.
12	Under line “# 5”: Calculate the “end” coordinate (from **`$start`** and **`$length`**) and call it **`$end`**.
13	Under line “# 6”: As a first step, convert each row into a BED-format row (called **`$bedRow`**) with the following tab-delimited fields: chr start end The chr is found in field 3 and you’ve already calculated **`$start`** and **`$end`**. Print **`$bedRow`**

14	Under line “#7 [Optional]”: Add fields for read name, score, and strand. To create **`$name`**, use the first field or the file row (a special Perl character: **`$.`**). To create **`$score`** (between 0 and 1000), just choose your favorite number or use the number of mismatches to create a scoring scheme. The last digit of the last field ($data[13]) contains the number of mismatches: **`$numMismatches = substr ($data[13], -1, 1);`** To get **`$strand`**, this information is encoded in field 2 ($data[1]). If field 2 is 0, it's +; if field 2 is 16, it's -. Create a new expanded **`$bedRow`** or add to the previous **`$bedRow`** with a command like **`$bedRow .=`** Print **`$bedRow`**
15	One possible solution (a completed script) is in `/nfs/BaRC_Public/Unix_Perl_Python/Perl/solutions/`