

## Problem Set 4

**Part 1: Due Friday 2/24/12 by 9:00 am**

**Part 2: Due Friday 3/9/12 by 9:00 am**

### Introduction

For this problem set, you're going to use real data from an ultra-high-throughput sequencing project to look for differential allelic expression in the developing mammalian brain. Gregg et al. ([Science 2010](#)) performed non-directional RNA-seq on homogenates of embryonic (day E15) mouse brains from reciprocal crosses of the reference strain *C57BL6/J* (“BL6,” “black six”) and the subspecies *Mus musculus castaneus* (*Cast/Ei*, “Cast”), and produced over 120 million short reads per cross on the Illumina Genome Analyzer platform. This allowed the researchers to measure allele-specific gene expression: in the F1 progeny of a cross between purebred strains, any Cast allele must have come from the Cast parent and BL6 allele from the BL6 parent (whose sexes are known), so the relative abundance of SNP-straddling RNA-seq reads can reveal differential regulation of the alleles. *Make sure you understand this experimental design before you go on.*

Although Gregg et al. used parental RNA-seq data to call their SNPs, we are going to take a more traditional route and use genomic DNA reads from the Sanger Institute's *castaneus* resequencing project to first call SNPs relative to BL6, then use those SNP calls to direct our study of allele-specific expression in the RNA-seq data.

**Q1. (a) The RNA-seq reads are 36 bases long and the size of the mouse genome is approximately 3 Gb. Assuming the entire genome is transcribed (as some have claimed), what is our “coverage,” i.e. how many times was each nucleotide of the genome read on average? Remember that there are 120 million short read pairs per cross. (b) According to the literature, the substitution rate between BL6 and Cast is approximately 0.005 SNPs per bp. How far apart are SNPs from each other on average? Would you expect them to be more or less dense in genes than in intergenic regions (and why)?**

**Q2. In this problem set we're only using data from one of the two crosses. What information would we gain by comparing these results with the reciprocal cross?**

Ultra-high-throughput sequencing produces a lot of information: even with compression, the genomic DNA alignments are over 80 GB in size! To save time and space, in this problem set we'll only consider a subset of the genome (the last 5 Mb of chromosome 7) – this still leaves us with big files (140 MB), so we have compressed all of them with `gzip`. You can't view the contents of a gzipped file directly, but you can stream it through `gzip` and redirect the output into your favorite text viewer, e.g.

```
gunzip < yourfile.gz | less
```

Likewise, you can use redirection to get Perl to read/write gzipped files, using the `open()` command:

```
open (INPUT, "gunzip < input_file.gz |") or die "error opening input: $!\n";
```

## Part 1

In this part, you will write a SNP caller to identify SNPs between *Cast* and the *BL6* reference sequence. To save you time, we've done the preliminary steps for you: we downloaded a BAM file (binary, compressed SAM) from the Sanger website that contains over a billion short reads aligned against the BL6 reference, and we used SAMtools to create a pileup file, which is the data format you will be working with in this part of the problem.

In pileup format, each line represents one nucleotide of the reference sequence, so for the mouse genome, a whole-genome pileup file will be over 2.7 billion lines. This is too much data to deal with for a problem set, so we've given you a subset of chromosome 7 that has less than 5 million lines. The format of the pileup file is tab-delimited, with the following fields:

- 1 - Reference chromosome
- 2 - Reference position (1-indexed)
- 3 - Reference base ("forward" strand)
- 4 - Sequence coverage
- 5 - Sequence pile
- 6 - Sequence base qualities

The 5th field, for which this format is named, has a single character for every read that mapped to the reference genome at this position. Additionally, the pile shows whether a base is the first or last base in a read, as well as whether insertions/deletions (indels) span the reference base. Here is an example line from a pileup file, with tab-delimiters explicitly shown and bolded:

```
chr1<tab>50<tab>C<tab>34<tab>.$.$.$.$T,..A.....+4AGGC.....,a,^F,^].^F,<tab>[[`p!
S__^\\a`ZU`X`B_Va^____`^[_a\W
```

Below is a table showing the valid characters in a pile, as well Perl regular expressions one could use to extract the given characters from the pile. For indels, the below lengths are just examples. Indels can be of any length.

Character(s)	Meaning	Suggested Perl Regular Expression
.	Match to reference base on forward strand	/\./
,	Match to reference base on reverse strand	/,/
ACGT	Mismatch on forward strand	/[ACGT]/
acgt	Mismatch on reverse strand	/[acgt]/
^x	Start of a read, where x is the mapping quality as ASCII character - 33(see below); precedes the character that indicates the nucleotide at that position from the read	/\^../
\$	End of read; follows the character that indicates the nucleotide at that position from the read	/\\$/
+2AA	Insertion of two bases relative to the reference. '+' indicates an insertion, '2' means it is 2 bases in length, 'AA' means that two 'A' bases are inserted.	/\+d+[ACGT]+/
-3ACG	Deletion of three bases relative to the reference. Same conventions as insertions, except a '-' signifying a deletion.	/\-d+[ACGT]+/

For the example line above, the reads map to chromosome 1, position 50. The reference base is a C and the sequencing coverage at this position is 34x. There are 4 reads that end at this position (represented by '\$'), and there are 3 reads that start at this position (represented by '^x', where 'x' is an ASCII character - 33, which

represents that read's mapping quality - in this case `^^F'`, `^^J'`, and `^^F'`). There is also one 4-base insertion relative to the reference (+4AGGC). There are 2 reads that have an 'A' at this position, and one read that has a 'T'. Finally, there are 31 reads that have a 'C' at this position, the same as the reference base, so they are represented by either a dot or comma, depending on strand. Notice that only dots, commas, and 'ACGTacgt' characters not involved in mapping quality or indels contribute to overall coverage. This is because the read start (^x) and read end (\$) indicators only serve as landmarks and do not contain sequence information. Indels also do not contribute to coverage since they are not present in the reference genome.

The 6th field contains the base qualities. There is one base quality character for each character contributing to coverage in the pile, so in the example above, there are 34 base quality characters. The conversion is the same as with mapping qualities, where the integer mapping quality is calculated by taking the ASCII value of the character minus 33. In Perl, this can be done using the `ord()` function:

```
my $quality = ord("P") - 33; # 47 in decimal
```

### Goal:

**Write a SNP caller** named `snp_caller.pl` that parses the pileup file line-by-line, and will call SNPs based on two criteria: sequencing coverage and the proportion of consensus mismatches relative to the reference. **Call a SNP if**

(1) the sequencing coverage is greater than or equal to 5X, but less than 100X, and

(2) the percentage of consensus *mismatches* relative to the reference is greater than or equal to 80% of the total number of bases in the pile.

However, **ignore bases whose sequencing quality score is below 20**. That is, do not use them to compute either the proportion of consensus mismatches, or the total coverage.

Additionally, upon calling a SNP, your code should be able to parse a gene annotation file (which we will supply). Your parser should **output the gene(s) (if any) that the SNP resides in** (more about this below).

### Methods:

SNP calling from piles is essentially an exercise in regular expressions. Since the piles are riddled with mapping qualities that can contain 'ACGTacgt' following a carat (^), as well as indels, you must parse the pile carefully, taking care to **only count the characters indicating a mismatch, and not indicating mapping qualities or indels**. That is, a mapping quality such as '^A' that occurs in the pile should not be taken as an 'A' from sequencing, but rather seen as a mapping quality and ignored. Similarly, indel characters such as '-5ATGCC' or '+2AA' should not contribute to coverage.

Output should have one SNP per line, with the following fields in tab-delimited format:

- 1 - Reference chromosome
- 2 - Reference position
- 3 - Gene(s) containing SNP
- 4 - Reference base
- 5 - Consensus base
- 6 - Proportion of consensus base
- 7 - Sequence pile
- 8 - Sequence base qualities

The following is a breakdown of the fields you should have in the output file, including where the values are coming from and if any calculation or processing is necessary to create the value:

### 1 – Reference chromosome

Take directly from pileup file. No processing necessary. Should always be '7'.

### 2 – Reference position

Take directly from pileup file. No processing necessary.

### 3 – Gene(s) containing SNP

Parse the file `/usr/class/gene211/misc/mouseGenes_chr7.txt` to find which gene(s) contain the SNP of interest. The format of `mouseGenes_chr7.txt` is as follows, in tab-delimited columns:

- 1 – Reference chromosome
- 2 – Gene start coordinate
- 3 – Gene stop coordinate
- 4 – RefSeq gene ID
- 5 – Gene name

The stop coordinate is always larger than the start coordinate, regardless of which strand the gene is on. Column 5, the gene name, should be reported in your output. Genes can overlap, so some SNPs will reside in multiple genes. In these cases, **report all the genes that a SNP is in**, and comma-delimit them in the output. For example, if the current SNP is in both *Tcf7l2* and *Ppnr*, your output for this column should be `'Tcf7l2,Ppnr'`. Oftentimes, SNPs fall within intergenic regions. In these cases, there will be no gene to report, so instead output `'int'` for this column.

### 4 – Reference base

Take directly from pileup file. Note that the reference base can be either uppercase or lowercase depending on whether the base is in a repeat-masked region as defined by the UCSC genome browser. You should not concern yourself with this and simply output it like it is in the pileup file.

### 5 – Consensus base

You must process the pile to get this information. The consensus base is the non-reference base with the highest frequency in the pile. Make sure to **treat uppercase and lowercase bases of the same letter as the same base** in your analysis when finding the consensus. The same is true for periods and commas when finding the total coverage.

### 6 – Proportion of consensus base

Again, process the pile to get this information. As mentioned above, **only high-quality bases should count towards the total and consensus coverage**. As such, you should not use the coverage supplied in the pileup file to calculate this proportion. Instead, calculate your own coverage as you parse the pile and use that coverage to test against the input coverage filters and also to calculate the proportion of consensus bases. This proportion ends up being the number high-quality ( $\geq 20$ ) consensus bases in the pile divided by the total number of high-quality bases in the pile. Since our cutoff for this parameter is 0.8, you should not see any numbers less than 0.8 in your output. **Round the proportion output to three decimal places**. One way to do this is using the `sprintf()` function:

```
my $rounded_prop = sprintf("%.3f", $proportion);
```

### 7 – Sequence pileup

Take directly from pileup file.

## 8 – Sequence base qualities

Take directly from pileup file

**Use the command line (@ARGV) to supply the cutoffs** for sequencing coverage ( $\geq 5$ ;  $< 100$ ), consensus mismatch proportion ( $\geq 0.8$ ), and base quality ( $\geq 20$ ). Also, **pass in the mouse gene annotation filename on the command line**. Use `gunzip` inside your script to read the pileup file as described above, and supply the name of the pileup file also on the command line.

If you look ahead to the next section, you will realize that SNP calling on pileup files is also part of determining allele-specific expression. With that in mind, **write a Perl library named `parse_pileup.pl`** that, at the very least, contains a subroutine to parse a line of a pileup file and get the count of each nucleotide at that position (you are encouraged to add any other functions that you think you might want later). **Call this library in `snp_caller.pl`** at the top via `require "parse_pileup.pl";`, and keep the library generalized and self-contained so that you can reuse it in your next script. Remember that the last thing evaluated in a Perl library must return true (e.g. end the file with `1;`).

Run `snp_caller.pl` on `cast_chr7_sub.pileup.gz` and `mouseGenes_chr7.txt`, which are on the server in `/usr/class/gene211/misc/`

Name the output file `snps.txt` using the following command to call your script:

```
perl snp_caller.pl 5 100 0.8 20 cast_chr7_sub.pileup.gz mouseGenes_chr7.txt > snps.txt
```

where the full paths of `cast_chr7.pileup.gz` and `mouseGenes_chr7.txt` are shortened for readability. The output file `snps.txt` should have around 30,000 SNPs.

**Q3: Look through your output for SNPs at positions 149,842,141 and 149,852,533, then go to the UCSC Genome Browser, <http://genome.ucsc.edu/>, and navigate to those positions in the mm9 mouse genome assembly. Look at the mammalian multiple alignment for these positions. What is a probable explanation as to how the evolutionary histories of these two SNPs differ?**

**If you are unfamiliar with the UCSC genome browser and wish to have a tutorial, please check out this link: <http://www.openhelix.com/ucsc>.**

### Determining allele-specific expression

Now that you've called SNPs, you're ready to measure gene expression at those sites. Specifically, for each SNP, you want to know the number of reads of each allele that overlap it, and with these numbers you can ask whether the alleles are expressed equally. This can be done most easily with a pileup file; we've already put one called `BL6_x_Cast_RNA_chr7.pileup.gz` in the usual place.

**Goal: Write a script called `allele_expression.pl`** that reads the list of SNPs in `snps.txt` and finds the allele-specific expression at each, from a pileup file. That is, for each SNP, count the number of RNA-seq reads with the reference allele and the number with the Cast allele. Your script should take the pileup file name, SNP file name, a base quality cutoff, and coverage lower bound (like in `snp_caller.pl`) as arguments. We won't set an upper bound on coverage because we expect RNA-seq to be highly heterogeneous. **Use your script to make an output file called `snp_expression.txt`** in a format similar to `snps.txt`: leave the first three columns intact, replace the reference and Cast bases with the *count* of each, report the proportion of Cast bases (see note below), and omit the pileup and quality scores, for six columns total. Only include high-quality bases in the SNP calculation ( $\geq 20$ ), and report only SNPs with at least 10 reads between the Cast and BL6 alleles combined.

**Method:** Because the SNP file and the pileup file are both sorted by position, you don't need to load either one into memory before you process the other. Write `allele_expression.pl` so that it streams through both files one line at a time. That is, for each line (SNP) in `snps.txt`, your script should advance to the appropriate line in `BL6_x_Cast_RNA_chr7.pileup.gz` if it exists, then output a line of `snp_expression.txt` before continuing to the next line of `snps.txt`, so that only one SNP's information needs to be in memory at any given time. Note: when you calculate the proportion of Cast-allele reads for a given SNP, **do not count reads that have an allele different from the Cast or reference alleles**, i.e. the denominator should only be the sum of the reference-allele reads and Cast-allele reads.

You will notice that allele-specific expression analysis is a lot like SNP calling, especially since you're working with another pileup file. **Reuse your pileup-parsing library**, `parse_pileup.pl`, in your new script in order to avoid having to rewrite similar functions.

Run your second script on your first script's output with this command:

```
perl allele_expression.pl BL6_x_Cast_RNA_chr7.pileup.gz snps.txt 20 10 >
snp_expression.txt
```

**Q4: Browse through your output for the *H19* and *Igf2* genes. What allelic expression patterns do you see? Based on known biology, what you would expect to see in the reciprocal cross? (You will want to consult the literature but your answer does not need to be too thorough.)**

#### **Submission of Part 1:**

Put the answers to the questions in a file `README`. Submit `snp_caller.pl`, `allele_expression.pl`, `parse_pileup.pl`, `snps.txt`, `snp_expression.txt`, and `README` in the usual way.

## Part 2

### View allelic expression in the UCSC browser

One way to visualize high-throughput sequencing data is to create custom tracks for the [UCSC Genome Browser](http://genome.ucsc.edu/goldenPath/help/customTrack.html#BED). The standard format for the UCSC browser is Browser Extensible Data (BED), which we encountered briefly in PS3 and which is explained in detail at <http://genome.ucsc.edu/goldenPath/help/customTrack.html#BED>. Your BED file will be customized to show each read as a separate feature, color-coded by strand and by whether or not it overlaps a SNP, and each feature will show the sequence of the read with any SNP positions highlighted.

Aligned reads from the RNA-seq experiments are stored in a compressed file called `b16_x_cast_chr7.sam.gz` in the same place as our other data. This file contains reads from two lanes on the Illumina Genome Analyzer 2 that have passed a quality filter and have been mapped against the `mm9` reference genome by Bowtie. It is in the Sequence Alignment/Map (SAM) format, which is tab-delimited and somewhat complicated, but the important columns for this problem set are:

- 2. bitwise flag including strand: for us, **0** means the read aligned to the forward strand, **16** reverse
- 4. leftmost coordinate of the aligned read (**1-based**)
- 10. sequence, from left to right regardless of the tag's orientation

**Goal:** Write a script called `sam_to_bed.pl` that reads the list of called SNPs from `snps.txt`, then parses through the SAM files and generates a **compressed** BED file (since it will be large) with a line for each read. Use the following RGB color codes (numbers and commas only, no spaces):

<b>No SNP or multiple SNPs with different alleles</b>	0,0,0 (black)
<b>SNP(s) with reference allele</b>	255,0,0 (red)
<b>SNP(s) with <i>Cast</i> allele</b>	0,0,255 (blue)

Your script should take the read file name, SNP file name, and output file name as arguments. Run it on the files above to generate `f1_rna-seq.bed.gz` with the following command:

```
perl sam_to_bed.pl b16_x_cast_chr7.sam.gz snps.txt f1_rna-seq.bed.gz
```

**Method:** In BED format, note that “chromStart” is always less than “chromEnd,” even if the feature is on the reverse strand, i.e. **coordinates are always left to right**. Also note that unlike in SAM, **the start position is 0-indexed but the end position is 1-indexed**. For the “name” field, use the sequence of the read as given in SAM, but **make all bases lowercase except those at positions you identified as SNPs**. We don't need the “score,” “thickStart,” or “thickEnd” fields, but they're not allowed to be empty, so just put a zero (“0”) in each. The color goes in the “itemRgb” field. Here is an example:

```
chr19<tab>4964825<tab>4964861<tab>ggggCccacagaagaagctgagccaaaggacttcat<tab>0<tab>+<tab>0<tab>0<tab>0,0,255
```

Finally, in order to display the reads in a compact form and in color, put the following track header in the first line of the BED file:

```
track name=F1_RNA-seq visibility=squish itemRgb=On db=mm9
```

You can use the `open()` command to `gzip` your output in Perl, just like `gunzip`:

```
open OUTPUT, "| gzip > output_file.gz" or die "error writing output: $!\n";
```

**Recommendation:** Make a lookup table of SNPs by position. For each read in the SAM file, scan the entire table to find SNPs hit by your read. You can use the power of Perl hashes to do this efficiently: **design your lookup table as a hash** where each key is a position and each element is a pointer to some other kind of data structure that contains all the information you need. Then for each RNA-seq read, **iterate over the positions it covers**, and for each of those, use the `exists()` function to test if that key exists in the hash table. This is orders of magnitude faster than iterating over the hash, even though it basically does the same thing!

## Using the browser

Now you can load your BED file in the browser. If you are working on a Stanford server, a fast way to do this is to host the file in your personal webspace (your `~/www/` directory on AFS). Go to the UCSC Genome Bioinformatics website (<http://genome.ucsc.edu/>) and click on the “Genomes” tab at the top of the page. Select the Mouse July 2007 assembly (mm9), then click “Add custom tracks.” Now if you hosted your file on a webserver, you can paste its URL (on the AFS, it will take the form `http://www.stanford.edu/~yourusername/yourfile.bed.gz`) and the browser will read it from there without ever having to go through your personal internet connection. Otherwise, upload the file directly from the computer you are using. The browser has no problem reading gzipped files.

You can verify that your scripts work correctly by changing the custom track's mode to “pack.” This will display each feature's “name” alongside it (and it will pop up if you mouse over it), so compare the bases in your read with the bases in the reference (click the “base” button near the top to zoom in far enough that the nucleotide sequence is shown). Also note that you can zoom in on a particular area by dragging a box in the position track at the top of the browser area. To go to a specific gene or position, type it in the search box. Or, you can zoom out to the whole chr7 (reads will be collapsed into a bar if there are too many to display) and pick genes on which to zoom in. You may want to click the “Configure” button and enable “Next/previous item navigation” so you can click on the arrows on the sides of the browser to skip from one feature to the next. As for annotations, “UCSC Genes” is a comprehensive track for all annotated genes, but feel free to experiment with other tracks and features.

**Q5. Browse through the exons of *Brsk2* (you will have to zoom in because the browser limits how many features you can display at once). Are the RNA-seq reads distributed evenly over the length of the gene? What are one biological phenomenon and one technical artifact that could create a skew?**

**Q6: In your allelic expression output, there is an interesting pattern in the intergenic region between *Rnh1* and *Hras1*. Look at this region on the genome browser. What might be going on?**

**Q7: The expression ratios for the *Ctnn* gene behave strangely. Look at it on the browser. What does this imply about the gene's behavior in the two mouse strains? (Hint: look at an mRNA track.)**

**Q8: If the Cast allele of a gene is underexpressed relative to the BL6 allele, and this is true in the reciprocal cross as well, what might be causing the difference? Where might we look for a called SNP to explain it?**

## Submission of Part 2:

Put your answers in `README` and submit `sam_to_bed.pl` and `README` the usual way. Since the BED file is large, we will run your script to generate it ourselves, so you don't need to submit it.