# 1 Answers

To run the commands alongside the answers you must first have run *all* of the tutorial commands for that section and generated the output files they reference.

## 1.1 Introducing the tutorial dataset

### 1.1.1 Q1: Why is there more than one FASTQ file per sample?

There are **2** FASTQ files for each sample e.g. MT1_**1**.fastq and MT1_**2**.fastq. This is because this was **paired-end** sequence data.

```
ls data/MT1*.fastq.gz
```

```
ls data/MT1*.fastq.gz | wc -l
```

With Illumina paired-end sequencing, you will have a fragment (hopefully longer than your reads) which is sequenced at both ends. This means that there will be a "left" read (sometimes known as *r1*) and its corresonding "right" read (sometimes known as *r2*). These are indicated by the **/1** and **/2** at the end of the read name in the **_1.fastq and _2**.fastq files respectively.

So, the left read header _@HS18_08296:8:1101:1352:48181#4_ in **MT1_1.fastq** has the **/1** suffix:

```
@HS18_08296:8:1101:1352:48181#4/1
```

And, the corresponding right read header in **MT1_2.fastq** has the **/2** suffix:

```
@HS18_08296:8:1101:1352:48181#4/2
```

### 1.1.2 Q2: How many reads are there for MT1?

There are **2.5 million** reads for MT1.

Ideally, you should count the reads in both files. This is because sometimes we have singletons (reads without a mate) after preprocessing steps such as trimming.

Our reads look like:

```
@HS18_08296:8:1101:1352:48181#4/2
ATCCGCCNANTTTNNNNATATAATTANNNGNAANNAANNNNAATNACANNNATTNNNNTAGNANNNGNNAGTNNACAAGGNTNNNNNNNAAAGNN
+
9AABE8A!D!DFA!!!!EE@CFCD@B!!!D!F6!!EE!!!!EE4!F5E!!!BEA!!!!B@6!A!!!D!!FD'!!C+D@@*!B!!!!!!!B>DE!
```

With the FASTQ format there are four lines per read. So, as long as our files are not truncated we can count the number of lines and divide them by four.

```
zless data/MT1_1.fastq.gz | wc -l
```

So, we can then divide this by four to give us 1.25 million. We will get the same for our r2 reads:

```
zless data/MT1_2.fastq.gz | wc -l
```

So, we have 1.25 million *read pairs* or 2.5 million (1.25 x 2) *reads* for our MT1 sample.

You may also have thought "aha, I can use the **@** which is at the start of the header"…

```
zgrep -c '^@' data/MT1_1.fastq.gz
```

```
zgrep -c '^@' data/MT1_2.fastq.gz
```

But, wait, there are 1,343,714 left reads and 1,250,000 right reads…can that be right…*no*.

Take a closer look at the quality scores on the fourth line…they also contain **@**. This is because the quality scores are Phred+33 encoded. For more information on quality score encoding see our Data Formats and QC tutorial or go here.

Instead, we can use the earlier information about the left and right read suffixes: **/1** and **/2**. This can be with two commands:

```
zgrep -c '/1$' data/MT1_1.fastq.gz
```

```
zgrep -c '/2$' data/MT1_2.fastq.gz
```

Or, with only one command:

```
zgrep -c '/[12]$' data/MT1*.fastq.gz
```

*Note: Don't forget to use the* `-c` *option for* `grep` *to count the occurences and* `$` *to make sure you're only looking for \1 or \2 at the* ***end of the line****.*

---

## 1.2 Mapping RNA-Seq reads to the genome using HISAT2

### 1.2.1 Map, convert (SAM to BAM), sort and index using the reads from the MT2 sample.

You can do this with several, individual steps:

```
hisat2 --max-intronlen 10000 -x data/PccAS_v3_hisat2.idx \
-1 data/MT2_1.fastq.gz -2 data/MT2_2.fastq.gz -S data/MT2.sam
```

```
samtools view -b -o data/MT2.bam data/MT2.sam
```

```
samtools sort -o data/MT2_sorted.bam data/MT2.bam
```

```
samtools index data/MT2_sorted.bam
```

Or, you can do it in one step:

```
hisat2 --max-intronlen 10000 -x data/PccAS_v3_hisat2.idx \
-1 data/MT2_1.fastq.gz -2 data/MT2_2.fastq.gz \
| samtools view -b - \
| samtools sort -o data/MT2_sorted.bam - \
&& samtools index data/MT2_sorted.bam
```

Or, you could write a for loop to run the command above for all five of your samples:

```
for r1 in data/*_1.fastq.gz
do
  sample=$(basename $r1)
  sample=${sample/_1.fastq.gz/}
  echo "Processing sample: "$sample
  hisat2 --max-intronlen 10000 -x data/PccAS_v3_hisat2.idx \
  -1 "data/${sample}_1.fastq.gz" -2 "data/${sample}_2.fastq.gz" \
  | samtools view -b - \
  | samtools sort -o "data/${sample}_sorted.bam" - \
  && samtools index "data/${sample}_sorted.bam"
done
```

For more information on how this loop works, have a look at our Unix tutorial and Running commands on multiple samples.

### 1.2.2 Q1: How many index files were generated when you ran `hisat2-build`?

There are **8** HISAT2 index files for our reference genome.

```
ls data/*.ht2
```

```
ls data/*.ht2 | wc -l
```

### 1.2.3 Q2: What was the *overall alignment rate* for each of the MT samples (MT1 and MT2) to the reference genome?

The *overall alignment rate* for MT1 was **94.16%** and % for MT2.

```
1250000 reads; of these:
  1250000 (100.00%) were paired; of these:
    105654 (8.45%) aligned concordantly 0 times
    329304 (26.34%) aligned concordantly exactly 1 time
    815042 (65.20%) aligned concordantly >1 times
    ----
    105654 pairs aligned concordantly 0 times; of these:
      1797 (1.70%) aligned discordantly 1 time
    ----
    103857 pairs aligned 0 times concordantly or discordantly; of these:
      207714 mates make up the pairs; of these:
        146078 (70.33%) aligned 0 times
        19877 (9.57%) aligned exactly 1 time
        41759 (20.10%) aligned >1 times
94.16% overall alignment rate

1250000 reads; of these:
  1250000 (100.00%) were paired; of these:
    139440 (11.16%) aligned concordantly 0 times
    483493 (38.68%) aligned concordantly exactly 1 time
    627067 (50.17%) aligned concordantly >1 times
    ----
    139440 pairs aligned concordantly 0 times; of these:
      4967 (3.56%) aligned discordantly 1 time
    ----
    134473 pairs aligned 0 times concordantly or discordantly; of these:
      268946 mates make up the pairs; of these:
        207579 (77.18%) aligned 0 times
        28834 (10.72%) aligned exactly 1 time
        32533 (12.10%) aligned >1 times
91.70% overall alignment rate
```

*Note: If a read pair is concordantly aligned it means both reads in the pair align with the same chromosome/scaffold/contig, the reads are aligned in a proper orientation (typically —-> <—-) and that the reads have an appropriate insert size.*

### 1.2.4 Q3: How many MT1 and MT2 reads were not aligned to the reference genome?

**146,078 reads (5.85%)** and **207,579 reads (8.31%)** did not align to the reference genome for MT1 and MT2 respectively.

Here is a brief summary of what the HISAT2 summary tells us for our MT2 sample and how we can tell which of the summary lines gives us this information:

- We have **1,250,000 read pairs** or **2,500,000 reads** (*2 x 1,250,000 pairs*)

```
1250000 reads; of these:
```

- All of our reads (100%) are paired - i.e. no reads without their mate

```
1250000 (100.00%) were paired; of these:
```

- 1,144,346 pairs (95.24%) align concordantly one (26.34%) or more (65.20%) times

```
329304 (26.34%) aligned concordantly exactly 1 time
815042 (65.20%) aligned concordantly >1 times
```

- 105,654 pairs (8.45%) or 211,308 reads (2 x 105,654) did not align *concordantly* anywhere in the genome

```
105654 (8.45%) aligned concordantly 0 times
```

- Of those 105,654 pairs, 1,797 pairs align *discordantly* (1.70% of the 105,654 pairs)

```
105654 pairs aligned concordantly 0 times; of these:
 1797 (1.70%) aligned discordantly 1 time
```

- This leave us with 103,857 pairs (105,654 - 1,797) where both reads in the pair do not align to the genome (concordantly or discordantly)

```
103857 pairs aligned 0 times concordantly or discordantly; of these:
```

- Of those 207,714 reads (2 x 103,857) we have 61,636 reads (29.67%) which align to the genome without their mate

```
207714 mates make up the pairs; of these:
...
19877 (9.57%) aligned exactly 1 time
41759 (20.10%) aligned >1 times
```

- Leaving us with a **94.16% overall alignment rate**

```
94.16% overall alignment rate
```

- That means **146,078 (5.84%)** of the **2,500,000 reads** (or 70.33% of the unaligned pairs) do not align anywhere in the genome

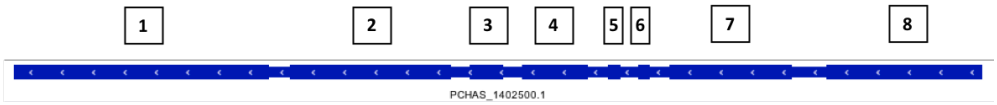
```
146078 (70.33%) aligned 0 times
```

## 1.3 Visualising transcriptomes with IGV

### 1.3.1 Q1: How many CDS features are there in "PCHAS_1402500"?

There are **8** CDS features in PCHAS_1402500. You can get this in several ways:

**Count the number of exons/CDS features in the gene annotation.**




**IGV - PCHAS_1402500 exons**

**Count the number of CDS features in the GFF file.**

First, get all of the CDS features for PCHAS_1402500.

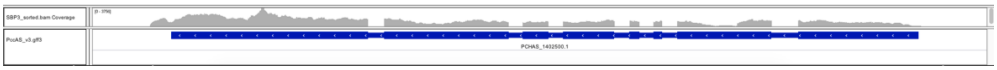
```
grep -E "CDS.*PCHAS_1402500" data/PccAS_v3.gff3
```

Then count the number of the CDS features for PCHAS_1402500.

```
grep -cE "CDS.*PCHAS_1402500" data/PccAS_v3.gff3
```

### 1.3.2 Q2: Does the RNA-seq mapping agree with the gene model in blue?

Yes. The peaks of the coverage tracks correspond to the annotated exon/CDS features.



**IGV - PCHAS_1402500 coverage**

### 1.3.3 Q3: Do you think this gene is differentially expressed and is looking at the coverage plots alone a reliable way to assess differential expression?

Possibly. But, you can't tell differential expression by the counts alone as there may be differences in the sequencing depths of the samples.

## 1.4 Transcript quantification with Kallisto

### 1.4.1 Use kallisto quant four more times, for the MT2 sample and the three SBP samples.

You can run the individual `kallisto quant` commands as you did for MT1 for each of the remaining samples:

```
kallisto quant -i data/PccAS_v3_kallisto -o data/MT2 -b 100 \
data/MT2_1.fastq.gz data/MT2_2.fastq.gz

kallisto quant -i data/PccAS_v3_kallisto -o data/SBP1 -b 100 \
data/SBP1_1.fastq.gz data/SBP1_2.fastq.gz

kallisto quant -i data/PccAS_v3_kallisto -o data/SBP2 -b 100 \
data/SBP2_1.fastq.gz data/SBP2_2.fastq.gz

kallisto quant -i data/PccAS_v3_kallisto -o data/SBP3 -b 100 \
data/SBP3_1.fastq.gz data/SBP3_2.fastq.gz
```

Or, you can write a `for` loop which will run `kallisto quant` on all of the samples:

```
for r1 in data/*_1.fastq.gz
do
  echo $r1
  sample=$(basename $r1)
  sample=${sample/_1.fastq.gz/}
  echo "Quantifying transcripts for sample: "$sample
  kallisto quant -i data/PccAS_v3_kallisto -o "data/$sample" -b 100 \
  "data/${sample}_1.fastq.gz" "data/${sample}_2.fastq.gz"
done
```

For more information on how this loop works, have a look at our Unix tutorial and Running commands on multiple samples.

### 1.4.2 Q1: What *k*-mer length was used to build the Kallisto index?

A *k*-mer length of **31** was used.

Look at the output from `kallisto index`:

```
[build] k-mer length: 31
```

Or, look for the `-k` or `--kmer-size` option in the `kallisto index` usage:

```
kallisto index
```

### 1.4.3 Q2: How many transcript sequences are there in *PccAS_v3_transcripts.fa*?

There are **5177** transcript sequences.

Look at the output from `kallisto quant`:

```
[index] number of targets: 5,177
```

Or, look for **n_targets** in one of the *run_info.json* files:

```
cat data/MT1/run_info.json
```

Or, you can run a `grep` on the transcript FASTA file and count the number of header lines:

```
grep -c ">" data/PccAS_v3_transcripts.fa
```

### 1.4.4 Q3: What is the transcripts per million (TPM) value for PCHAS_1402500 in each of the samples?

| Sample | Transcripts Per Million (TPM) |
|---|---|
| MT1 | 2342.23 |
| MT2 | 1354.42 |
| SBP1 | 2295.24 |
| SBP2 | 3274.98 |
| SBP3 | 2536.17 |

You can look at each of the individual abundance files:

```
grep "^PCHAS_1402500" data/MT1/abundance.tsv
grep "^PCHAS_1402500" data/MT2/abundance.tsv
grep "^PCHAS_1402500" data/SBP1/abundance.tsv
grep "^PCHAS_1402500" data/SBP2/abundance.tsv
grep "^PCHAS_1402500" data/SBP3/abundance.tsv
```

Or you can use a recursive `grep`:

```
grep -r "^PCHAS_1402500" data/*.tsv
```

Or you can use a loop:

```
for r1 in data/*_1.fastq.gz
do
  sample=$(basename $r1)
  sample=${sample/_1.fastq.gz/}
  echo $sample
  grep PCHAS_1402500 "data/${sample}/abundance.tsv"
done
```

For more information on how this loop works, have a look at our Unix tutorial and Running commands on multiple samples.

### 1.4.5 Q4: Do you think PCHAS_1402500 is differentially expressed?

Probably not. We would need to run statistical tests to really be sure though.

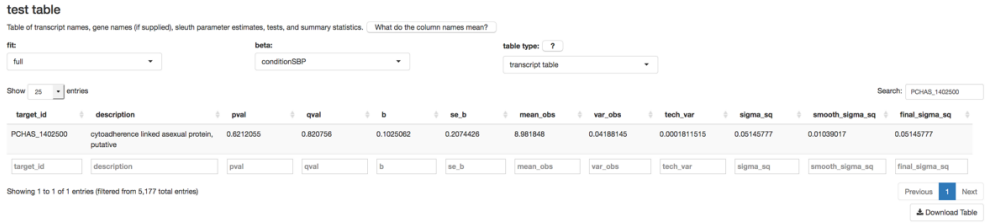
---

## 1.5 Identifying differentially expressed genes with Sleuth

### 1.5.1 Q1: Is our gene from earlier, PCHAS_1402500, significantly differentially expressed?

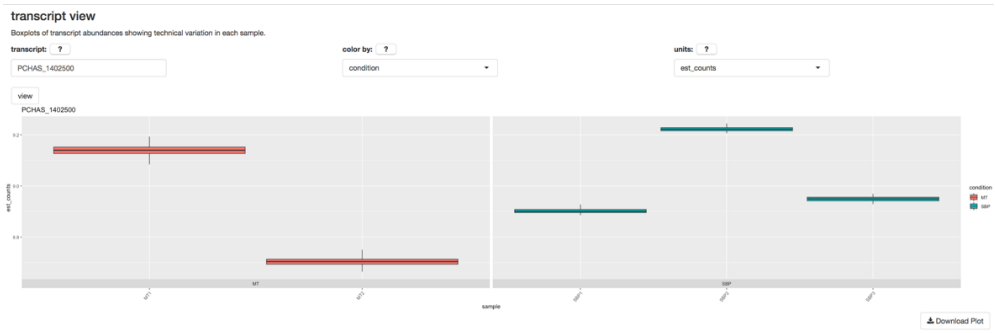
**No.**

Look at the transcript table.




**sleuth - PCHAS_1402500**

And the transcript view.




**sleuth - PCHAS_1402500**

Although this gene looked like it was differentially expressed from the plots in IGV, our test did not show it to be so (q-value $> 0.05$). This might be because some samples tended to have more reads, so based on raw read counts, genes generally look up-regulated in the SBP samples.

Alternatively, the reliability of only two biological replicates and the strength of the difference between the conditions was not sufficient to be statistically convincing. In the second case, increasing the number of biological replicates would give us more confidence about whether there really was a difference.

In this case, it was the lower number of reads mapping to MT samples that mislead us in the IGV view. Luckily, careful normalisation and appropriate use of statistics saved the day!

```
grep PCHAS_1402500 data/kallisto.results | cut -f1,4
```

## 1.7 Normalisation

### 1.7.1 How we got the information to help the questions

To answer the questions you needed the following for each sample:

- Number of reads assigned to PCHAS_1402500
- Length of exons in PCHAS_1402500 (bp)
- Total number of reads mapping
- Total RPK

First, take a quick look at the first five lines of the `abundance.tsv` for MT1.

```
head -5 data/MT1/abundance.tsv
```

There are five columns which give us information about the transcript abundances for our MT1 sample.

| Column | Description |
|---|---|
| target_id | Unique transcript identifier |
| length | Number of bases found in exons. |
| eff_length | *Effective length*. Uses fragment length distribution to determine the effective number of positions that can be sampled on each transcript. |
| est_counts | *Estimated counts*. This may not always be an integer as reads which map to multiple transcripts are fractionally assigned to each of the corresponding transcripts. |
| tpm | *Transcripts per million*. Normalised value accounting for length and sequence depth bias. |

First, look for your gene of interest, **PCHAS_1402500**. Run this as a loop to `grep` the information for all five samples.

```
for r1 in data/*_1.fastq.gz
do
  sample=$(basename $r1)
  sample=${sample/_1.fastq.gz/}
  echo $sample
  grep PCHAS_1402500 "data/$sample/abundance.tsv"
done
```

Now you have the length (eff_length) and counts (est_counts) for PCHAS_1402500 for each of your

samples. Next, you need to get the total number of reads mapped to each of your samples. You can use a loop to do this.

In the loop below `samtools flagstat` gives you the number of mapped paired reads (reads with itself and mate mapped) and those where one read mapped but its mate didn't (singletons). If then uses `grep` to get the relevant lines and `awk` to add the mapped paired and singleton read totals together.

```
for r1 in data/*_1.fastq.gz
do
  sample=$(basename $r1)
  sample=${sample/_1.fastq.gz/}

  total=` samtools flagstat "data/${sample}_sorted.bam" | \
          grep 'singletons\|with itself and mate mapped' | \
          awk 'BEGIN{ count=0} \
               {count+=$1} \
               END{print count}'`

  echo -e "$sample\t$total"
done
```

Finally, to calculate the TPM values, you need the total RPK for each of your samples. Again we use a loop. Notice the use of `NR>2` in the `awk` command which tells it to skip the two header lines at the start of the file. You will also notice that we divide the eff_length by 1,000 so that it's in kilobases.

```
for r1 in data/*_1.fastq.gz
do
  sample=$(basename $r1)
  sample=${sample/_1.fastq.gz/}

  awk -F"\t" -v sample="$sample" \
      'BEGIN{total_rpk=0;} \
       NR>2 \
       { \
         rpk=$4/($3/1000); \
         total_rpk+=rpk \
       } \
       END{print sample"\t"total_rpk}' "data/${sample}/abundance.tsv"
done
```

### 1.7.2 Q1: Using the `abundance.tsv` files generated by Kallisto and the information above, calculate the RPKM for PCHAS_1402500 in each of our five samples.

| Sample | Per million scaling factor | Reads per million (RPM) | Per kilobase scaling factor | RPKM |
|---|---|---|---|---|
| MT1 | 2.353922 | 1079.450 | 3.697 | **292** |
| MT2 | 2.292421 | 1479.484 | 3.709 | **399** |
| SBP1 | 2.329424 | 6269.662 | 3.699 | **1695** |
| SBP2 | 2.187862 | 7908.131 | 3.696 | **2140** |
| SBP3 | 2.164148 | 6767.421 | 3.699 | **1830** |

### 1.7.3 Q2: Using the `abundance.tsv` files generated by Kallisto and the information above, calculate the TPM for PCHAS_1402500 in each of our five samples.

| Sample | Per kilobase scaling factor | Reads per kilobase (RPK) | TPM |
|---|---|---|---|
| MT1 | 3.697 | 687.30 | **2342** |
| MT2 | 3.709 | 914.50 | **1354** |
| SBP1 | 3.699 | 3947.87 | **2295** |
| SBP2 | 3.696 | 4681.81 | **3275** |
| SBP3 | 3.699 | 3959.78 | **2536** |

### 1.7.4 Q3: Do these match the TPM values from Kallisto?

**Yes.**

Well, almost. They may be a couple out because we rounded up to make the calculations easier.

### 1.7.5 Q4: Do you think PCHAS_1402500 is differentially expressed between the MT and SBP samples?

**Probably not.**

If we were to look at only the counts and RPKM values then it appears there is an 8 fold difference between the MT and SBP samples. However, when we look at the TPM values, they are much closer and so differential expression is less likely.