assignment1_final

September 28, 2021

Responses: Michael Fernandez

What is your bowtie2 commandline?

bowtie2 -x Index/bowtie2/grch38_transcriptome -U Data/Series1_NHBE_SARS-CoV-2_1/SRR11412227.fastq,Data/Series1_NHBE_SARS-CoV-2_1/SRR11412228.fastq,Data/Series1_NHBE_SARS-CoV-2_1/SRR11412229.fastq,Data/Series1_NHBE_SARS-CoV-2_1/SRR11412230.fastq -S Alignments/bowtie2/sc2_1.transcriptome.sam

How many reads aligned uniquely? 1840075 (12.24%) aligned exactly 1 time

How many reads aligned at all? 13844522

How many lines are the sc2_1.idxstats file? 188754

What does each line represent? each line is a reference sequence name, sequence length, number of mapped reads and number of unmapped reads.

How many genes are in the human genome approximately? 30,000

Kallisto command: kallisto quant -i Index/homo_sapiens/transcriptome.idx -o Alignments/kallisto –single -l 200 -s 20 Data/Series1_NHBE_SARS-CoV-2_1/* &

What does the difference between those two numbers tell us, and how is that relevant to the various alignment and quantification tools we discussed? The difference shows us that out of all the reads we get, we do not get signal from all the genes. We are not mapping to all the genes in the human genome. Not all cells transcribe all the possible genes, and therefore if we sequence all the reads we won't get reads from all the genes.

How many lines are in the kallisto output file, abundance.tsv? 188753

What do the -l 200 and -s 20 indicate? I guessed 200 as the number, but what if I was wrong and it should be more like 400? -l represents the average fragment length, the -s represents the estimated standard deviation of fragment length. If it was suppose to be 400 compared to 200, then at 200 reads would be mapping to more transcripts then what its supposed to be compared to at 400.

Now that you've made your plot, what does this show you about the similarity of estimates? For alot of the genes, along the diagonal, the kallisto and bowtie2 alignments are identical.

From what we've discussed in class, what might contribute to the differences? Kallisto is able to identify splice sites and allows for reads to be assigned to multiple transcripts compared to bowtie2 that attempts to identify the one transcript the read belongs to.

The difference is not due to pseudoalignment vs alignment. Rather, it has to do with what else kallisto does beyond that. What other tools that we discussed might be a good second step after

bowtie alignment, to get closer to the kallisto output?

Have a correction or next step to map those reads that span an intron-exon junction. This can be done using the tophat work flow that includes collecting initially unmapped reads and building a seed talbe index from those reads. While at the same time use mapped reads from bowtie to assemble consensus of covered regions, then identify possible splices sites. Using these possible splices sites and the index of unmapped reads then map initially unmapped reads to possible splices via seed-and-extend methodology.

Our goal is to make a scatterplot comparing the per-transcript counts from bowtie2 and kallisto. First, import pandas, a popular python library for data analysis, and import matplotlib for plotting.

In this notebook, if you see a string saying 'fill in ...', you're supposed to enter the appropriate value there.

[1]:
```
import pandas
```

[2]:
```
import matplotlib.pyplot as plt
```

We'll read in the data with pandas, creating two dataframes. The bowtie2 output file doesn't have column labels, so you'll have to add them. I looked at the samtools idxstats documentation to find what they are: "transcript", "length", "count", and "unmapped".

[9]:
```
bowtie = pandas.read_table( "~/Alignments/bowtie2/sc2_1.idxstats" ,␣
 ↪header=None, names=('transcript','lenght','count','unmapped'))
bowtie
```

[9]:

	transcript	lenght	count	unmapped
0	ENST00000631435.1	12	0	0
1	ENST00000434970.2	9	0	0
2	ENST00000448914.1	13	0	0
3	ENST00000415118.1	8	0	0
4	ENST00000632684.1	12	0	0
…	…	…	…	…
188749	ENST00000639660.1	284	0	0
188750	ENST00000643577.1	105	0	0
188751	ENST00000646356.1	900	0	0
188752	ENST00000645792.1	930	1	0
188753	*	0	0	1187574

[188754 rows x 4 columns]

That last line is the count of all the reads that don't align to a transcript. We don't want that, so save everything but that line.

[17]:
```
bowtie = bowtie[:188753]
bowtie
```

[17]:

	transcript	lenght	count	unmapped
0	ENST00000631435.1	12	0	0
1	ENST00000434970.2	9	0	0
2	ENST00000448914.1	13	0	0
3	ENST00000415118.1	8	0	0
4	ENST00000632684.1	12	0	0
…	…	…	…	
188748	ENST00000639790.1	1370	0	0
188749	ENST00000639660.1	284	0	0
188750	ENST00000643577.1	105	0	0
188751	ENST00000646356.1	900	0	0
188752	ENST00000645792.1	930	1	0

[188753 rows x 4 columns]

And now, read in the kallisto output. This one does have a line of column headers, hooray.

[23]:
```
kallisto = pandas.read_table( "~/Alignments/kallisto/abundance.tsv", header=0)
kallisto
```

[23]:

	target_id	length	eff_length	est_counts	tpm
0	ENST00000631435.1	12	2.60424	0.0	0.000000
1	ENST00000434970.2	9	2.52391	0.0	0.000000
2	ENST00000448914.1	13	2.62202	0.0	0.000000
3	ENST00000415118.1	8	2.48058	0.0	0.000000
4	ENST00000632684.1	12	2.60424	0.0	0.000000
…	…	…	…	…	
188748	ENST00000639790.1	1370	1171.00000	0.0	0.000000
188749	ENST00000639660.1	284	85.00110	0.0	0.000000
188750	ENST00000643577.1	105	4.43819	0.0	0.000000
188751	ENST00000646356.1	900	701.00000	0.0	0.000000
188752	ENST00000645792.1	930	731.00000	1.0	0.107765

[188753 rows x 5 columns]

As you can see, the first columns of each file look similar. Maybe they're identical! That would make it really easy to compare the counts from the two files. First, we'll try this with '=='.

[24]:
```
bowtie["transcript"] == kallisto["target_id"]
```

[24]:
```
0         True
1         True
2         True
3         True
4         True
          …
188748    True
188749    True
```

```
188750    True
188751    True
188752    True
Length: 188753, dtype: bool
```

Does that tell you for sure? Now try it with the pandas function equals(), which works on a whole dataframe or a column of a dataframe (called a 'series' in pandas).

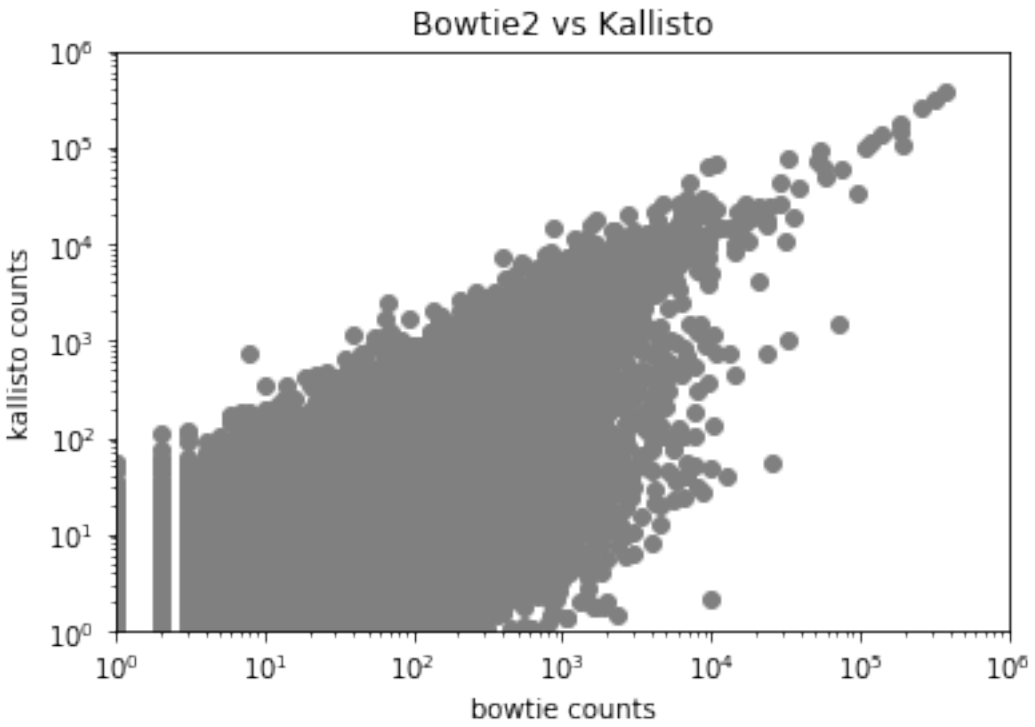
[25]:
```
bowtie["transcript"].equals(kallisto["target_id"])
```

[25]:
```
True
```

Hooray! Now we'll plot them against each other. Plot it in log scale, because that's a more natural scale for count data like this. Remember to label the axes and the plot so I know what I'm looking at!

[26]:
```
plt.plot( bowtie["count"], kallisto["est_counts"], 'o', color='grey')
plt.xscale('log')
plt.xlim(1,1e6)
plt.yscale('log')
plt.ylim(1,1e6)
plt.xlabel('Bowtie counts')
plt.ylabel('Kallisto counts')
plt.title('Bowtie2 vs Kallisto')
```

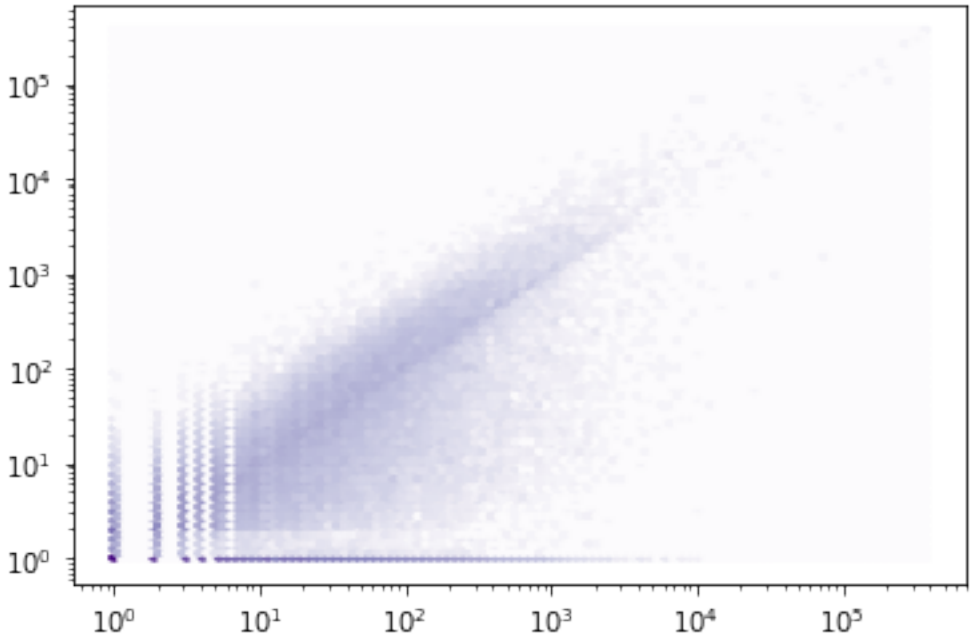
[26]:
```
Text(0.5, 1.0, 'Bowtie2 vs Kallisto')
```



You can tell a lot of points are plotted over each other. So next, let's try to look at the density of points. Hexbin is a quick way to do this. It calculates the number of points within each little hexagon of a grid. Here I'm using a quick and sloppy trick of adding 1 to all the values so it doesn't complain about taking log(0).

[27]:
```
plt.hexbin( bowtie["count"]+1, kallisto["est_counts"]+1, bins='log',␣
 ↪xscale='log', yscale='log', cmap='Purples')
# and give it x and y labels and a title!
```

[27]:
```
<matplotlib.collections.PolyCollection at 0x7ffa6c7ef790>
```



There you have it! Save the completed notebook and submit it along with your answers to the assignment. The assignment has a few more questions you should answer based on this graph.

[ ]:
```

```