Next Generation Sequencing Technologies: Data Analysis and Applications

Data Normalization Methods

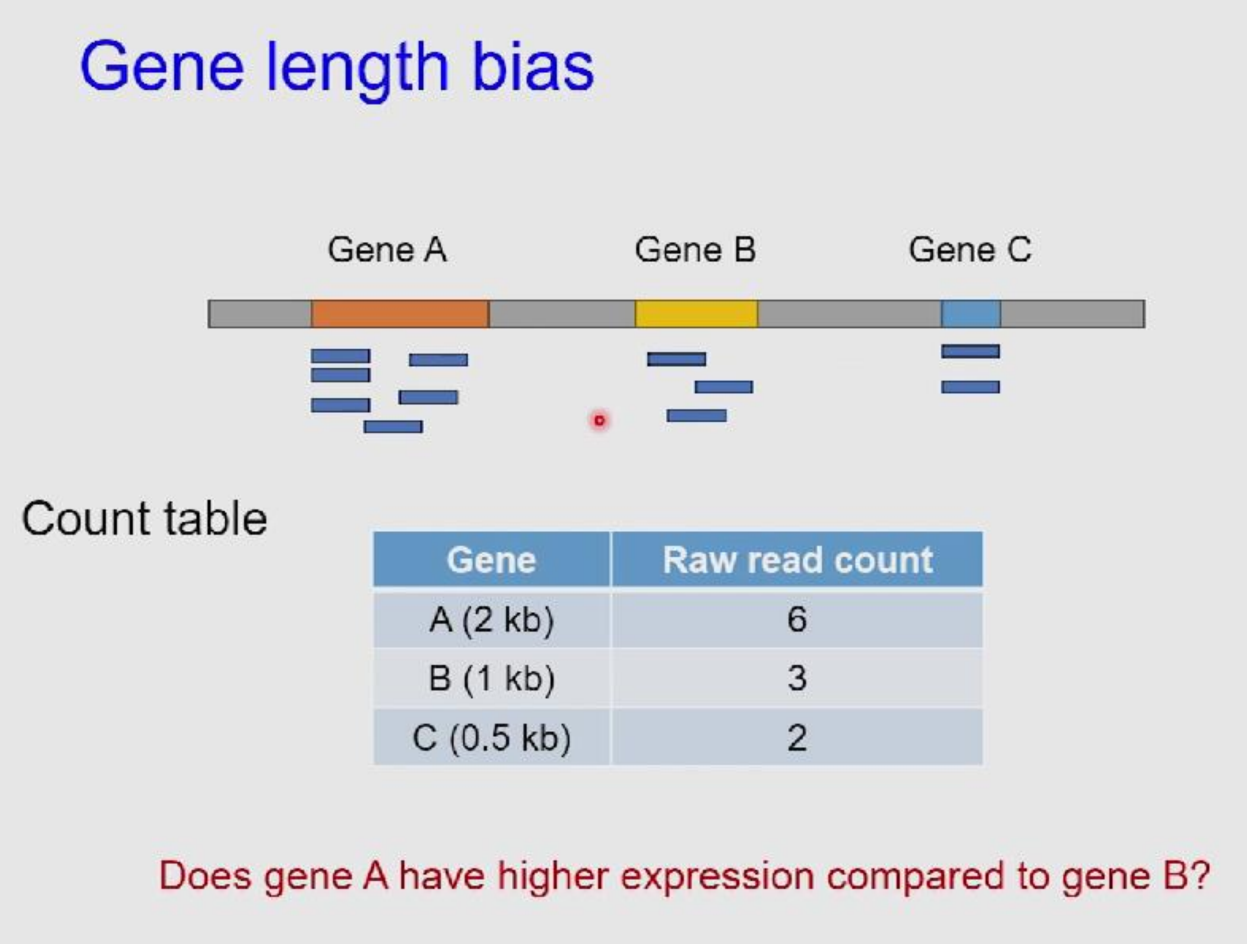
Dr. Riddhiman Dhar, Department of Biotechnology
Indian Institute of Technology, Kharagpur

Good day, everyone. Welcome to the course on Next Generation Sequencing Technologies, Data Analysis, and Applications. In the last class, we started a discussion on biases in RNA sequencing data. We talked about different types of biases, and we also discussed some of the methods and tools to correct for those biases. And then we started talking about normalization. So, data normalization corrects for any technical variation that is present in the data.

Of course, it cannot completely eliminate technical variation, but the goal is to minimize the technical variation so that we can study the biological variation. In this class, we will continue our discussion on data normalization methods. We will introduce more and more sophisticated methods for data normalization. In this class and the next, we will talk about different types of normalization.

For example, I have introduced, we will talk about sample normalization, then we will talk about sample normalization when we have multiple samples within a data set. And then, finally, we will talk about different types of data sets, combining different data sets, and then doing something called batch correction. So, today's agenda will be: first, we will talk about normalization for read length and sequencing depth. And then second, we will talk about between-sample normalization, or at least some of the techniques for between-sample normalization. Here are the keywords that will come up: RPKM, FPKM, and TPM.

So, let us begin with the same diagram that we used in the last class. So, here we have these three genes; you can see genes A, B, and C, and you have reads mapping to these genes, right? So, these reads are shown in blue, and they are mapping to this, which we can now convert to the count. So, we have this read count; a simple read count will give A; for A, we have 6 reads; for B, we have 3 reads and for C, we have 2 reads. And we also talked about gene length bias, right? So the genes are of different lengths, so I have also mentioned the size of this gene B, right? So, one of the questions that we talked about is: does gene A have a higher expression compared to gene B in this sample? Can we answer that question right from the data that we have? So, the answer is no, right? We need a lot more data and a lot more methods to actually analyze this properly so that we can get an answer to this question, ok?



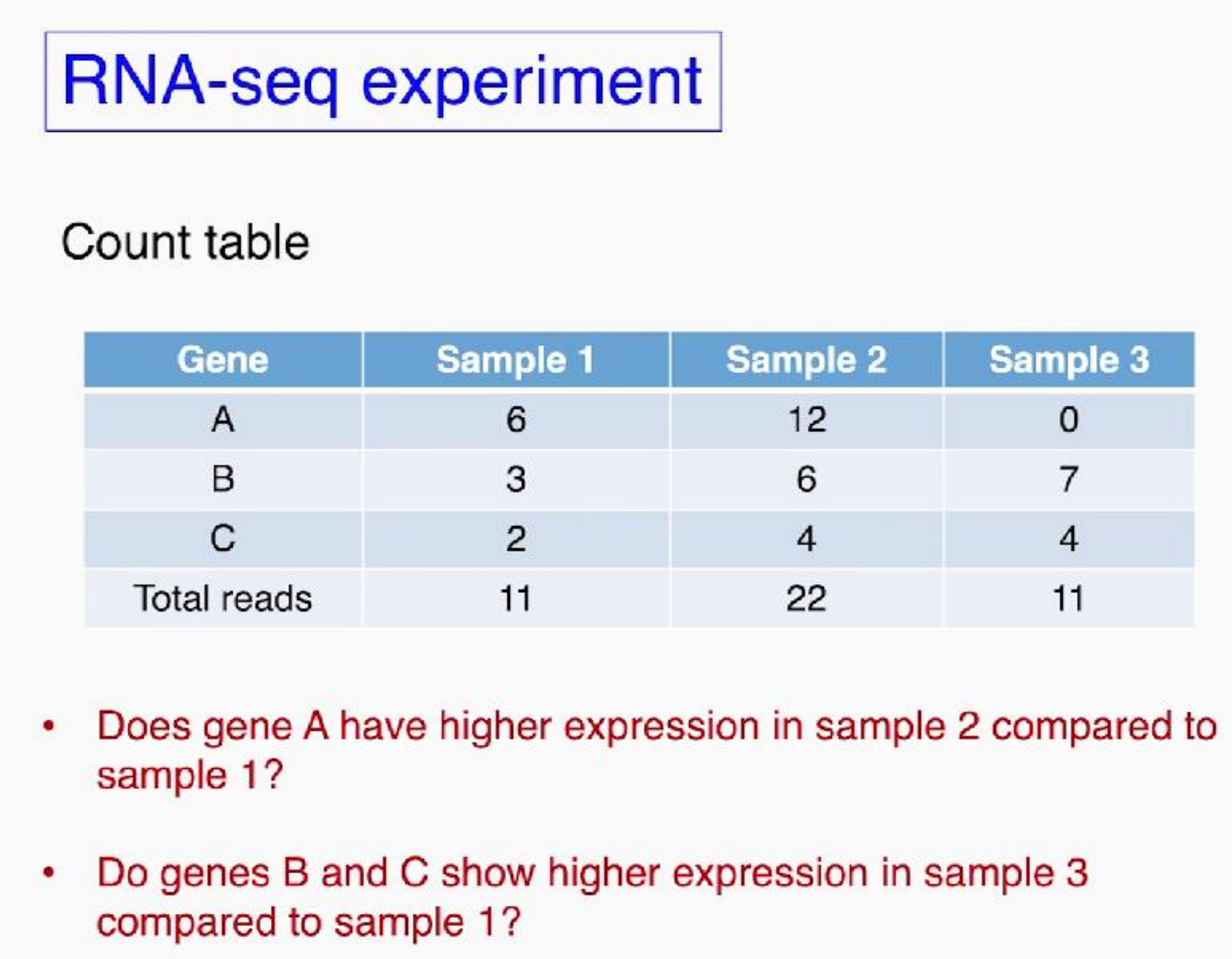
Gene length bias

Count table

Gene	Raw read count
A (2 kb)	6
B (1 kb)	3
C (0.5 kb)	2

Does gene A have higher expression compared to gene B?

We also introduced another example, where we are comparing among different samples. So, here we have sample 1, sample 2, and sample 3. We have these 3 genes again, and we have corresponding read counts, and we also have the total reads, right? So, we also talked about, right, we have this gene length bias, and we also have to look for sequencing depth, right? So, each sample will have its own sequencing depth that will be different from other samples, right? This is common across all RNA-seq experiments because you cannot always ensure that you will get exactly the same number of reads for each sample.



RNA-seq experiment

Count table

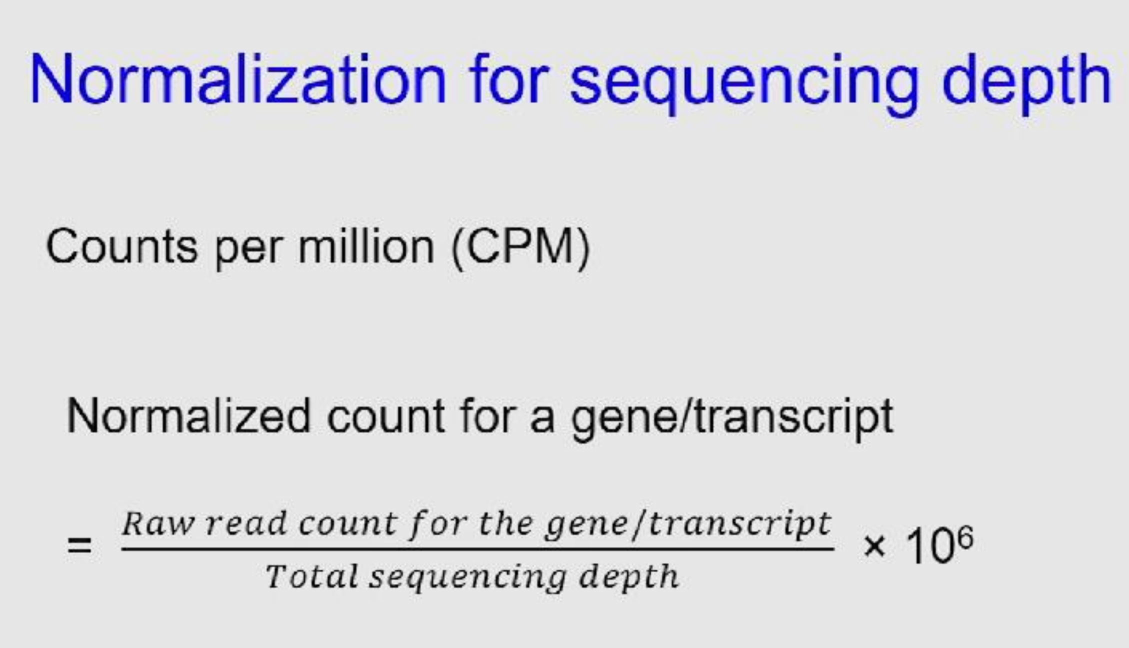
Gene	Sample 1	Sample 2	Sample 3
A	6	12	0
B	3	6	7
C	2	4	4
Total reads	11	22	11

- Does gene A have higher expression in sample 2 compared to sample 1?
- Do genes B and C show higher expression in sample 3 compared to sample 1?

So, two questions that I introduced in the last class: right, does gene A have higher expression in sample 2 compared to sample 1? This is again dependent on a lot of factors, which we will actually analyze today, and we will see if we can answer this question in this class. And then we have the second question: do genes B and C show lower expression in sample 3 compared to sample 1? So, again, we will see if we can answer that question in this class, or maybe in the next class we will have more sophisticated tools that we can utilize to answer this question. So, the first question we will answer is: how can we normalize raw count data for gene length and sequencing depth?

So, we introduce this gene length bias because genes are of different lengths, and if you have a long gene, it is likely to have more fragments generated in the library, and hence you will get more reads for that gene. So, this is what we call gene length bias, and then we also have sample-specific sequencing depth variation, right? Each sample will get a different total number of reads, and that can again influence the outcome. So, if you have more reads in a sample, it is very likely that all the genes will also get more reads by chance, right? So, again, we would have to do something about that, right?

This variation in sequencing depth, so that we can actually do a fair comparison, is okay. So, this is the idea behind normalization, and the first question is: can we normalize for gene length and sequencing depth? So, let us start with a very simple normalization, which actually normalizes for sequencing depth only; it is called counts per million, or, in short, CPM, and then this is the very simple formula. So, we have a normalized count for a gene or transcript. So, here, the normalized count is calculated from the raw count using this formula.



Normalization for sequencing depth

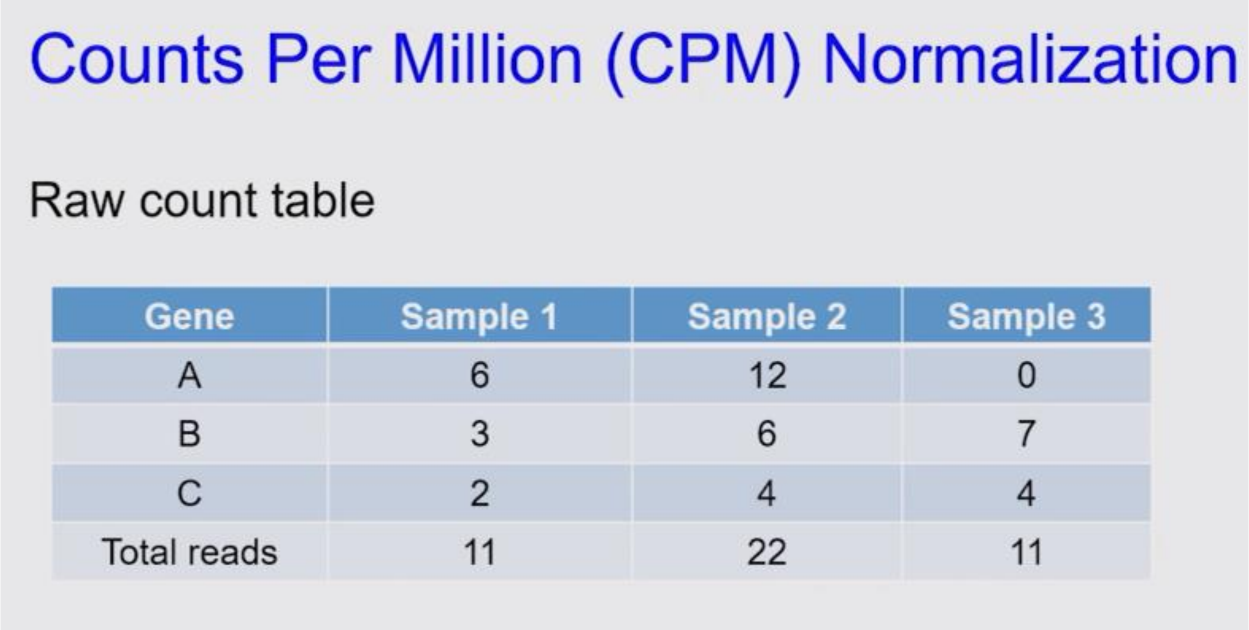
Counts per million (CPM)

Normalized count for a gene/transcript

$$= \frac{Raw\ read\ count\ for\ the\ gene/transcript}{Total\ sequencing\ depth} \times 10^6$$

So, you have the raw read count for the gene or the transcript divided by the total sequencing depth by 10 to the power of 6, ok? So, this is counts per million, right? So, we have to multiply with 6, 10 to the power of 6, ok? So, let us do this operation here. So, we will go slowly so that you understand all these operations, and this will actually be helpful as we go into more complex operations, ok?

So, we will start with the same example, right? We have these three samples, and we have these read counts for these genes A, B, and C. So, the first step is to divide by the total sequencing depth. So, the total is there, right? So, we divide everything.

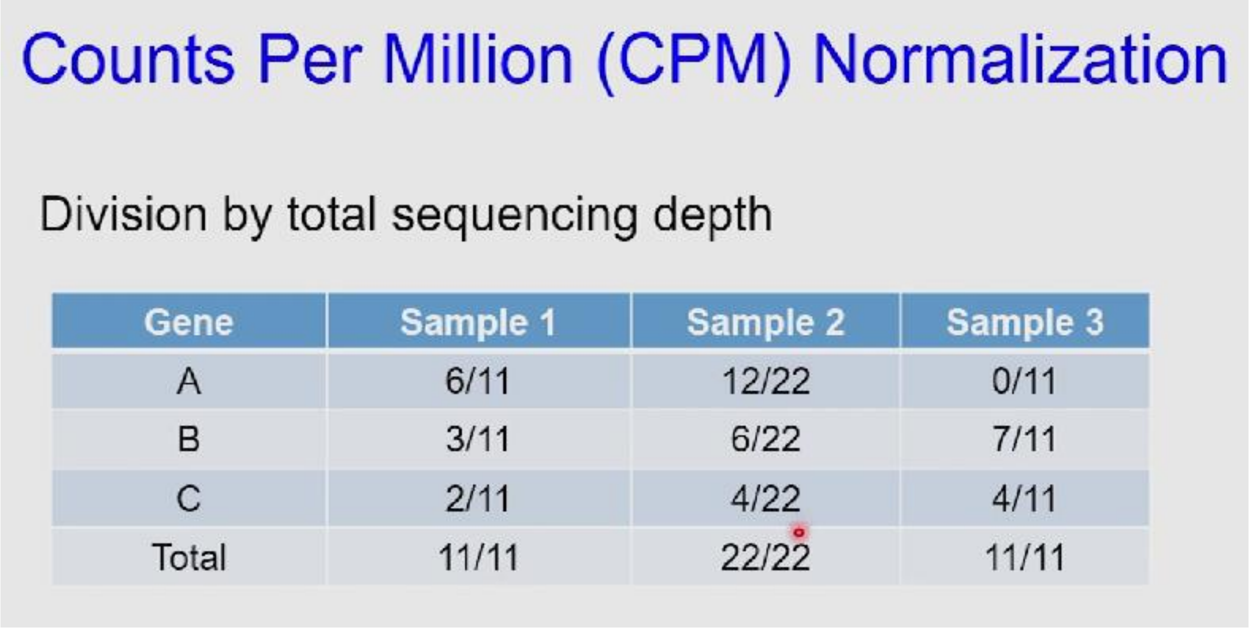


Counts Per Million (CPM) Normalization

Raw count table

Gene	Sample 1	Sample 2	Sample 3
A	6	12	0
B	3	6	7
C	2	4	4
Total reads	11	22	11

So, for sample 1, the total is 11. So, we divide this column by 11. For sample 2, the total is 22. So, we divide by 22. For sample 3, it is 11 again, and we divide by 11.

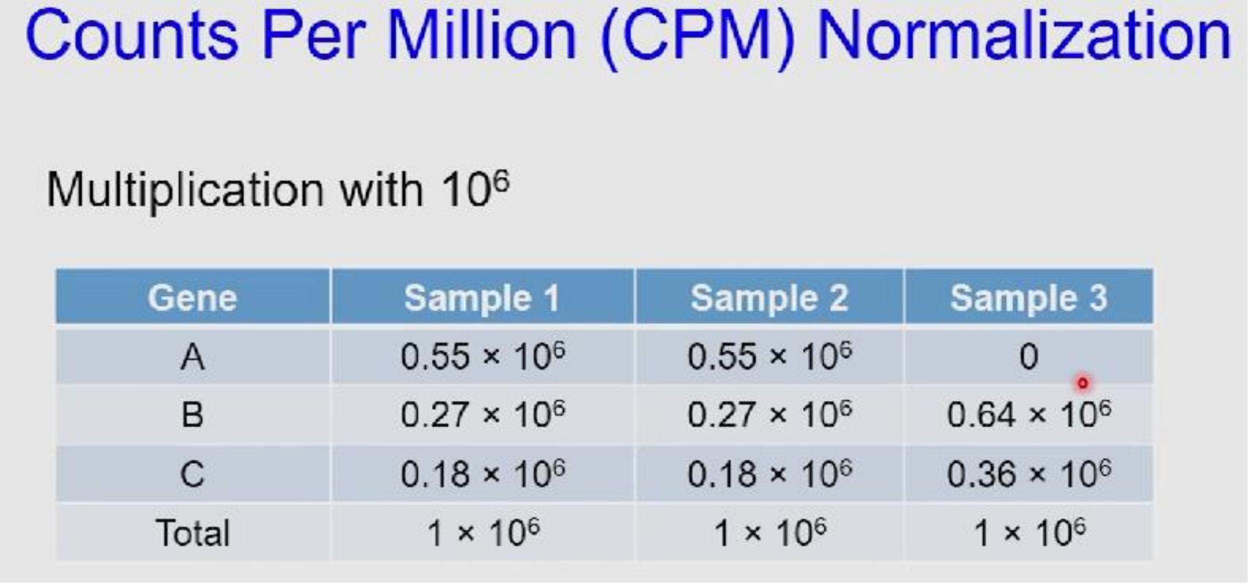


Counts Per Million (CPM) Normalization

Division by total sequencing depth

Gene	Sample 1	Sample 2	Sample 3
A	6/11	12/22	0/11
B	3/11	6/22	7/11
C	2/11	4/22	4/11
Total	11/11	22/22	11/11

So, this is just an example with a very small number of genes, ok? And we do the same thing for real data sets where you have thousands of genes, ok? So, then once we have divided, right, we just multiply with 10 to the power of 6, ok, and you get this kind of answer, ok. And what you notice here is the total, which comes to 1 into 10 to the power of 6 for each sample. So, that is the idea, right?



Counts Per Million (CPM) Normalization

Multiplication with 10^6

Gene	Sample 1	Sample 2	Sample 3
A	0.55×10^6	0.55×10^6	0
B	0.27×10^6	0.27×10^6	0.64×10^6
C	0.18×10^6	0.18×10^6	0.36×10^6
Total	1×10^6	1×10^6	1×10^6

That is what we want to do. We want to normalize the sequencing depth that was variable between these three samples. We have normalized for that, and we have reached the same total sequencing depth, ok? Now, one of the questions you might have is: why are we multiplying with 10 to the power 6? We could have just left it like this without this multiplication, right? The sum would be just 1, right?

It is just a convention, ok, because when you deal with the real data sets, we have millions of reads coming for each sample, right? So, if you look at the read data, the total read for a real data set will be in the order of millions. So, if some nowadays, it could be even more, right, in billions. Now, what will happen is if you divide the count for each gene by that big number. So, if you have let us say for gene A you have count maybe 1000, for gene B maybe you have count probably 100 or maybe less, right?

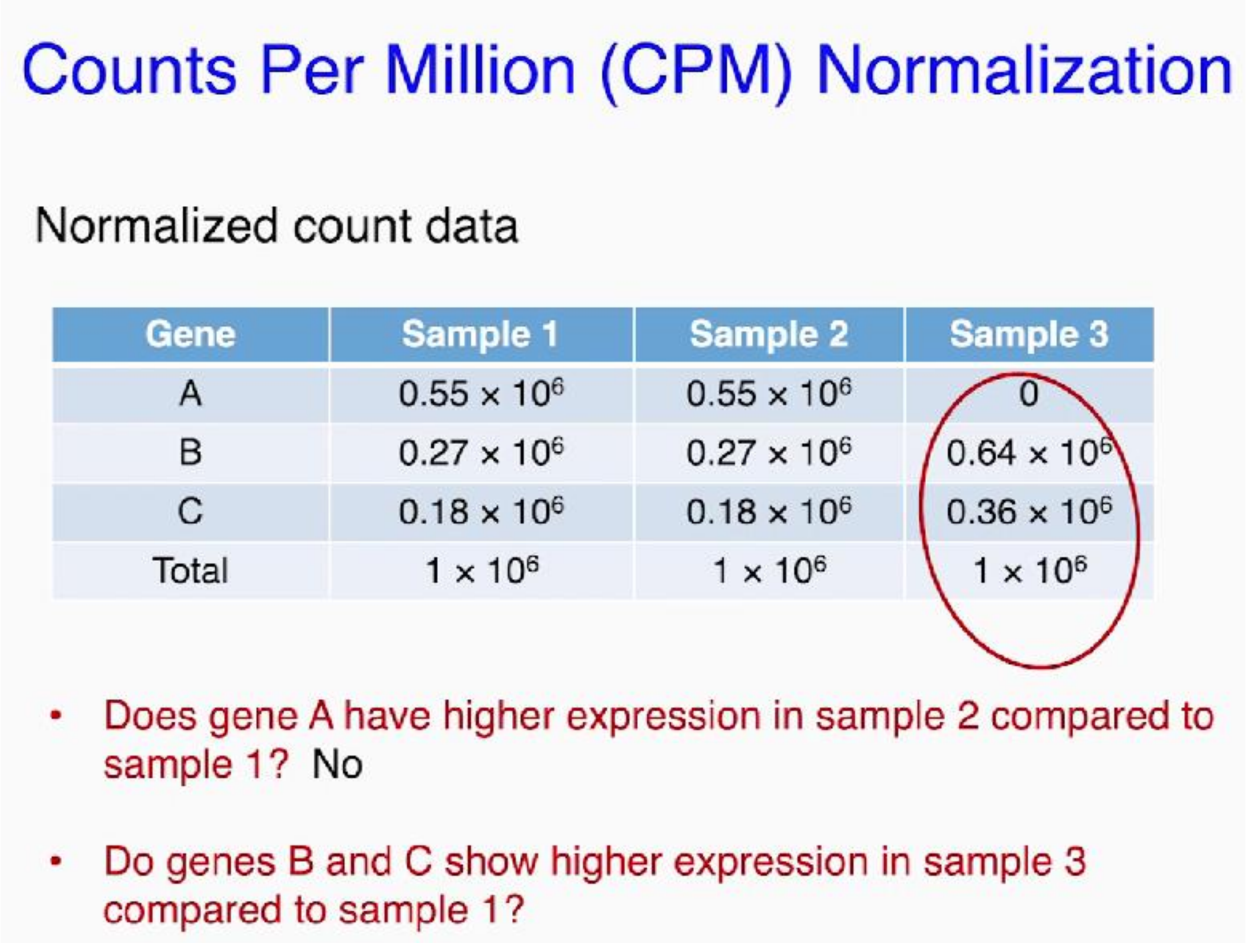
And that number you divide by, let us say, 20 million, ok? So, what will happen is that this normalized number that you will get just by this division will be very, very small. As you can imagine, if you divide 100 by 20 million, that would be a very small number. That would be a fraction, right? So, to get something that is easily readable and that we can handle better, . So, that is why we multiply with 10 to the power of 6, right?

So, we get something like a float or very close, at least some integer values are there, etcetera, right? So, we have something like 1, 2, or something like that, ok? Instead of very small fractions, which read like 0.000 or something, ok. So, that is the idea behind this multiplication, ok?

So, that is what we have answered, right? Why do we multiply with 10 to the power of 6? So, that we get numbers that are easily readable, right for us, ok? So, this is a very simple normalization. So, of course, it has a lot of drawbacks. So, one of the major drawbacks is that we are not doing any normalization for gene length.

We are only normalizing the total sequencing depth, ok? So, as you can see, we have the same total for all the samples, but we have not touched or worked with the gene length at all. So, that means, again, we cannot really compare the expression levels of genes even within a sample. This is one. The second is that we have not addressed RNA composition bias, ok?

So, what is this RNA composition bias? We briefly mentioned it in the last class, ok? So, if you look at sample 3, right gene A, the read count is 0, ok? So, let us imagine that the expression of gene A has dropped, and it is so low that we cannot detect it by sequencing. So, it is set to 0. What it means is that the total number of reads that are present and available for sample 3 would be distributed between gene B and gene C.



Counts Per Million (CPM) Normalization

Normalized count data

Gene	Sample 1	Sample 2	Sample 3
A	0.55×10^6	0.55×10^6	0
B	0.27×10^6	0.27×10^6	0.64×10^6
C	0.18×10^6	0.18×10^6	0.36×10^6
Total	1×10^6	1×10^6	1×10^6

- Does gene A have higher expression in sample 2 compared to sample 1? No
- Do genes B and C show higher expression in sample 3 compared to sample 1?

So, now, what will appear is that gene B and gene C are showing higher expression in sample 3 compared to, for example sample 1. So, this is the RNA composition bias. The read count for gene B and gene C is affected because the expression of gene A has changed. So, what you probably can now guess is that the read count for a single gene is not just dependent on its expression but is dependent on the expression of the whole transcriptome, ok? So, that is how the whole thing works, ok?

So, this composition bias has not been addressed, right? So, if you just look at this CPM data, this normalized count data, what you see is that sample 3 has a higher count for gene B and gene C compared to sample 1 or sample 2, right? And you might say, OK, these genes B and C are showing higher expression in sample 3 compared to sample 1 and sample 2, ok. So, if you conclude that that will be erroneous, right, because we have not addressed this RNA composition bias, right, that that is occurring just because gene A's expression has gone down, ok. Maybe B and C have not changed their expression, but because of the change in expression of gene A, the distribution of reads is such that gene B and gene C appear to be highly expressed.

So, this is something that this method cannot address, which means we cannot really compare gene expression across samples, ok? So, this is a method that we will not use at all because there are a lot of problems and drawbacks, but I introduced you to this method because it will be helpful, right, for the further discussion in this class. So, let us move on to the next part, right? So, again, because we have not addressed this RNA composition bias, we cannot answer the second question in these two questions that I said, right? So, the first question was: does gene A have higher expression in sample 2 compared to sample 1? Then the answer is no, right?

So, as you can see, the numbers are exactly identical because we have taken that sequencing depth into account. Now, we can say they are not different, right? If you look at samples 2 and 1, you see, right, for the expression of gene A, the read count and normalized count data are exactly identical, ok? So, the answer is no, but we cannot answer the second question, right, because we have not addressed the RNA composition bias. So, genes P and C show lower expression in sample 3 compared to sample 1.

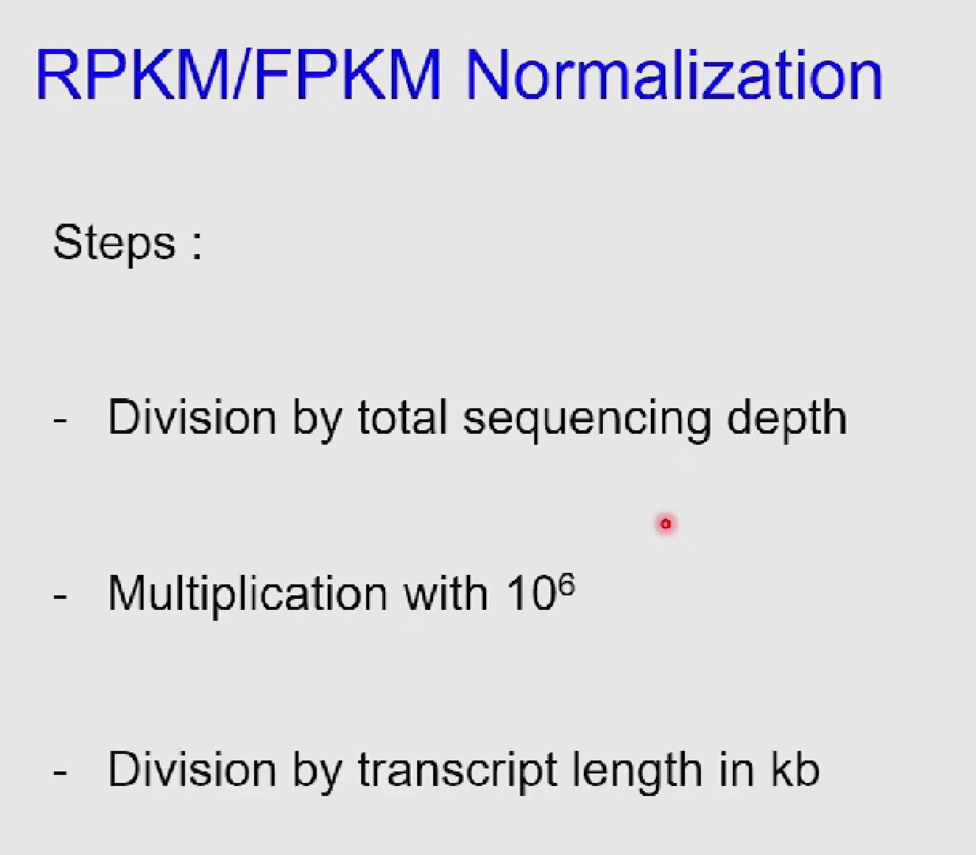
We cannot answer that question because we have not addressed this RNA composition bias, alright? So, let us now move on to the next normalization, ok? So, this will address gene length normalization as well as normalization for sequencing depth, ok? So, the first one we will talk about is called RPKM or FPKM normalization, ok? I will introduce these terms in a moment and explain what RPKM and FPKM mean, and then we will also talk about TPM normalization.

So, let us start with RPKM and FPKM normalization, ok? So, RPKM stands for reads per kB per million map reads, and FPKM stands for fragments per kB per million map reads, ok? So, in RPKM and FPKM, the methods are exactly identical. The only difference is that instead of counting reads, we count them as fragments. So, why do you count them as fragments? Because if you are dealing with paired-end data, it does not make any sense, right?

The fragments make more sense there, ok? So, if you are dealing with single-end data, you can use

RPKM, but if you are dealing with paired-end data, you would have to use the FPKM measure. So, the terminologies are different, but the normalization procedure is exactly the same for both of them. So, let us now look into the normalizations, ok? So, here are the steps for this method. So, the first step is to divide by the total sequencing length and depth.

So, we have this depth calculation. So, it is actually very similar to what we discussed just before this, right? So, for the CPM normalization method, we follow the same process, right? We divide by the total sequencing depth, ok? Then we multiply with 10 to the power of 6 as before, and you know the reason now. Then we actually normalize for gene length, and to do that, we divide by transcript length in KB.

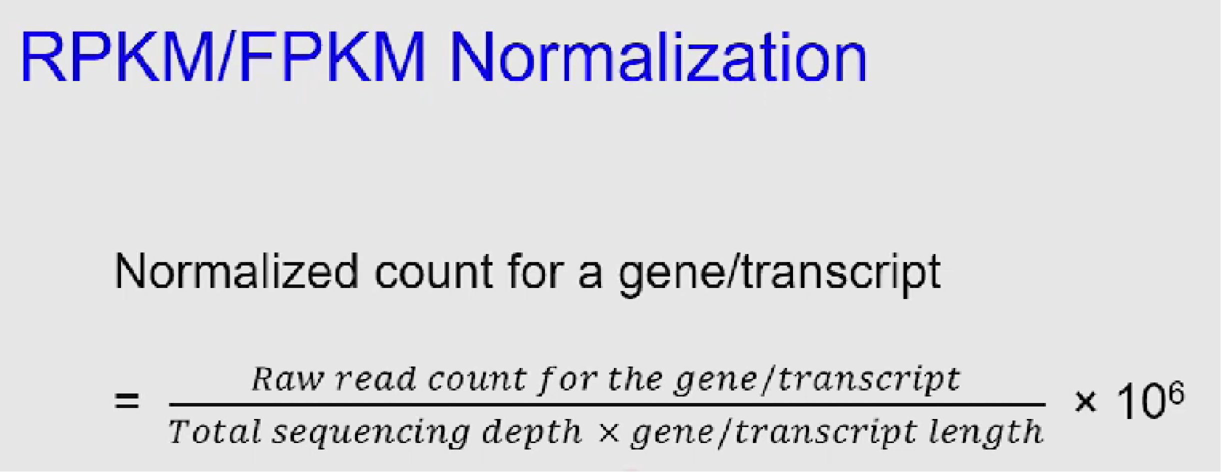


RPKM/FPKM Normalization

Steps :

- Division by total sequencing depth
- Multiplication with 10^6
- Division by transcript length in kb

So, this is gene length or transcript length in KB, ok? So, that is something you should note, ok? So, the formula is like this, right? So, we have to get the normalized count for a gene or transcript. So, by dividing the raw read count by total sequencing depth as well as by gene and transcript length, we multiply by 10 to the power of 6.

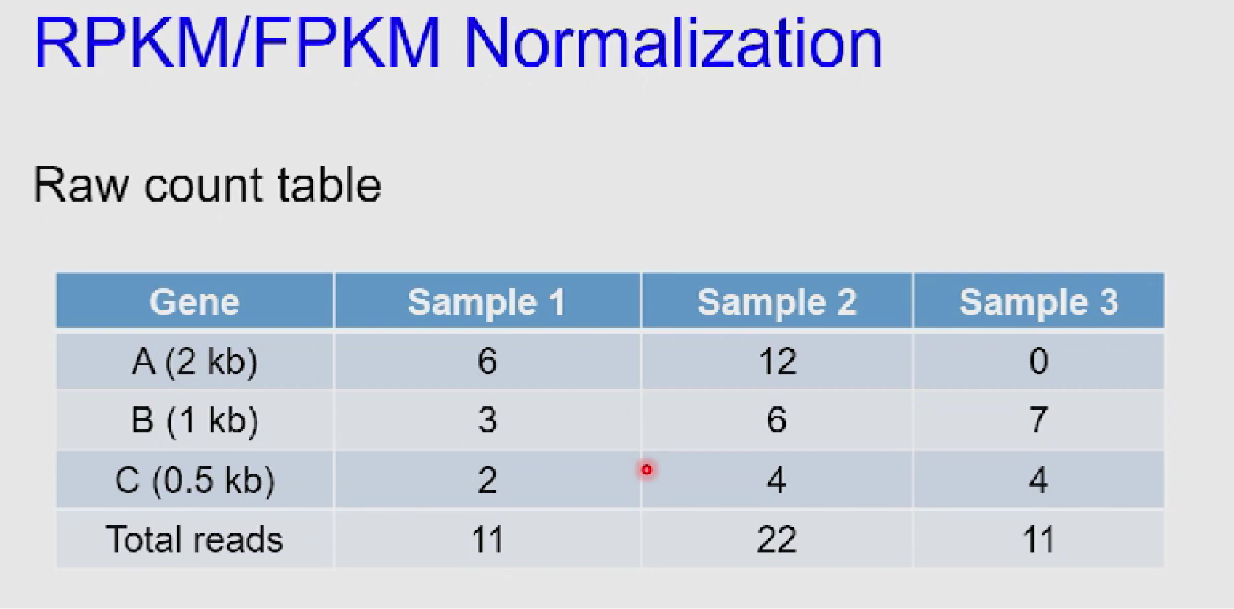


RPKM/FPKM Normalization

Normalized count for a gene/transcript

$$= \frac{Raw\ read\ count\ for\ the\ gene/transcript}{Total\ sequencing\ depth \times gene/transcript\ length} \times 10^6$$

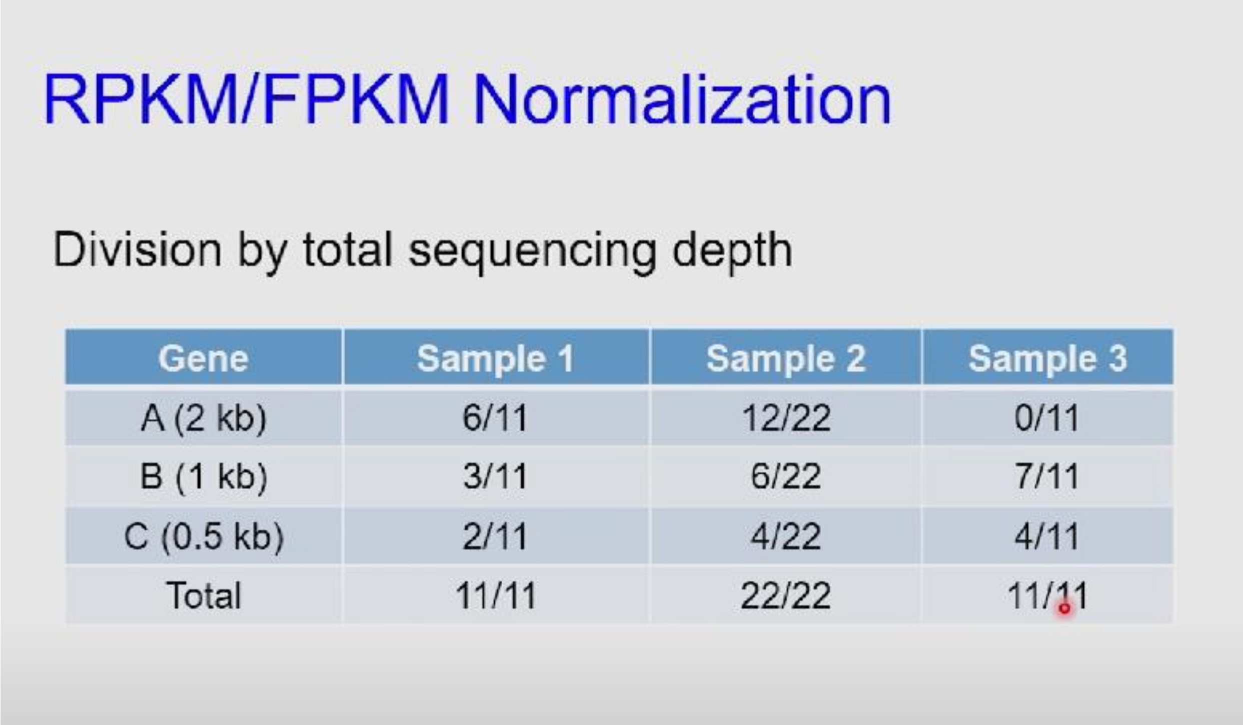
So, million again, we understand why we do that, ok? So, let us take this earlier example, and let us now do this RPKM and FPKM normalization, right? We follow this method, ok? So, first, we divide by the total sequencing depth. So, in this case, for sample 1, it is 11, for sample 2, it is 22, and for sample 3 it is 11, ok.



RPKM/FPKM Normalization

Raw count table

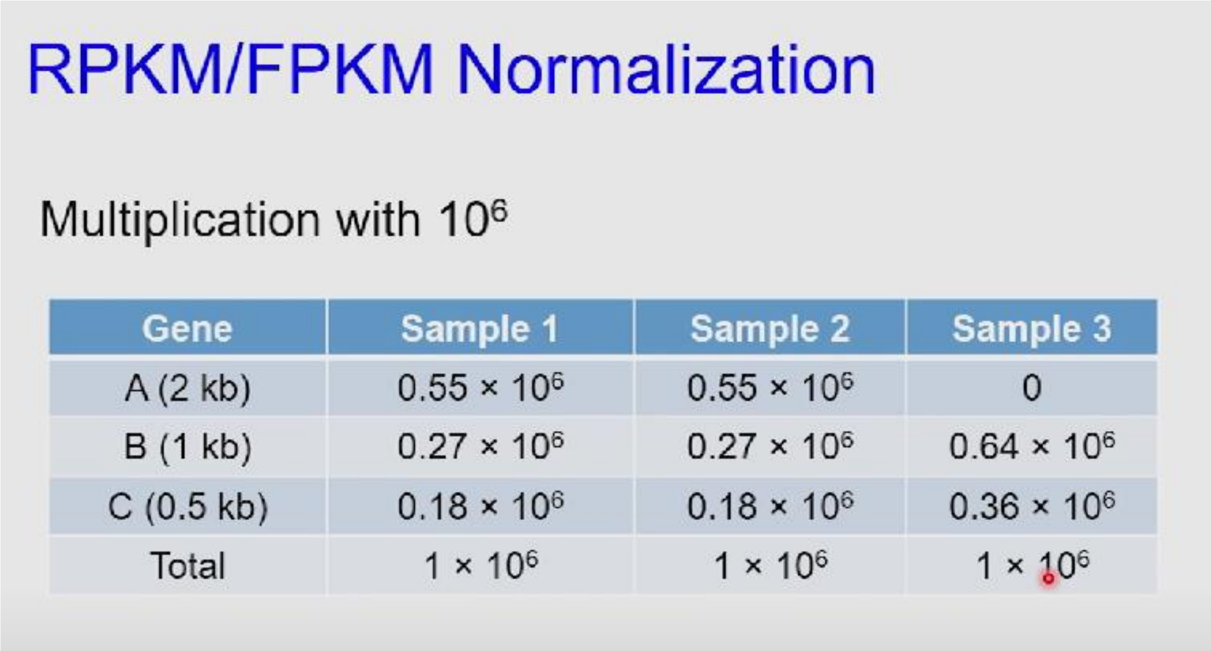
Gene	Sample 1	Sample 2	Sample 3
A (2 kb)	6	12	0
B (1 kb)	3	6	7
C (0.5 kb)	2	4	4
Total reads	11	22	11



RPKM/FPKM Normalization

Division by total sequencing depth

Gene	Sample 1	Sample 2	Sample 3
A (2 kb)	6/11	12/22	0/11
B (1 kb)	3/11	6/22	7/11
C (0.5 kb)	2/11	4/22	4/11
Total	11/11	22/22	11/11



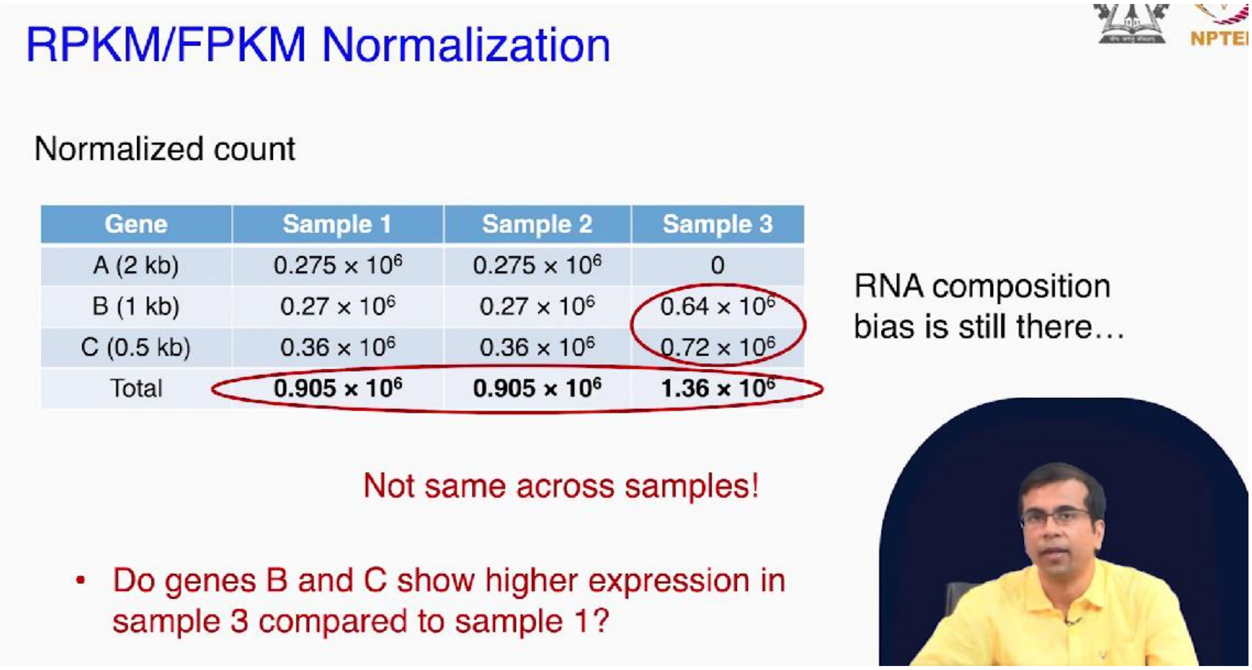
RPKM/FPKM Normalization

Multiplication with 10^6

Gene	Sample 1	Sample 2	Sample 3
A (2 kb)	0.55×10^6	0.55×10^6	0
B (1 kb)	0.27×10^6	0.27×10^6	0.64×10^6
C (0.5 kb)	0.18×10^6	0.18×10^6	0.36×10^6
Total	1×10^6	1×10^6	1×10^6

So, let us do that. Let us do the division as before, right? We have seen this before, ok, and then we multiply with 10 to the power of 6, ok, and these are the numbers that we get, ok. We did this before, right? Just before this method, ok? So, we get that this total is the same for all the samples, ok? Now, what we do is divide by gene transcript length in KB.

So, you can see this gene length here, right? So, for gene A, we have 2 KB; for gene B, the length is 1 KB; and for gene C, it is 0.5 KB. Okay, and that is what we do. We divide these counts that we got by this gene length in KB, ok? So, divide by 2, divide by 1, and divide by 0.5, ok, and we once do that, right? So, we can calculate what these values would be, and we get the total, ok, and what you would notice now is that the total is actually not the same for all samples, ok, and this has changed, ok, and this is a problem, right?



RPKM/FPKM Normalization

Normalized count

Gene	Sample 1	Sample 2	Sample 3
A (2 kb)	0.275 × 10^6	0.275 × 10^6	0
B (1 kb)	0.27 × 10^6	0.27 × 10^6	0.64 × 10^6
C (0.5 kb)	0.36 × 10^6	0.36 × 10^6	0.72 × 10^6
Total	**0.905 × 10^6**	**0.905 × 10^6**	**1.36 × 10^6**

RNA composition bias is still there…

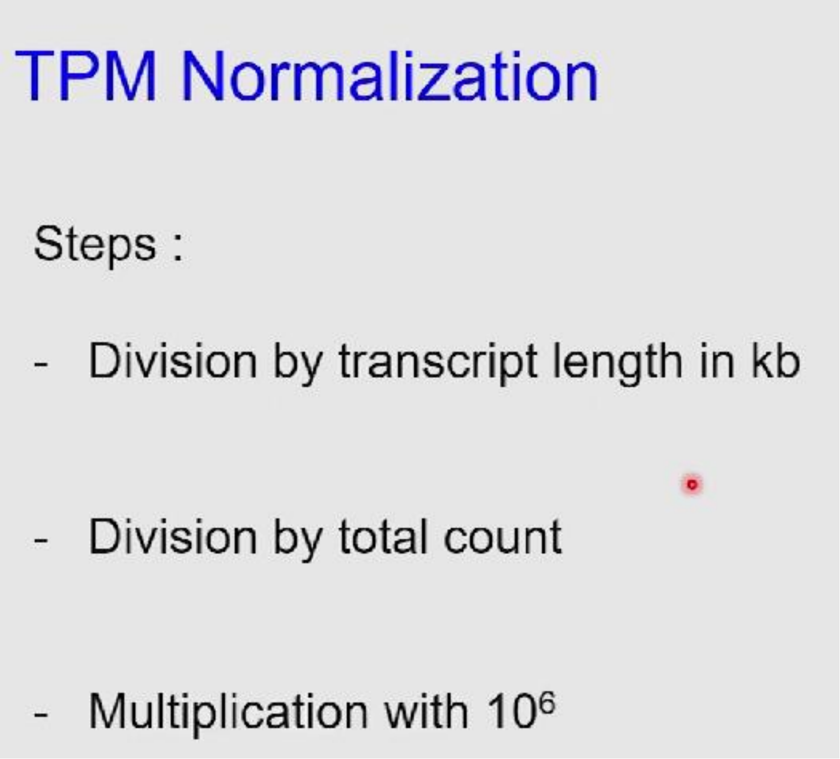
Not same across samples!

- Do genes B and C show higher expression in sample 3 compared to sample 1?

Again, we are comparing different numbers of total reads, or total sequencing depth. This is a transformed depth; of course, it has been transformed with all those divisions, etcetera, but again, if you look at the total, this is different, right? For example, sample 3 is different from sample 1 and sample 2, ok? So, this is a problem, ok because then we are not comparing across samples that have exactly the same sequencing depth that we got earlier when we were doing this CPM normalization.

So, this creates a problem because you cannot then compare across samples anymore, and also because the RNA composition bias is still there. We have not done anything about that yet, right? Because, again, if you see this number, it is actually higher. For sample 3, these counts that we got, normalized counts, are higher in sample 3 for gene B and gene C compared to sample 1 and sample 2, and you might erroneously say, OK, gene B and gene C are highly expressed in sample 3, but that is probably not the case again because gene A has changed its expression. So, what are the solutions then, ok? We cannot answer this question correctly yet, right? We can answer, of course, but that will be an erroneous answer, ok?

So, let us look at the other normalization, which is a slight deviation from the RPKM and FPKM normalizations, ok? So, this was proposed later after researchers found out the drawbacks of this RPKM, FPKM normalization, and TPM normalization. So, the TPM stands for transcripts per million, ok? So, very similar idea, but when you see the formula, you will see the formula is actually identical, but the steps are slightly different, ok? So, what are the steps? So, here, you first divide by the transcript length in KB, ok?



TPM Normalization

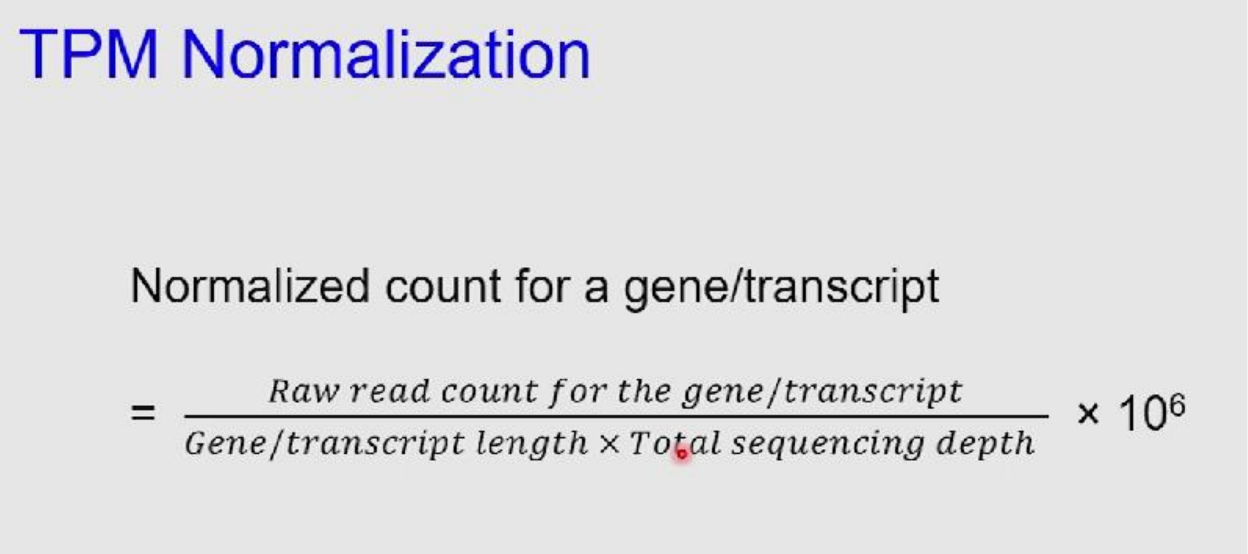
Steps :

- Division by transcript length in kb
- Division by total count
- Multiplication with 10^6

Instead of dividing by the total sequencing depth, you first divide by the transcript length in KB. Once you have done that, you divide by the total count. I am not saying total sequencing depth because the total will change because of this division by transcript length, right? So, it is a total count that will be obtained after this division, ok, and then multiplied by 10 to the power of 6, ok.

So, you see, just one step like this division by transcript length is the first step now instead of the last step in FPKM, RPKM normalization, ok?

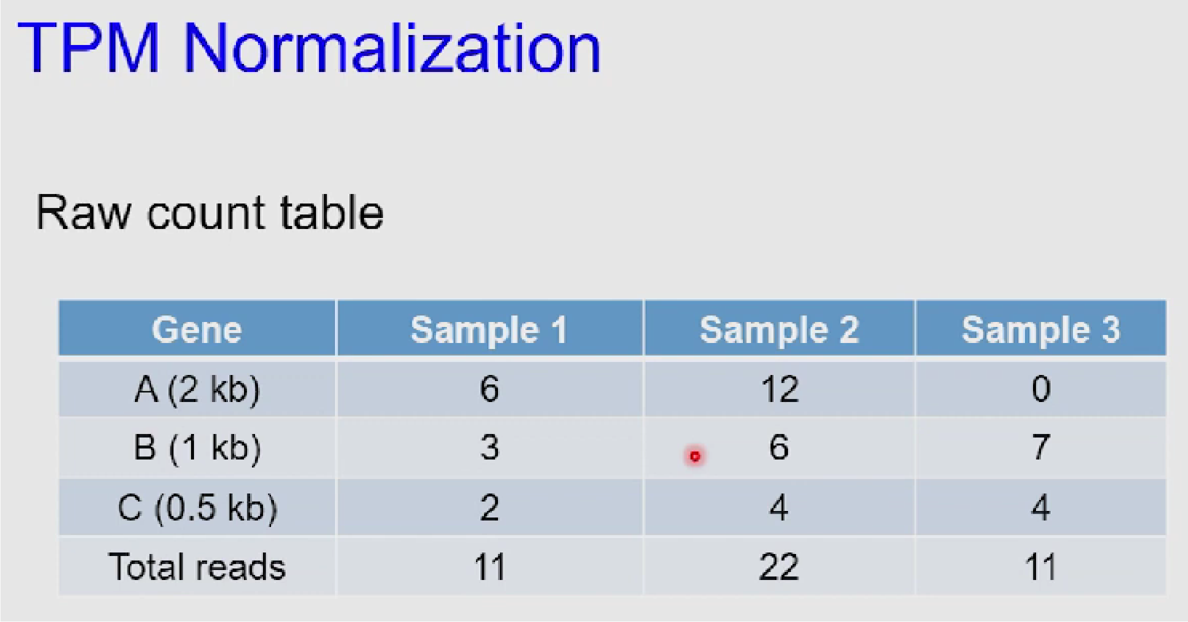
So, let us take this example again and see what we get when you do the TPM normalization, ok? So, the formula is actually identical. As you see, we get a normalized count for a read for a gene or a transcript by dividing the raw read count for the gene or transcript by the gene transcript length and also by the total sequencing depth, which is 10 to the power of 6. So, this is the formula that we also got earlier, but the steps are slightly different in order. So, let us take the example and let us do the analysis, do the TPM normalization ourselves with this very small data set, and we get an idea, ok? So, here again, we have three samples, and we have the genes A, B, and C, right, for which we have the raw read counts.



TPM Normalization

Normalized count for a gene/transcript

$$= \frac{Raw\ read\ count\ for\ the\ gene/transcript}{Gene/transcript\ length \times Total\ sequencing\ depth} \times 10^6$$

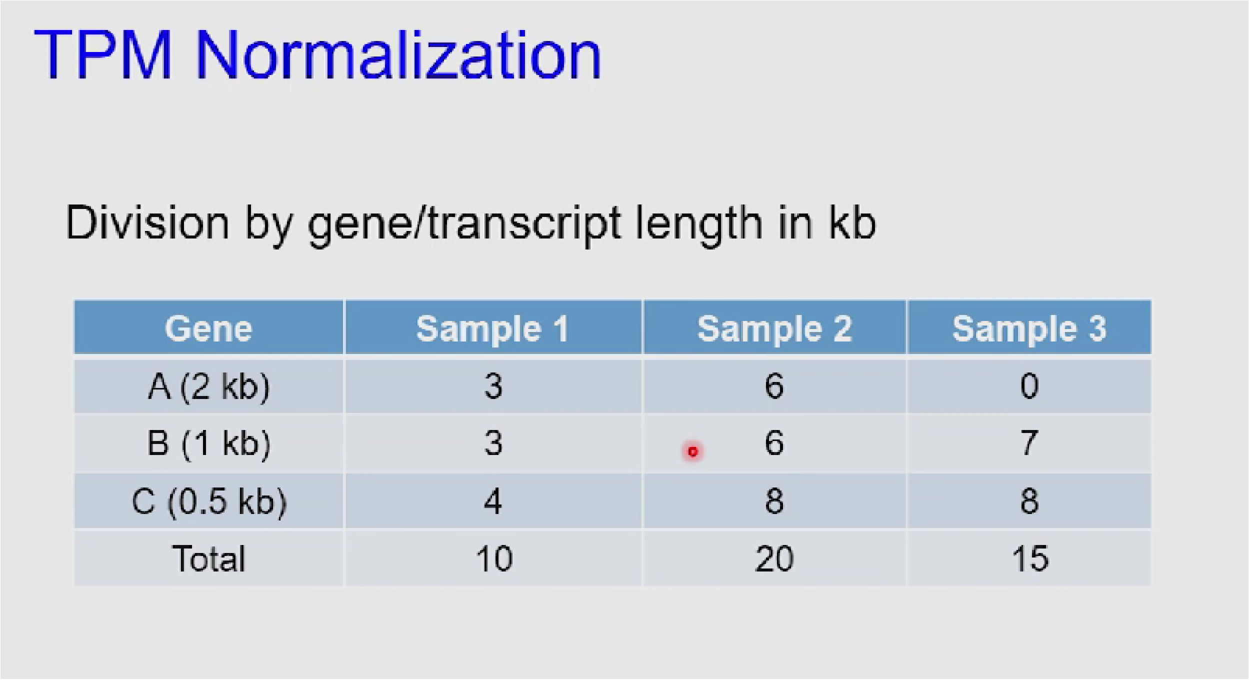


TPM Normalization

Raw count table

Gene	Sample 1	Sample 2	Sample 3
A (2 kb)	6	12	0
B (1 kb)	3	6	7
C (0.5 kb)	2	4	4
Total reads	11	22	11

So, remember now that the first step is division by the gene length, ok? So, we will divide by this in lengths of 2 kB, 1 kB, B, 0.5 kB, and we will get some numbers, ok? So, we do that, ok, and I have not written the total. We will get the total once we have done the division, ok, and you see, after the division, this is what we get, ok. This is not the total number of reads; it is actually some sort of transformed total, right?

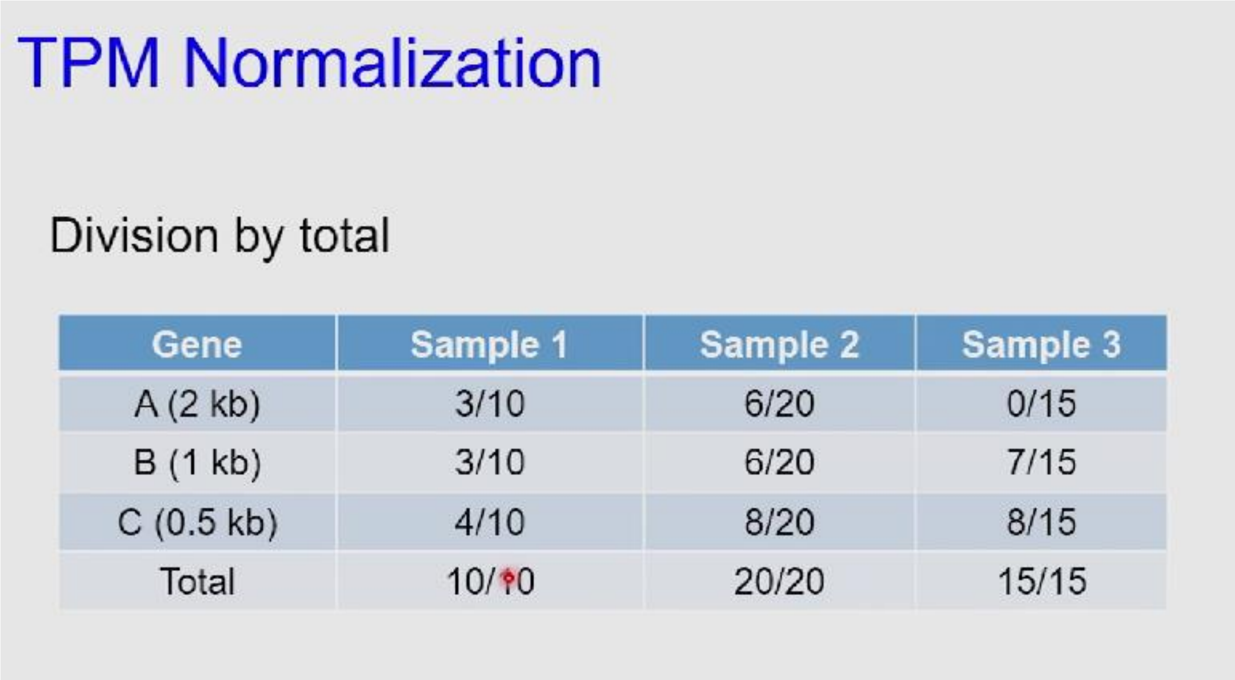


TPM Normalization

Division by gene/transcript length in kb

Gene	Sample 1	Sample 2	Sample 3
A (2 kb)	3	6	0
B (1 kb)	3	6	7
C (0.5 kb)	4	8	8
Total	10	20	15

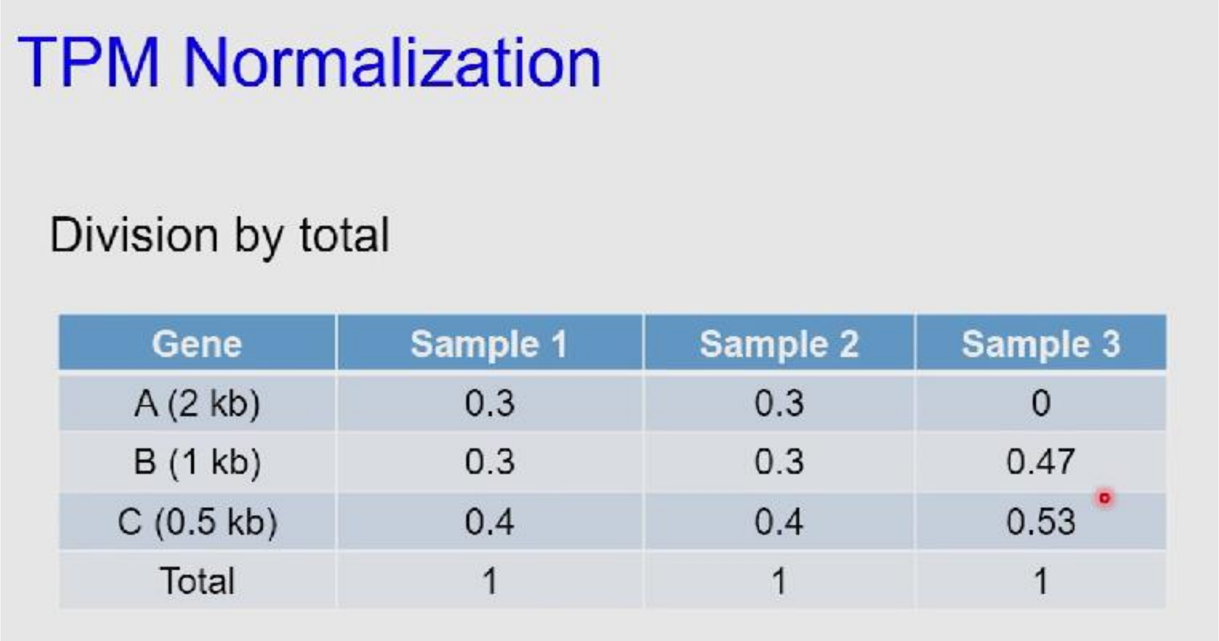
So, you have 10, 20, and 15, ok? So, the next step now is to divide by this total, ok, that we have got for each sample, ok. That is because the totals are different, right? We have to divide by this, ok? So, once we divide by the total, what we get is, as you can see, sample 1 by 10, sample 2 by 20, for each column. So, and so here for sample 3, we have divided by 15, right? So, we get the numbers, right, and as you can see now, the sum is 1 for all of them, ok?



TPM Normalization

Division by total

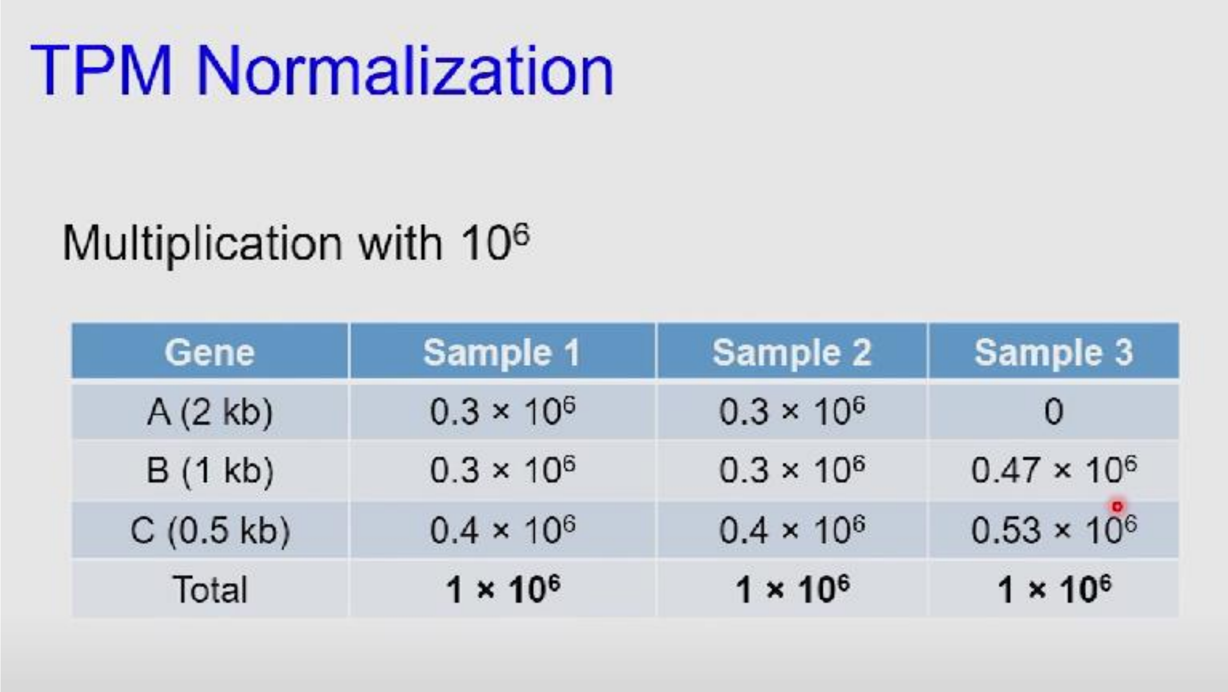
Gene	Sample 1	Sample 2	Sample 3
A (2 kb)	3/10	6/20	0/15
B (1 kb)	3/10	6/20	7/15
C (0.5 kb)	4/10	8/20	8/15
Total	10/10	20/20	15/15



TPM Normalization

Division by total

Gene	Sample 1	Sample 2	Sample 3
A (2 kb)	0.3	0.3	0
B (1 kb)	0.3	0.3	0.47
C (0.5 kb)	0.4	0.4	0.53
Total	1	1	1



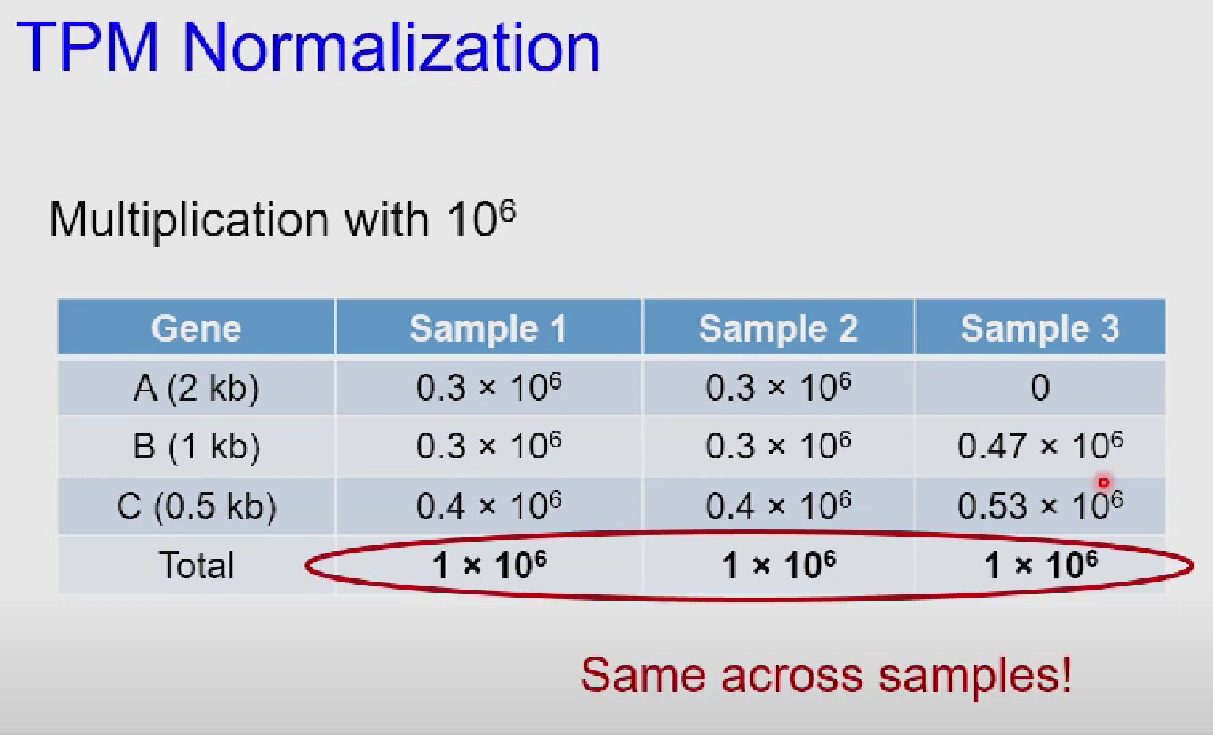
TPM Normalization

Multiplication with 10^6

Gene	Sample 1	Sample 2	Sample 3
A (2 kb)	0.3×10^6	0.3×10^6	0
B (1 kb)	0.3×10^6	0.3×10^6	0.47×10^6
C (0.5 kb)	0.4×10^6	0.4×10^6	0.53×10^6
Total	**1×10^6**	**1×10^6**	**1×10^6**

So, just a simple tweak can actually change the whole thing, ok? Sum is 1, the next step is multiply with 10 to the power 6. In this example, because the read numbers are very small, right, and we are dealing with a very small number of genes, we are getting pretty good numbers, right, 0.3, 0.4, etcetera, but in a real-life scenario with real data sets where you have millions of reads, these numbers will be very small if you just leave after the division, right, so 0.

00 something, ok. So, once we have multiplied by a million, we see that this total is exactly the same for each sample. So, that is a good thing, right? We have normalized for total sequencing depth, ok? And now we can then compare perhaps these across samples, but the problem is we have not addressed RNA composition bias, and it is still there. So, which means again we will reach some erroneous confusions because we have not addressed the issue that gene B and gene C get more reads only because gene A shows lower expression in sample 3, ok? So, to summarize whatever we have discussed so far, right, these are the drawbacks of CPM, RPKM, and TPM normalization, ok?



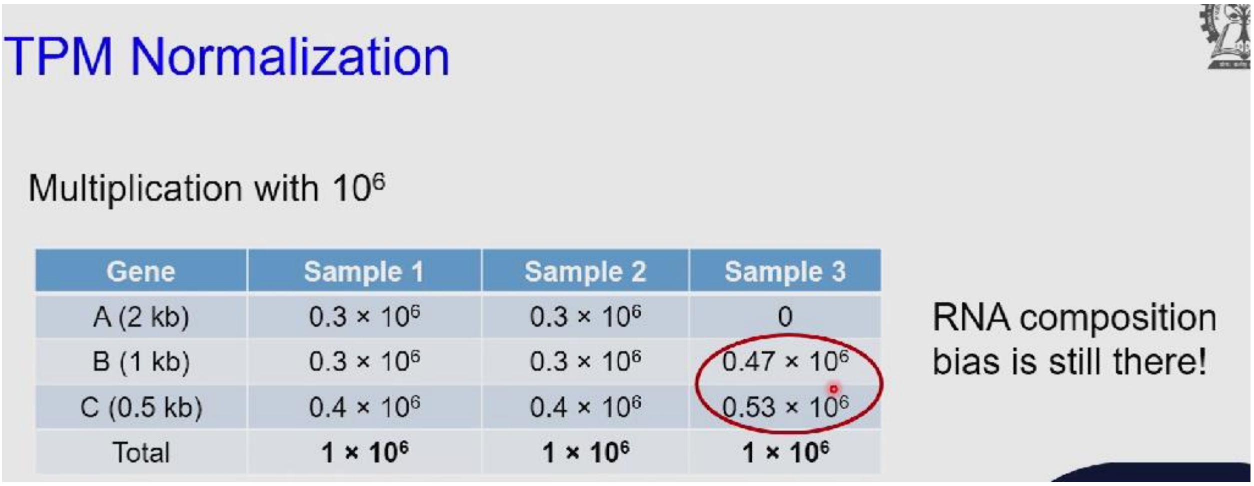
TPM Normalization

Multiplication with 10^6

Gene	Sample 1	Sample 2	Sample 3
A (2 kb)	0.3×10^6	0.3×10^6	0
B (1 kb)	0.3×10^6	0.3×10^6	0.47×10^6
C (0.5 kb)	0.4×10^6	0.4×10^6	0.53×10^6
Total	1×10^6	1×10^6	1×10^6

Same across samples!

We have discussed these methods, and we have seen that these are quite simple normalization methods. So, one of the problems with CPM normalization is that it does not normalize gene length, right? So, again, we cannot compare the expression levels of genes even within a sample because we have not taken gene length into account. We have also seen that RPKM and TPM normalization do not allow us to compare gene expression across samples because the total count is different. At the end of normalization, the total count that we get is different for each sample, and that means we have not done proper normalization, right?



TPM Normalization

Multiplication with 10^6

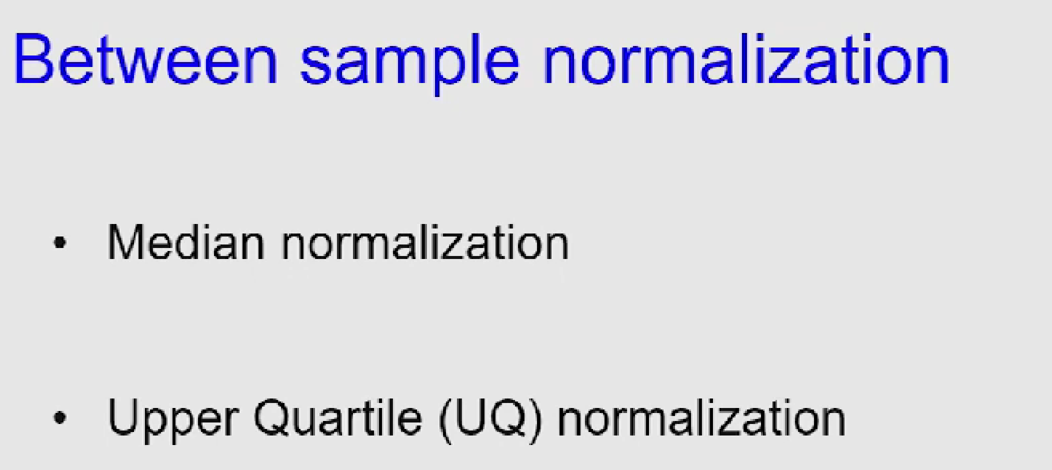
Gene	Sample 1	Sample 2	Sample 3
A (2 kb)	0.3×10^6	0.3×10^6	0
B (1 kb)	0.3×10^6	0.3×10^6	0.47×10^6
C (0.5 kb)	0.4×10^6	0.4×10^6	0.53×10^6
Total	**1×10^6**	**1×10^6**	**1×10^6**

RNA composition bias is still there!

And also, in addition, you have the RNA composition bias that we have not addressed, ok? And TPM normalization addresses these issues; we get the same total count for each sample at the end of normalization, and we also normalize for gene length, but it still does not address the RNA composition bias issue. So, we cannot really compare across samples because we might end up with some erroneous conclusions, ok? So, as you can probably understand, these methods especially RPKM and TPM, are good for within-sample comparisons. So, if you are comparing gene A versus gene C or gene 100 to gene 1000, that is probably ok with these methods.

If the moment you go comparison, you want to compare across samples, then you will face some issues. You might get some erroneous results and some erroneous conclusions, ok? So, what are the solutions, right? So, because you have seen this RPKM, TPM does not really normalize for this RNA composition bias. So, what can we do to address these situations? So, the answer is, of course, that we need to do something called sample normalization. So, RPKM or TPM methods, right, are not doing sample normalization; they are simply doing sample normalization.

So, we do the gene length and also divide by the sequencing depth. So, this is within sample normalization, and we are not comparing across samples within a data set. So, this becomes very important in many analyses that we do, ok? So, one of the major applications of RNA-Seq is to do something called differential gene expression analysis. We will talk about this differential gene expression analysis later on in the subsequent classes, where we actually look into the expression levels of the same gene across different conditions across different samples. So, which means we have multiple samples within a data set and we are comparing expression levels of genes across these samples, ok?

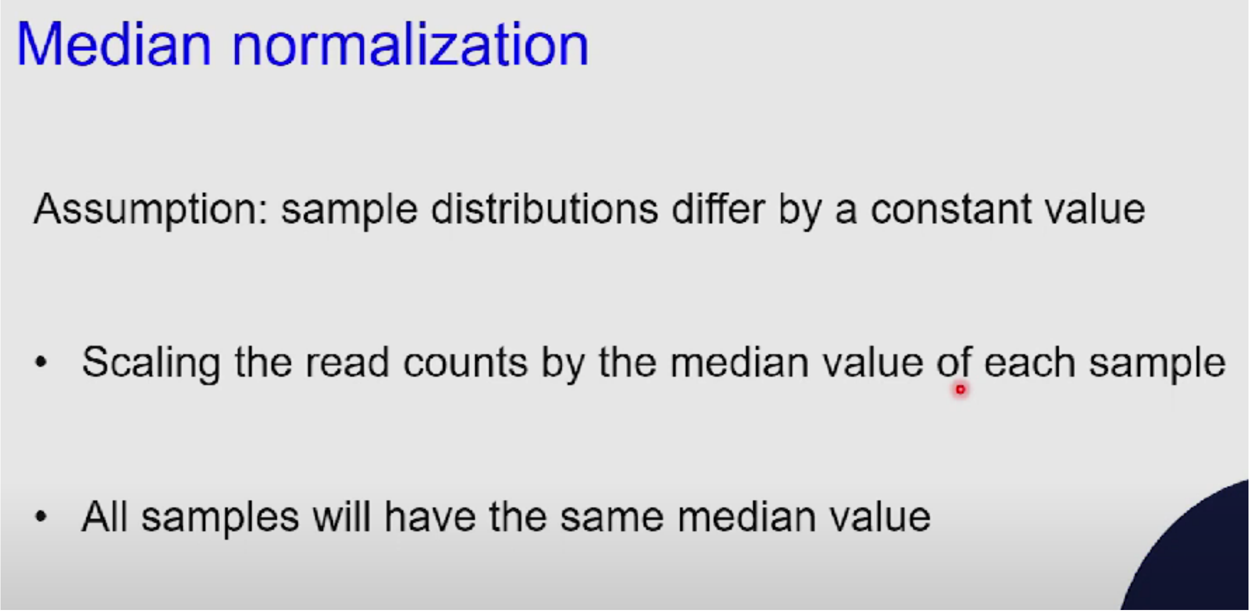


Between sample normalization

- Median normalization
- Upper Quartile (UQ) normalization

So, to do that effectively, we need something called between sample normalization, right? So, we need to address some of the issues that we have just mentioned, such as RNA composition bias, etcetera. Now, we will talk about these two methods today. Of course, there are many more sophisticated methods that we will talk about in the next class. So, the first one is called median normalization, and the second is called upper quartile or UQ.

So, let us now go into this median normalization and UQ and see, right, how the normalization is done between samples, ok? So, again, this will help you understand the more sophisticated methods in the next class, ok? So, let us talk about median normalization, ok? So, one of the assumptions in median normalization is that the read count distributions in the samples differ by a constant value, right?



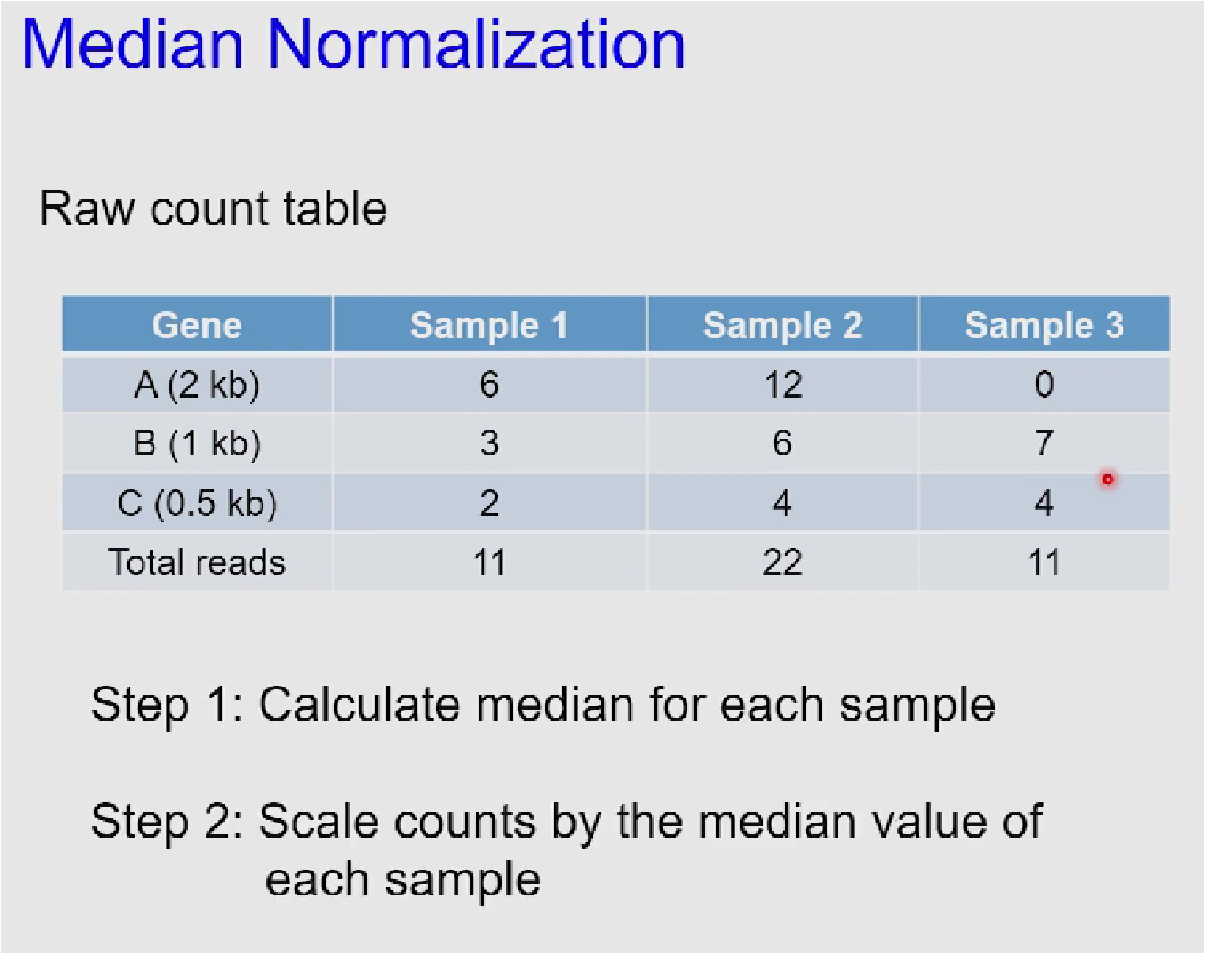
Median normalization

Assumption: sample distributions differ by a constant value

- Scaling the read counts by the median value of each sample
- All samples will have the same median value

So, if you have seen these examples of three samples, right. So, the assumption here is that the distributions differ by a constant value, and we can simply scale these distributions to bring them to the same median. So, once we start with this assumption, the goal becomes very simple, right? So, we actually scale the read counts by the median value of each sample, ok? So, we calculate the median, and we just scale the read counts by the median value. And at the end of normalization, what you would see is that all samples will have the same median value, right?

That is the idea behind this normalization, right? So, because of the assumption, the goal is simply to bring the median value to the same value for all samples. So, again, if we go back, we will not be able to calculate, of course, the median for only three numbers, which will not be appropriate, right? So, again, you can think about, right, how you are going to proceed, and then I will show the analysis with a real data set where you have 6000 or 7000 genes, and then you can probably guess, right, what will happen, ok? So, let us just for a moment think about, right, how you are going to do this normalization, median normalization, for these three samples that you see, ok, for sample 1, sample 2, and sample 3. So, what you have to do here is calculate the median value for each of these samples, right?



Median Normalization

Raw count table

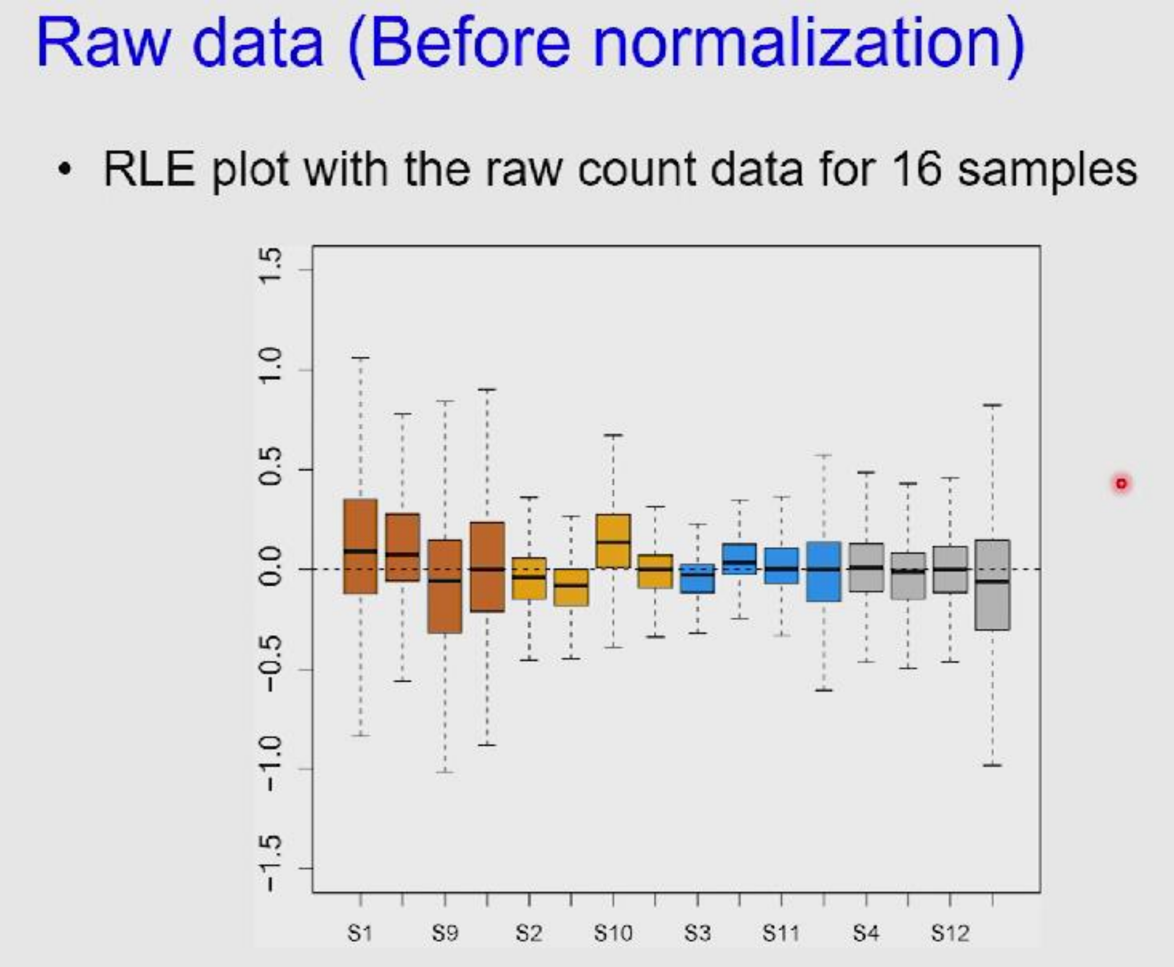
Gene	Sample 1	Sample 2	Sample 3
A (2 kb)	6	12	0
B (1 kb)	3	6	7
C (0.5 kb)	2	4	4
Total reads	11	22	11

Step 1: Calculate median for each sample

Step 2: Scale counts by the median value of each sample

So, you have this read count. So, for sample 1, you calculate the median value; for sample 2, you calculate the median value; for sample 3, you calculate the median value; and then you scale by the median value. So, you kind of scale each of these read counts with the median value that you get from each sample, ok? So, this is the idea, right? So, for one, we calculate the median for each sample here and then scale the counts by the median value of each sample. And at the end, what you will get is that the median value for each sample will be the same, ok?

Before you start the normalization, of course, they are not the same, but after normalization, they will be the same, ok? So, what I will do now is actually show the results from a real data set with 6000–7000 genes, and we can then do the median normalization, ok? So, that is what I am going to do with a real RNA-seq data set, and I will show the results in a plot called RLEplot. So, do not worry about this; it is called RLEplot, right? So, relative log expression plot. We will talk about this relative log regression calculation a bit later, but it is a good tool to visualize our results, ok?

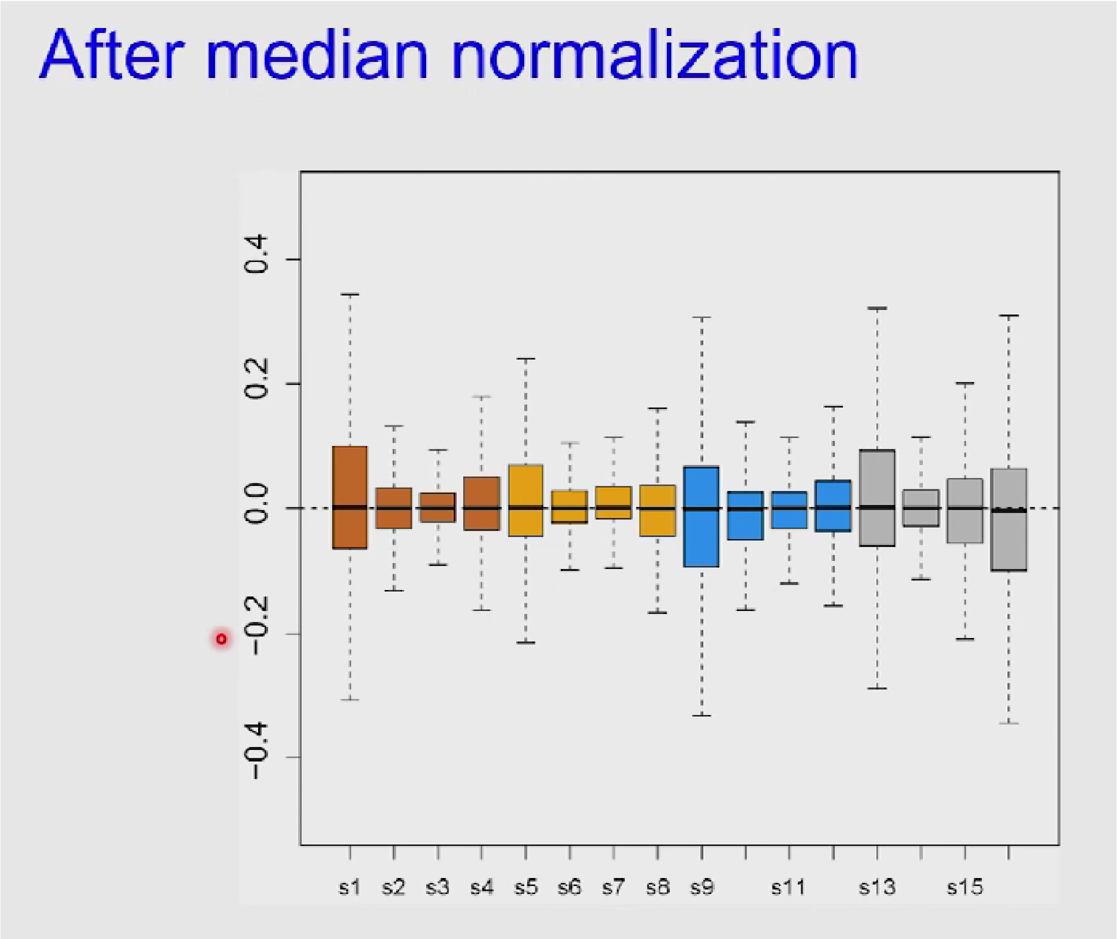


Raw data (Before normalization)

- RLE plot with the raw count data for 16 samples

So, here is the raw count data for 16 samples before normalization, ok? And the samples, ok, you can ignore the sample since they are labeled in some way, but pay attention to the colors that you see, ok. So, the colors, the samples that are of the same color, right, they are technical replicates, ok, of the same condition or same sample, ok. So, you have 16 samples, and we have 4 conditions, ok? And for each condition, you have four replicates.

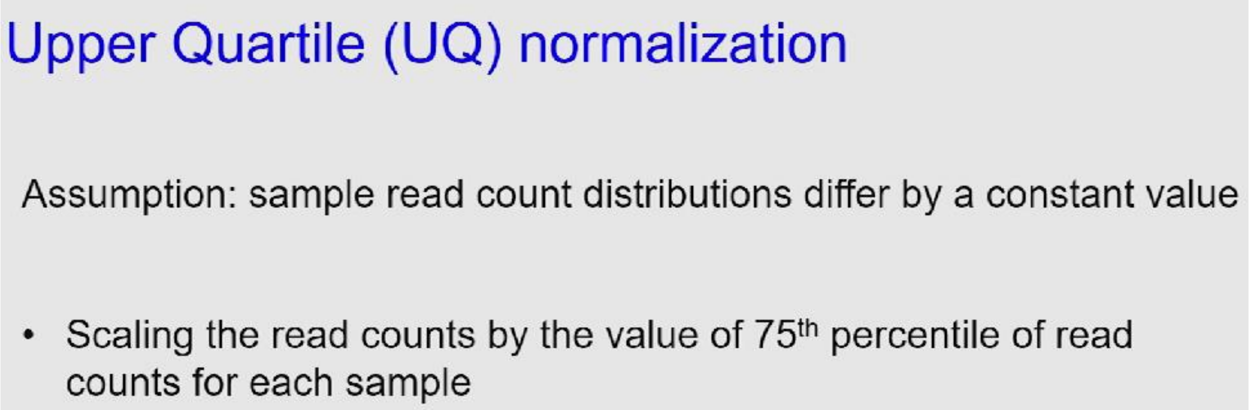
And as you can see, these replicates again show quite a bit of difference, right? For example, you can notice here, right, that this replicate of this set shows quite a different distribution, right, for raw count. For now, again, I have not explained what this RLE is, right? So, you can ignore the y axis now because this is a transformed axis because raw count cannot be negative. Right, you cannot see raw count is not negative. So, this is a transformed one, but we will use the same scale for all the plots that I showed, ok? Now, once we do the normalization, ok, you see, right, because we have done median normalization, the median is the same and is set to 0 for all the samples, right, because we have scaled by the median of that sample.



After median normalization

So, the median should become 0 for all samples, ok? So, this is after median normalization, ok? Again, we are showing the RLEplot after median normalization, ok? And you see median values are the same for all samples here, ok? So, that is the idea behind doing this median normalization. There is a variant of this, right?

So, this is called upperquartile normalization or UQ normalization, ok? So, the assumption is that sample distributions again differ by a constant value, right? So, again, read count distributions; they differ by a constant value across samples, and we can simply scale these distributions and bring the upper quartile to the same value. So, hopefully, you know what an upper quartile is, right? So, here is the scaling of the read counts by the value of the 75th percentile of read counts for each sample.

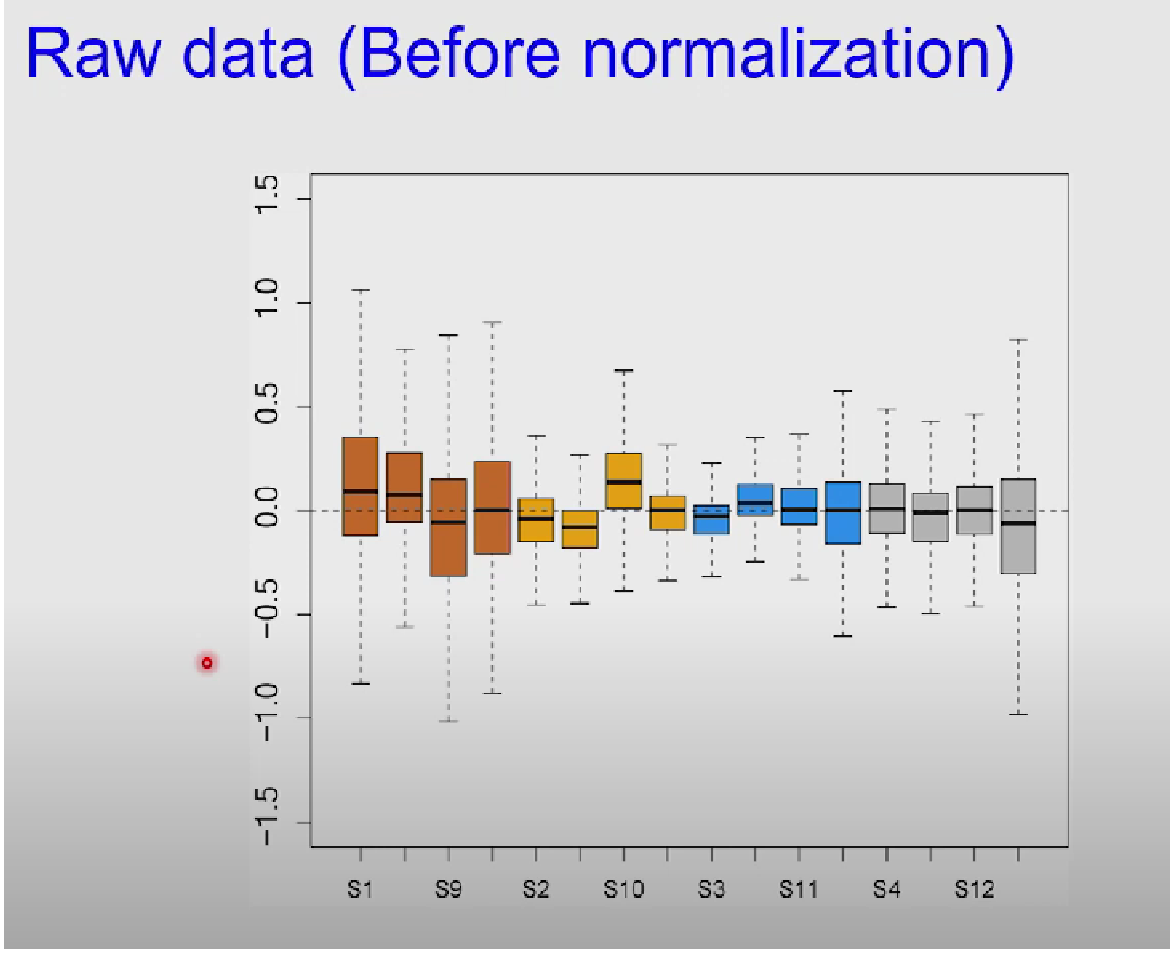


Upper Quartile (UQ) normalization

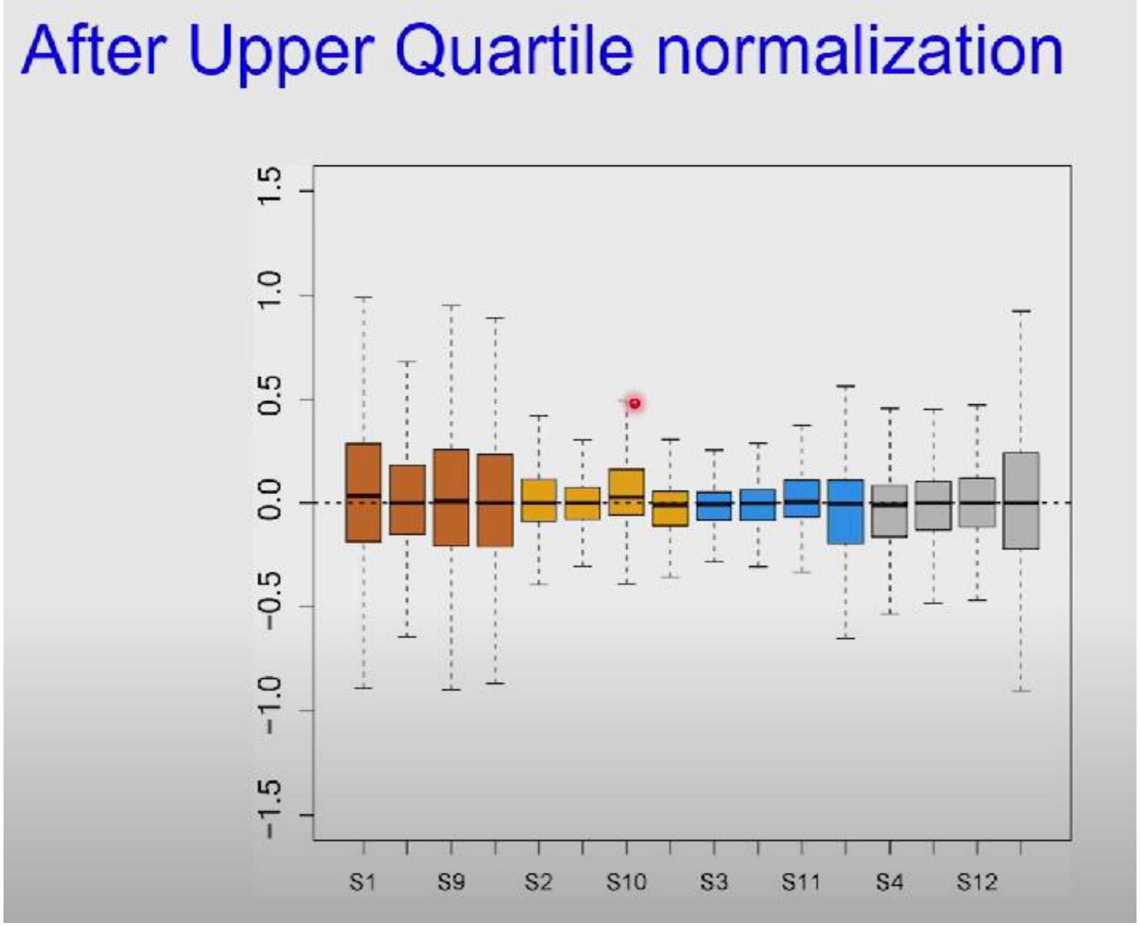
Assumption: sample read count distributions differ by a constant value

- Scaling the read counts by the value of 75th percentile of read counts for each sample

So, we have a median here; it is the 75th percentile, ok? So, again, I will show you the results from the same data, ok? So, here again is the raw data—raw count data before normalization. And this is after upper-quartile normalization. You can see that for some samples, ok, it looks pretty good, but for some samples here, for example, this one or this one, the median is not 0, ok. And that is expected because here we set the 75th percentile to the same value and not the median.

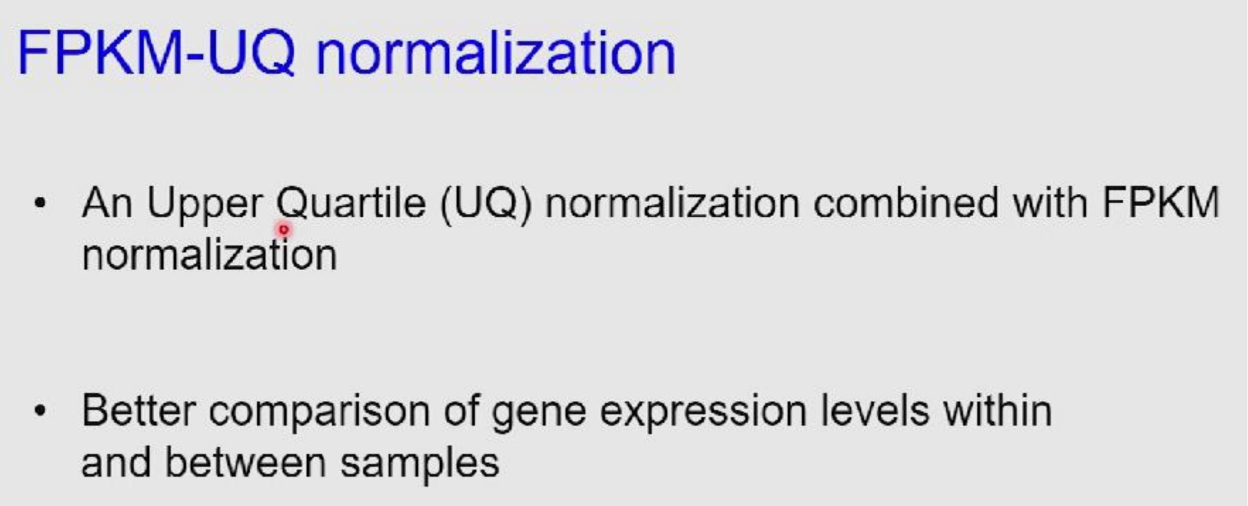


Raw data (Before normalization)



After Upper Quartile normalization

So, these are some of the methods that we can use. Now, there is a hybrid method that combines this FPKM with upper quartile normalization. So, in FPKM, at the end, what you got was that the total count was different for each sample, right? And so, you can address that if you do something called upper quartile normalization, combine that with equal normalization, and you can address this issue, right? So, here the upper quartile normalization is combined with FPKM normalization, and this helps us in a better comparison of seen expression levels within and between samples, ok? We are doing within sample normalization, which is the FPKM, followed by an between sample normalization, which is the UQ normalization.



FPKM-UQ normalization

- An Upper Quartile (UQ) normalization combined with FPKM normalization
- Better comparison of gene expression levels within and between samples

And this has been done for many datasets. You will find many public datasets where the count data that is given is sometimes referred to as FPKM UQ data. So, once you see this kind of data, you will probably then understand, right, what has happened. The first normalization was FPKM, then there was a UQ normalization, ok? So, what are the drawbacks, right? So, we have talked about some of the differences between sample normalization and one of the first drawbacks, which is that the assumption that the sample will vary by a constant value does not hold for most cases. It is not just that the distributions are shifted and the median should be brought to the same value; this should not hold for most samples, ok?

And also, we have not addressed the issue of RNA composition bias here, right? So, we have not touched upon that at all when you are talking about this between sample normalization here, ok? So, here are the references that we have utilized for this part. So, to summarize, we have looked into normalization methods for gene length and total sequencing depth.

We have talked about some of these methods, right? So, FPKM, RPKM method, and TPM method, right? And then we have seen, right, that these methods are good for comparison and expression within samples, but they are not suitable for comparison between samples. So, we have explained the reasons why that is the case. And then we talked about normalization between samples, as well as median normalization and UQ normalization. So, these are only two methods that we have discussed, but again, they have not addressed some of the issues that are still remaining, and that is why we need better methods, which we will talk about in the next class. Thank you.