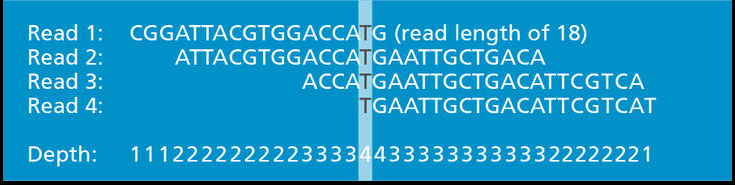
<http://www.rna-seqblog.com/rpkm-fpkm-and-tpm-clearly-explained/>

<https://statquest.org/2015/07/09/rpkm-fpkm-and-tpm-clearly-explained/>

https://github.com/hbc/knowledgebase/wiki/Count-normalization-methods

**Coverage** (or **depth**) in [DNA sequencing](https://en.wikipedia.org/wiki/DNA_sequencing) is the number of reads that include a given [nucleotide](https://en.wikipedia.org/wiki/Nucleotide) in the reconstructed sequence.



So we can see here there’s a max read depth of 4.

From: https://genohub.com/next-generation-sequencing-guide/

BUT for RNA sequencing, its abit more complicated, because of differential expression, meaning that more reads will be captured from highly expressed genes while fewer reads will be captured by genes expressed at low levels, there is no constant template amount to base our depth.

Therefore we use mapped reads.

It is important to distinguish between total reads and mapped reads, as not all reads will map onto a reference genome, so the number of usable reads will be less than the number of actual reads.

Okay what is RPM? And how does LiLing do it?

Wait a second, it seems like we are 1 step to RPKM.

She just didn’t divide by length of each feature. If not we’ll have RPKM… Hmnn

If we see a spike in 1 library, it could mean that that library has a deeper sequencing depth,

If we see a spike in 1 type of catergory, it could mean that its features are significantly longer

Question:

1. She did not use RPKM, ie. Normalizing for the length of each mapped feature,

i.e obviously a longer feature will have a higher propensity to generate reads.

But can we? We would have to do it really downstream right?

1. Is this sufficient? Should/ can we even us TPM? Can we even do that?

Question:

What the fuck is TPM?

So apparently we change the order of RPKM steps?

1. We first map, get total counts that map to each feature.
2. Scale counts by feature length to get RPK for each feature.
3. Then add up the RPKs of all features, divide by million, we get the RPK scaling factor
4. Then divide each feature RPK by RPK scaling factor to get TPM of each feature.

Now what happens is that, for each library/ replicate, the total TPM of all features == 1. We therefore get a sense of proportion.

So TPM for each feature tells us, for each feature, what proportion of all mapped reads map to that feature.

“**feature**”, I’m referring to an expression feature, by which I mean a genomic region containing a sequence that can normally appear in an RNA-Seq experiment (e.g. gene, isoform, exon).

“**Counts**” usually refers to the number of reads that align to a feature

“**Random variable** X_i” to denote the counts you observe from a feature of interest i.

* is thus dependant on : 1) number of fragments sequenced and 2) effective length of feature i.

**Effective (Comparing sample counts)**

**Effective length** reflects of possible start positions for a read or fragment within that transcript, given that the read or fragment must fit entirely within the transcript boundaries.

This basically makes an estimate of

**Effective counts** then takes counts \* length/effective length. Now, considering how (length/effective length) is most likely > 1, this then scales up our counts quite obviously. Corrects for feature size, smaller features would have a lower effective length?

Therefore the larger the feature, the closer to 1, length/effective length is.

So all we have done above lets us compare the **same feature between samples, ASSUMING SAME SEQUENCING DEPTH with effective counts.**

**TPM (Comparing within samples)**

Okay, now lets say we want to compare counts between **feature X and Y within a sample**. But lets think about it for a second, seems pretty unfair to compare counts between features of different lengths. Of course a longer feature is gonna have a higher chance of accumulating counts.

We therefore have to normalize for each feature and come up with a “**rate of reads**”, think about it like a reads per something. This then gives us a fair base for comparison.

Our rate in this case is “**Counts per base**”, basically “**Observed Counts/ Effective length**”

We are then able to use this fair basis for comparison to derive Transcripts per million (**TPM**). This is a measurement of the proportion of transcripts in your pool of RNA. In a longer version, that if you were to sequence one million full length transcripts, **TPM is the number of transcripts you would have seen of type i**, given the abundances of the other transcripts in your sample.

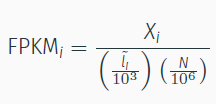
\*\*This is why we cannot use TPMi to compare between samples, because between the samples, the abundances of other transcripts and sequencing depth will surely differ.

**RPKM FPKM (Comparing between samples)**

Why for SE reads, RPKM=FPKM.. Hmnn makes sense right, 1 read = 1 fragment.

But for PE reads, we SHOULD NOT use RPKM, instead we use FPKM? Cos we are PE sequencing 1 entire fragment?

So lets just work with FPKM, k. **Fragments per kilobase of exon per million fragments mapped**.



N=Total number of reads sequenced (\*\*For paired end we would take N/2, because we are looking for total fragments sequenced)

From here you can see that its basically effective counts / (effective length/1kb)\*(total fragments sequenced/1M)

We therefore expect to see FPKMi fragments for each thousand bases in the feature for every N/10^6 fragments sequenced.

So what FPKM magically does is that it corrects not only for feature size, BUT ALSO for the total number of fragments sequenced (sequencing depth). <- link back to the earlier green point!

