At the end of either bulk RNA-seq or scRNA-seq, all quality controls are done and you have annotation and counts. Basically, we have a table with columns that represent either different samples (in the case of bulk analysis) or different cells (scRNA-seq analysis). Now the focus will be on bulk analysis. How to visualize data in a way that allows to highlight differences in gene expression between different samples?

First step: **data normalization**. The standard format is TPM (transcripts per million), then there is the CPM (counts per million). TPM has replaced the previous normalization method (FPKM). We will first see another format, which is RPKM (reads per kilobase million), analogue to FPKM:

RPKM (Reads Per Kilobase Million)

Let's imagine this RNA-seq data: 3 replicates (called Rep 1, 2, 3) for a genome made of 4 genes (A, B, C, D) which differ in size (A = 2kb, B = 4kb, C = 1kb, D = 10 kb).

Gene Name	Rep1 Counts	Rep2 Counts	Rep3 Counts
A (2kb)	10	12	30
B (4kb)	20	25	60
C (1kb)	5	8	15
D (10kb)	0	0	1

While Rep1 and Rep2 are very similar, Rep3 has way more counts, suggesting Rep3 is richer in counts than Rep1 and Rep2. Gene D is poorly expressed across all of the samples. Note that gene B is twice longer than gene A, so the number of reads of gene B is double compared to that of gene A.

Step 1: normalize for read depth → the number of reads (fragments) is related to the length of genes: a 2kb gene will likely give half reads than a 4kb gene. This concept is even more complicated considering real experiments with tons more and longer genes. So let's sum all reads for each column (e.g., 35 reads for column Rep1, 45 for Rep2, and 106 for Rep3). Then we divide these total read counts for a number that in normal situations would be 1M, while in this instance where there are few reads we divide by 10: we obtain the Tens of Reads (our reference values). We basically scaled the number of reads.

Gene Name	Rep1 Counts	Rep2 Counts	Rep3 Counts
A (2kb)	10	12	30
B (4kb)	20	25	60
C (1kb)	5	8	15
D (10kb)	0	0	1
Total reads	35	45	106
Tens of reads	3.5	4.5	10.6

At this point, the **RepX Counts** for each gene should be divided by the corresponding **Tens of reads** value in order to remove the effect given by the fact that each single column has more read than the others: in the end the three replicates (columns) display comparable number of reads for each gene:

Gene Name	Rep1 Counts RPM	Rep2 Counts RPM	Rep3 Counts RPM
A (2kb)	10/3.5 = **2.86**	12/4.5 = **2.67**	30/10.6 = **2.83**
B (4kb)	20/3.5 = **5.71**	25/4.5 = **5.56**	60/10.6 = **5.66**
C (1kb)	5/3.5 = **1.43**	8/4.5 = **1.78**	15/10.6 = **1.43**
D (10kb)	0/3.5 = **0**	0/4.5 = **0**	1/10.6 = **0.09**

Normally we divide by 1M and indeed we obtain scaling using the "per million" factor (M). After **Step 1**, the number of reads for each replicate is comparable.

<u>Step 2: normalize for gene length</u> → there are still differences between genes related to highly different lengths (e.g., gene A *vs* gene B). This step will solve this issue. We need to divide

Gene Name	Rep1 Counts RPKM	Rep2 Counts RPKM	Rep3 Counts RPKM
A (2kb)	2.86/2 = **1.43**	2.67/2 = **1.33**	2.83/2 = **1.42**
B (4kb)	5.71/4 = **1.43**	5.56/4 = **1.39**	5.66/4 = **1.42**
C (1kb)	1.43/1 = **1.43**	1.77/1 = **1.78**	1.43/1 = **1.43**
D (10kb)	0/10 = **0**	0/10 = **0**	0.09/10 = **0.009**

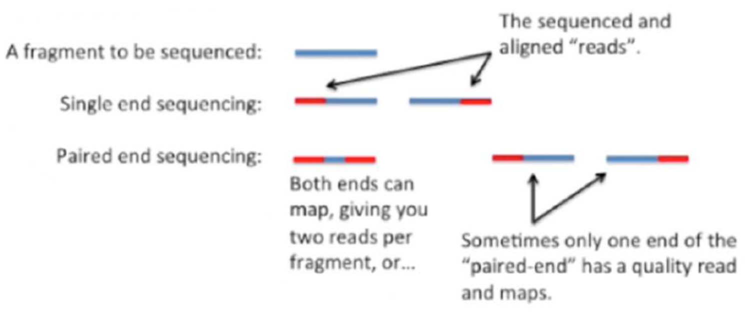
Now reads are scaled both for depth (M) and gene length (K). At this point wit removed difference both between columns (Step 1 made the different Replicates more similar in terms of number of reads) and between rows (**Step 2** made a normalization to give you the <u>number of normalized reads per base</u>).

This is the RPKM normalization and it is summarized by this formula:

$$\mathrm{FPKM}_i = \frac{X_i}{\left(\frac{\tilde{l}_i}{10^3}\right)\left(\frac{N}{10^6}\right)} = \frac{X_i}{\tilde{l}_i N} \cdot 10^9$$

X_i = counts of gene i, which is divided by: the length of the gene ($\tilde{l}_i$) in turn divided by 1000 (as the bases of our genes are represented as a number of kilo: kb) multiplied to the total number of reads for each column (replicate) divided to 1M ($\frac{N}{10^6}$).

The $\left(\frac{N}{10^6}\right)$ part of the equation is basically the Step 1, while the whole formula is what we do in Step 2.


A fragment to be sequenced:
Single end sequencing:
Paired end sequencing:
The sequenced and aligned "reads".
Both ends can map, giving you two reads per fragment, or...
Sometimes only one end of the "paired-end" has a quality read and maps.

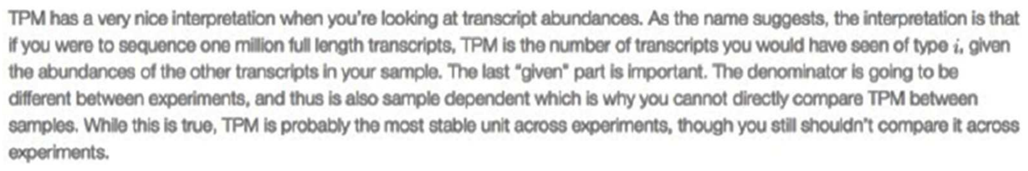
- **RPKM = Reads Per Kilobase Million** (for single end RNA-seq) → single end sequencing gives 1 read per fragment.
- **FPKM = Fragments Per Kilobase Million** (for paired end RNA-seq) → FPKM keeps track of fragments: one fragment with 2 reads (paired end sequencing gives 2 reads per fragment) is not counted twice.

<u>TPM (Transcripts Per Million)</u>

This normalization method works oppositely compared to RPKM:

- **Step 1** → normalization of data in the dataset to the size of genes (**each count divided by gene length**)
- **Step 2** → calculation of the scaling factor for each column (**normalization to sequencing depth**)

$$\mathrm{TPM}_i = \frac{X_i}{\tilde{l}_i} \cdot \left(\frac{1}{\sum_j \frac{X_j}{\tilde{l}_j}}\right) \cdot 10^6$$


TPM has a very nice interpretation when you're looking at transcript abundances. As the name suggests, the interpretation is 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. The last "given" part is important. The denominator is going to be different between experiments, and thus is also sample dependent which is why you cannot directly compare TPM between samples. While this is true, TPM is probably the most stable unit across experiments, though you still shouldn't compare it across experiments.

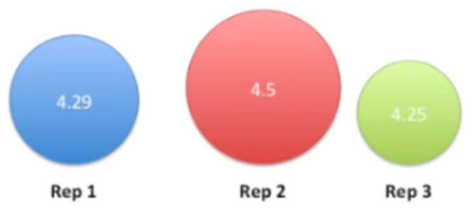
TPM normalization is better than RPKM as it enables ranking of gene expression (genes are ranked based on their expression level). The advantage of TPM is that all of the samples have the same "total size" after normalization and can be easily compared.

What discriminates TPM from RPKM is the part after normalization: the sums of each column are very different in the two cases (see tables below). The main point is that in TPM all samples (Rep) have the same total size after normalization and can be compared, while in RPKM they don't.

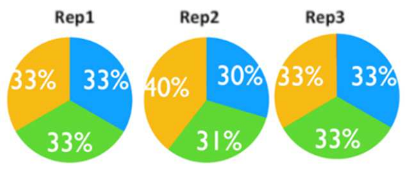
Gene Name	Rep1 Counts RPKM	Rep2 Counts RPKM	Rep3 Counts RPKM
A (2kb)	**1.43**	**1.33**	**1.42**
B (4kb)	**1.43**	**1.39**	**1.42**
C (1kb)	**1.43**	**1.78**	**1.43**
D (10kb)	**0**	**0**	**0.009**
Total:	4.29	4.5	4.25

Gene Name	Rep1 Counts TPM	Rep2 Counts TPM	Rep3 Counts TPM
A (2kb)	**3.33**	**2.96**	**3.326**
B (4kb)	**3.33**	**3.09**	**3.326**
C (1kb)	**3.33**	**3.95**	**3.326**
D (10kb)	**0**	**0**	**0.02**
Total:	10	10	10

RPKM:



TPM:



VISUALIZING EXPERIMENTAL DATA

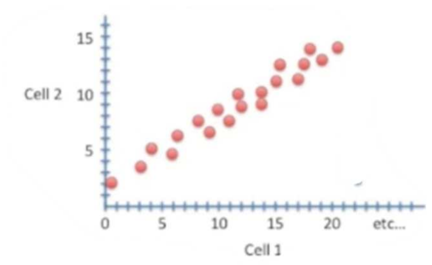
One way is through the **PCA** (Principal Component Analysis) which represents data in a reduced space based on **variance** between data → dimensional reduction is always necessary when representing this kind of data, as all these experiments are multidimensional and cannot be represented in 3D.

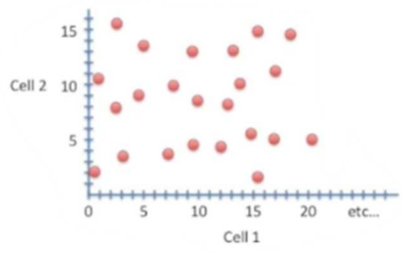
Check this video about PCA: https://www.youtube.com/watch?v=_UVHneBUBW0

Let's consider to have this kind of data set:

Gene	Cell1 reads	Cell2 reads
a	#	#
b	#	#
c	#	#

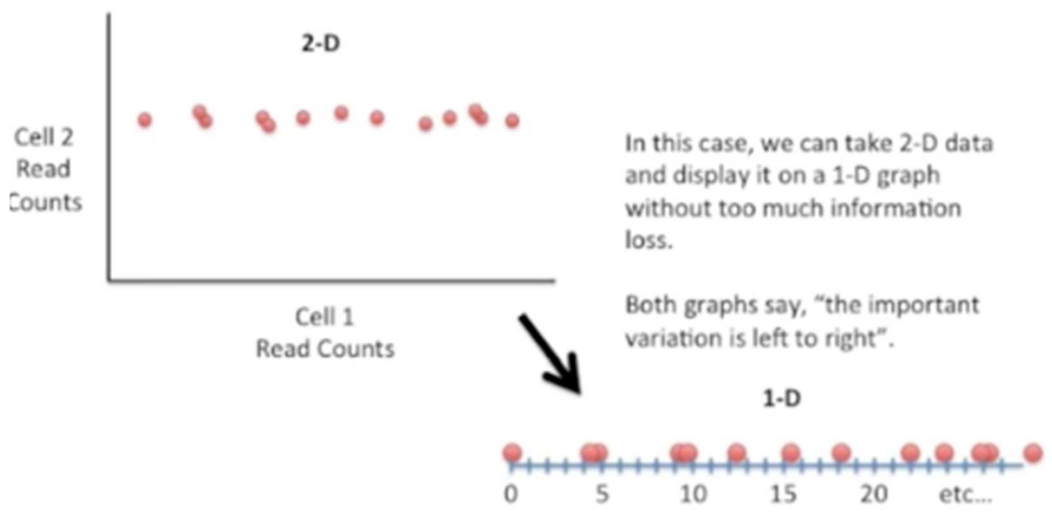
In the graph on the left, gene expression in cell 1 and cell 2 is correlated, while in the right graph it isn't:





Dots represent genes.

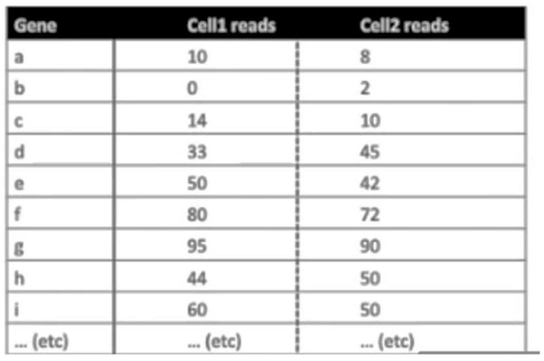
Basically, each cell we sequence adds another dimension. There are instances in which some dimensions are more important (informative than others):



In this case, we can take 2-D data and display it on a 1-D graph without too much information loss.

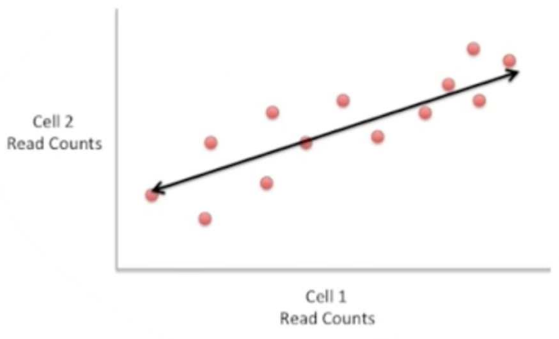
Both graphs say, "the important variation is left to right".

Now let's consider this data set (just 2 cells):

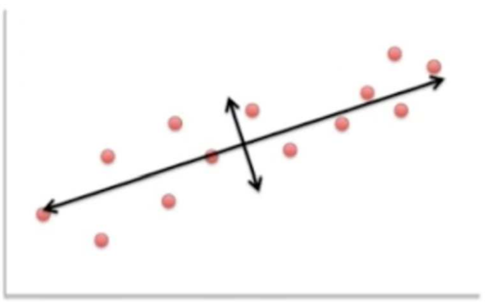


Gene	Cell1 reads	Cell2 reads
a	10	8
b	0	2
c	14	10
d	33	45
e	50	42
f	80	72
g	95	90
h	44	50
i	60	50
... (etc)	... (etc)	... (etc)

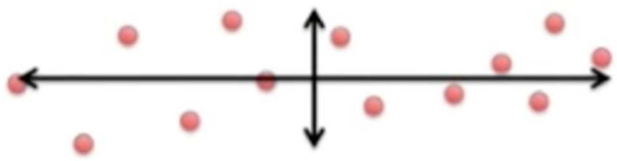
Genes are dots and in this instance are spread along a diagonal → i.e. the maximum variation in the data of the data set is in between the two endpoints of this line:



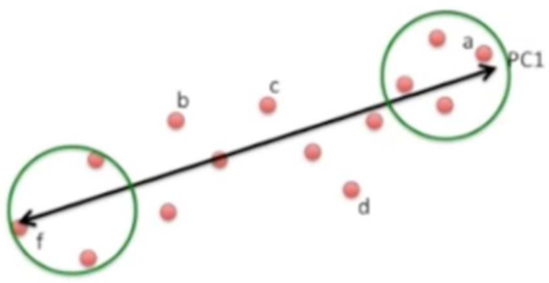
The 2nd maximum variation that can be defined is the following one (as dots/genes are also spread a little above and below the first line):

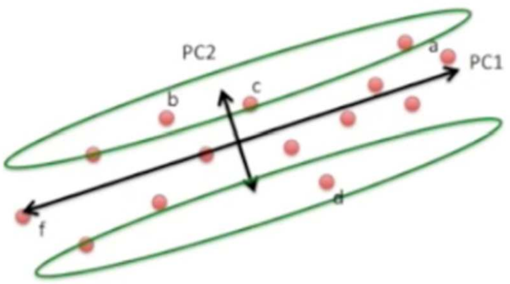


Now interestingly by rotating these two lines we can define new X and Y axes:

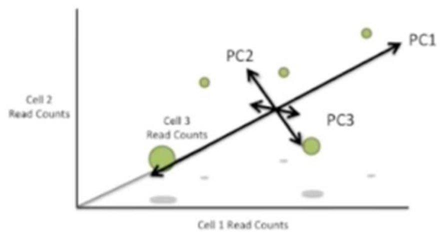


In particular, X and Y axes take the name of Principal Component 1 and Principal Component 2 (PC1, PC2). Of course not all genes are equally important in estimating the differential expression between the two cells (in this case we oversimplified considering two cells, but a real situation would refer to different samples and/or replicates of an experiment), meaning that considering the PC1 gene a and f are more relevant, while gene b and d have an impact on the PC2 dimension:

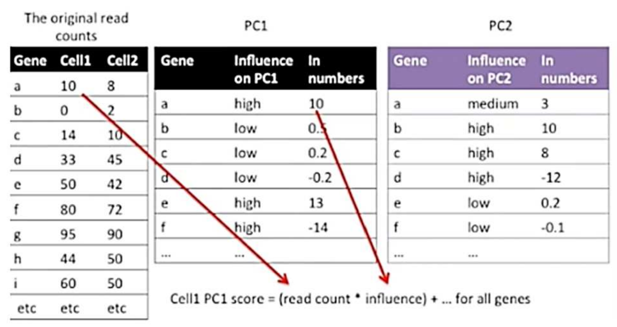




PC dimensions span the direction where variation is enhanced the most. If we had 3 cells instead of 2, so if we delt with a 3D space (3 axes), then we would have had 3 PCs (where PC3 spans the direction of the 3rd most variation, and so on by adding even more cells/dimensions):



Each gene has an influence on each PC component (the entity of this influence is expressed as a number). It is possible to calculate the PC1 and PC2 (and PC3, 4, … it depends on how many dimensions you have) scores for each cell in this way:



The original read counts

Gene	Cell1	Cell2
a	10	8
b	0	2
c	14	10
d	33	45
e	50	42
f	80	72
g	95	90
h	44	50
i	60	50
etc	etc	etc

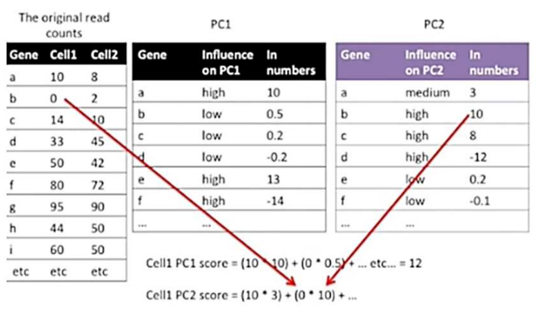
PC1

Gene	Influence on PC1	In numbers
a	high	10
b	low	0.5
c	low	0.2
d	low	-0.2
e	high	13
f	high	-14
…	…	

PC2

Gene	Influence on PC2	In numbers
a	medium	3
b	high	10
c	high	8
d	high	-12
e	low	0.2
f	low	-0.1
…	…	

Cell1 PC1 score = (read count * influence) + … for all genes



The original read counts

Gene	Cell1	Cell2
a	10	8
b	0	2
c	14	10
d	33	45
e	50	42
f	80	72
g	95	90
h	44	50
i	60	50
etc	etc	etc

PC1

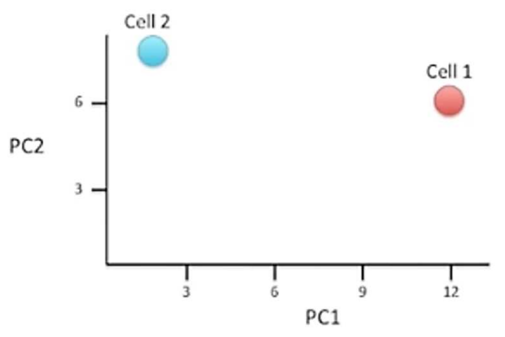
Gene	Influence on PC1	In numbers
a	high	10
b	low	0.5
c	low	0.2
d	low	-0.2
e	high	13
f	high	-14
…	…	

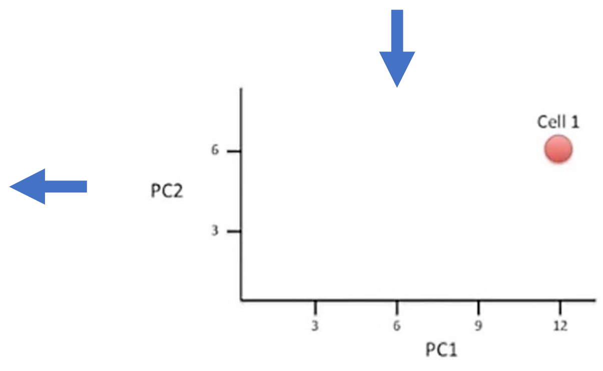
PC2

Gene	Influence on PC2	In numbers
a	medium	3
b	high	10
c	high	8
d	high	-12
e	low	0.2
f	low	-0.1
…	…	

Cell1 PC1 score = (10 * 10) + (0 * 0.5) + … etc… = 12

Cell1 PC2 score = (10 * 3) + (0 * 10) + …





Now calculate scores for Cell2

Cell2 PC1 score = (8 * 10) + (2 * 0.5) + ... etc... = 2

Cell1 PC1 score = (10 * 10) + (0 * 0.5) + ... etc... = 12

Cell1 PC2 score = (10 * 3) + (0 * 10) + ... etc... = 6

If we had sequenced a 3rd cell, and its transcription were similar to that of Cell1, then Cell3 would have had a score similar to Cell1 (and the two cells would have been close on the graph).

Influence numbers are weights for the importance of each gene to a PC. In PCA terminology, weights are called "loadings" and an array of "loadings" for a given PC is called an "eigenvector".