Data analysis – Prof. Calogero

Lesson 6

SUMMARY OF THE TOPICS DONE UNTIL NOW

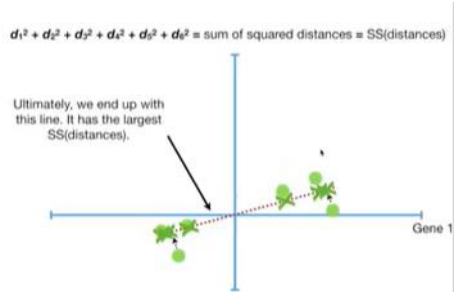
We went in parallel with bulk RNAseq and scRNA seq, performing the quality control of data. Bulk → fastQC and multiQC. Sc → fastQC of the R2 (which contains the mapping information). The converter of the fastq to counts provides some levels of quality control of the reads that are mapped on a single cell, and the features of the mapping. We have evaluated quality at the level of experimental setting.

In bulk, PCA → overall view of similarity/differences of data during differential expression; sc → data reduction is already part of the way I characterize the population during a sc experiment. Bulk RNAseq tries to extract genes that change between different conditions, while scRNAseq tries to show what changes during a perturbation of a dataset, or what changes in the sub-populations of cells present in a complex sample. The datasets that I will use in these experiments are 2 different views of the same experiment: from the bulk I have the PC9 cell line, which is EGFR+ → it is sensitive to Osimertinib. There are different replicates of the same cell line, which are treated chronically or acute treated. In this case, the bulk analysis that I do consists in evaluating the differentially expressed RNAs during and after the drug treatment. Regarding the sc analysis, I have the untreated cell line injected in a mouse (similar physiological tumor growth) and the same cell line injected in a mouse and treated with Osimertinib for 24 hours. What am I expecting? From one side, I analyse how genes change during the acute and the chronic treatment → are the acute and chronic treatments change the same set of genes? In case of the sc, the question is: are we going to see the same number of subpopulations related to the cells, or are there new ones? What happens if I take the same subsets of genes of the bulk and cluster them during my sc experiment? General point: understand how resistance is generated in this tumor, working with bulk and sc RNA analyses. In the 1st part of this lesson, we focus on the bulk, and then we will move to scRNAseq.

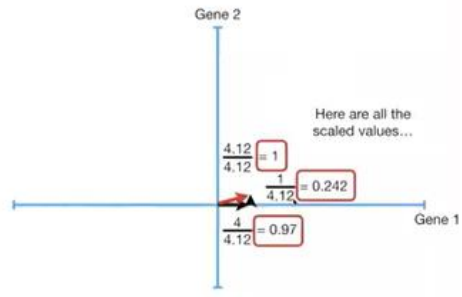
TODAY'S LESSON

In the previous example, the weights are associated to each gene, and they determine variance in the experiment. How can I define the weights? To do so, I work with 2 genes and many cells (which is the opposite thing that I did in the previous lesson, where many gens and only 2 cells were used). Given 6 mice, I can take the expression level of a gene (ex: gene 1) and plot it on the x axis, which will be called "gene 1". Then, I can calculate the mean value of the numbers, to obtain the central point on the x axis. Now I can do the same procedure on the y axis, that will be called "gene 2". Now, I can build 2D representation of the 6 mice working with 2 genes. The median point is given by the coordinates of the median values, and I can now use it as a reference: it must be centered at the value 0 of the x and y axis. Now, I don't consider anymore gene 1 and gene2, but I consider how the mice start changing their expression according to the average one. Data will be represented according to this mean value (which is used as the 0 point → data will be represented according to the mean of the 2 genes). At this point, I draw a line, centered at the 0 coordinate of x and y axes: this line will be used to represent a sort of projection of the data on this diagram. This projection allows to represent the plotted data on the line. Why do I do this? Because I firstly have to graphically represent the greatest amount of variance of data. The principal component 1 (PC1) can be obtained by minimizing the distances between the projections and the real points: this can be done by moving the segment until the distance point-projection is minimized. Another way I can look at data is to maximize the projections over the 0 point. As the projection gets nearer the real point, the distance projection-0 increases. The preferred thing is to look at the projections over the segment, because it is much easier to be calculated.

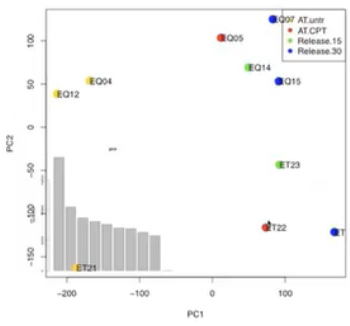
Example: let's look at one mouse.



There is a specific distance between the dot and 0, and I can draw it. Since the segment is fixed, other segments are used, and they represent the distance between the point and the segment. A square triangle is formed → Pitagora's theorem to calculate a (the distance from 0 and the point). While a is fixed, the other 2 elements (b and c)change: b is minimized, c is maximized as they get nearer a. Now, the various points can be projected on a random segment, and the distance between the points and the segment can be calculated. To eliminate the – values, every number (+ and -) has to be squared. Then, the values are summed, to obtain the sum of the square distance of the projected points. Since I want to maximize the value, the sum of the distances increases when I get farer from the 0 point: in the picture I can see the distance at which I maximise the distances with respect to the origin of the projected value. Once distances are maximized, the segment representing the largest variance between the points is called "PC1". The second one is perpendicular to the first one, the third is perpendicular to the 2nd and so on. So, they are easier to calculate. In this example, only two components will be generated, because I work with 2 sets of data only; number of components = number of datasets, even though the first 2 or 3 elements express most of the variance. PC1 has a slope (in the example, it is 0.25: this number is used to facilitate the understanding how the PCA components are generated). Each point of the line can be projected on the x and y axis, according to the formula y=mx. Given the slope of 0.25 (y/x), 4 parts of gene 1 (PC1) and 1 part of gene 2 (PC2) are needed, because ¼ = 0.25. Practically, in order to generate the diagonal, 4 parts of gene 1 and 1 part of gene 2 are needed → slope.



What is the problem at this level? PCA is always represented by making the hypotenuse equal to 1: this way, every PCA can be compared when I look at the same data. In the example, the hypotenuse is equal to 4.12. In order to make it equal to 1, everything must be divided by 4.12 (1/4.12, 4/4.12 and 4.12/4.12) → the hypotenuse is 1, and the 2 lines are 0.242 and 0.97, however their ratios do not change (they are always ¼). This type of normalization is useful to have "easy" PCA component. To make PC1, I need to mix 0.97 parts of gene 1 and 0.242 parts of gene 2 → the best fitting is obtained for PC1. What about PC2? PC2 will be perpendicular, and data will be simply projected on the new segment. The calculation of PC2 allows me to understand which is the amount of gene 1 and gene 2 that is describing this variance; the count I have to do is the same, but I perform it on the PC1's perpendicular component. The PC1's one-unit long vector is called "eigenvector", while the parts of gene 1 and 2 are called "loading scores" (the weights that I have to use to correct the counts to make the specific gene important for a specific component during my analysis). Loading scores = weights that represent the PC on a plot. What is critical about PCA? The definition of the 1st components: all the others are easy to define.



PCA is a linear data reduction approach used to represent data, because I linearly look at the dispersion of data (because I work with lines in a graph, and not with curves). Why do I need that? Example. The plot indicates 10 possible principal components (because 10 samples are present). The 1st one (PC1) represents 100% of the variance, while the 2nd one represents the 50% of the variance. The numbers of each component are referred to each component only: what happens on PC1 and what happens on PC2 is different and driven by different genes. The last component has the variance very close to 0.

PCA example #1

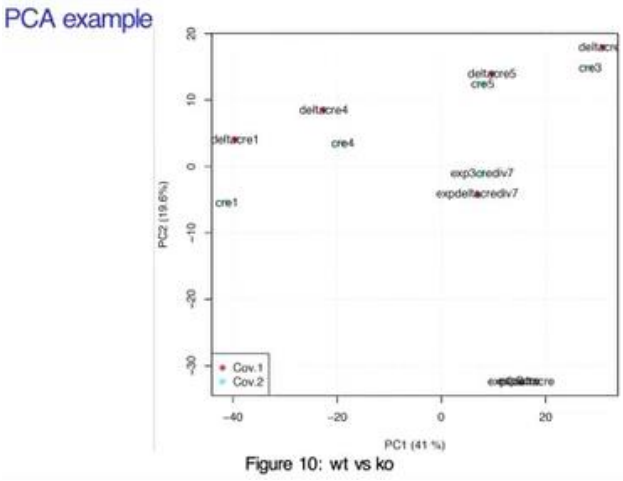


Figure 10: wt vs ko

There are 6 transgenic mice: 1 is the WT, while the others are KD for a gene. In this example, I look at the transcription profile of fibroblasts in the 2 conditions. Since this is an inducible KD, the procedure is done on the same cell: given a fibroblast, I can choose to maintain its WT phenotype or to switch-off a gene. In case of deltacre1 and cre1 (the 1st mouse) and deltacre4 and cre4 (the 4th mouse), their dimension is mainly given by PC2. In mouse 5, there is a smaller difference. There is a mouse (present in the bottom left part of the graph) in which the treatment and the non-treatment do not change the transcription profile: the system did not work → the gene has not been knocked down. What I can learn from this experiment is that the animal mainly drives most of the variance of the experiment. The difference that exists between the animals is mainly characterized by the 2nd dimension. As the procedure did not go well in a sample, its removal may increase the variance among all the others. This information is very important: when I do a differential expression analysis between WT and KD samples, I won't obtain most of differentially expressed genes because most of the variance is given by the difference among animals → the variance among specimens masks the real differences I want to look for. However, since I know that WT and KD animal are different, I can build a specific type of differential expression analysis, in which I block the group of the animal: each animal is allowed to represent a block of the experiment. The 1st step that people perform during statistical analyses is to remove the variance between the blocks, so that the final variance that I get will truly express the difference between treated and untreated samples. Blocking can be done only if I see, by PCA, the overall features of my experiment and to appreciate the differences between the 2 groups.

PCA example #2, representing what I should not get when I perform PCA

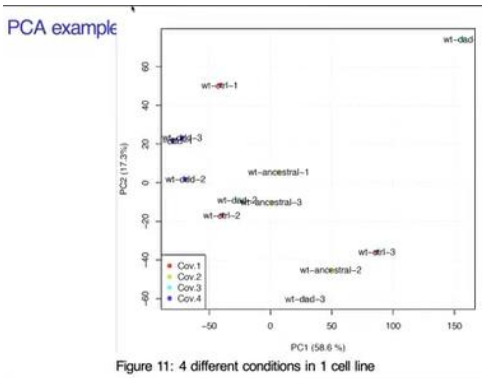


Figure 11: 4 different conditions in 1 cell line

In this experiment there is a cell line and 4 treatments: there is the WT, the control (untreated) and different treatments. There is no way to separate the control from the treated samples by PCA, while in the previous examples I can do it. The aim of PCA is to draw a line that separates the control from the treatment. In this example, there is a problem with my experiment because I cannot differentiate the 2 conditions. In fact, researchers obtained a transfection efficiency of 30%. Moreover, when the differences between 2 samples or conditions is very tiny (as it happens for the 5th mouse in the previous example), this is due to the fact that PCA analyses ALL the genes present in a sample: if the differences are due to some genes, the 2 samples will show small differences.

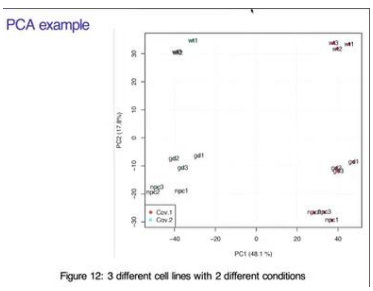
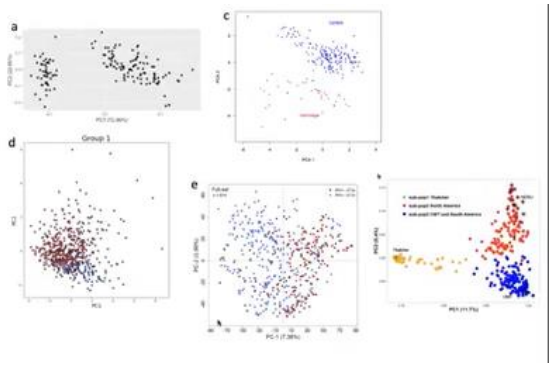


Figure 12: 3 different cell lines with 2 different conditions

PCA example #3, an ideal example of what I am expecting

In this care, 2 cell lines are treated in the same way. On the first component there is a difference between cell line 1 and cell line 2; looking at cell line 1, there is a clear separation in 2 groups in PC2. I can identify the genes that differ between the WT, independently from the cell line; moreover, I can analyse the differences between the 2 treatments and the WT. in this example I have a clear separation between the various segments (there is a

difference of 48.1% between the 2 cell lines and a difference of 17% in the same cell line). In order to identify the genes that change during the treatment, I can make a block: cell line A treated and untreated; cell line B treated and untreated →if I mix the 2 cell lines, I can detect which are the differentially expressed genes, independently from the cell line. Another thing I can do is to compare the WT and treated cell line 1 with WT 2 and treated 2, to evaluate the cell line-specific genes.



PCA can clearly tell me if the experiment that I study during my analysis is good or not.

- Panel a: the line can be drawn very well in PC1
- Panel c: the line can be drawn on the PC2: there is a big variance among the various elements of the 2 experiments, and the difference between them is smaller than the overall difference existing between the various elements that are representing the 2 datasets that I compare.
- Panel d: the probability to find a difference between the two groups is very small: the noise is so big that it masks the differences, or there are no differences at the transcriptional level.
- Panel b: the yellow sample is different from the other two, and the separation can be done on PC1; for the other ones, the separation is done on PC2.
- Panel e: this is a noisy sample → drawing the line will imply sample loss, but there is a good number of blue cells that are separated from the other blue ones.

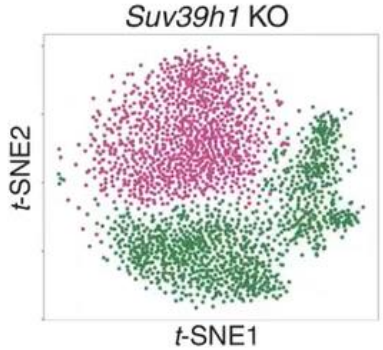
<u>Exercise</u>

The dataset, present on Moodle, contains 3 replicates of the PC9 cell line. Download docker4seq and apply the PCA function. Data should be in a tab-delimited file. Open RStudio and go in the folder where data are located. Apply the pca function in RStudio or, if it does not work, on the docker. The output result is a PDF.

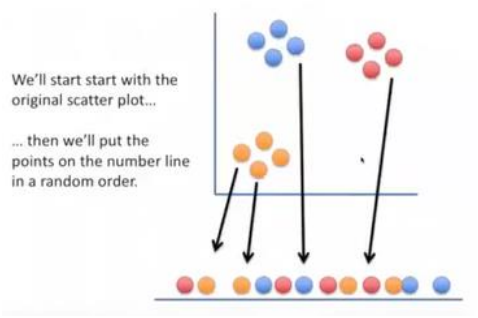
Pca function requires the experiment.table (which is my tab-delimited file), the type (I have to insert "count" in this case), covariatesInNames = T will assign a different colour to each covariate in the final PDF. samplesName = T means that the sample's name is plotted over the points. Principal.components are, typically, 1 and 2. If I don't see a nice separation between the components, I have to try to look at the 2nd and 3rd components. Legend.position allows me to move the sticker in the plot.

<u>tSNE</u>

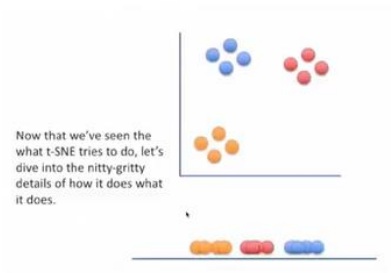
The problem of PCA, which is a linear dimensionality reduction method, is that it is very noisy in sc analyses, because they are 0-inflated and data are less strong → non-linear separation methods are necessary to reduce the complexity of the data. However, the correspondence with the original data is not always maintained. Examples of non-linear separation methods are tSne and umap. They try to represent data in another dimension, in order to make elements much more separated.

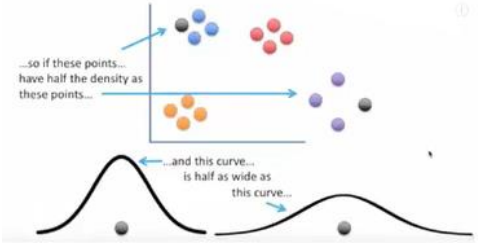


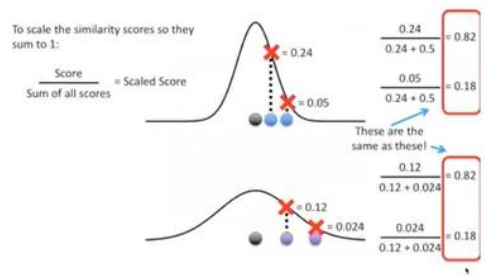
tSNE was not designed for sc, but to handle big data, which are represented in 2 or 3 dimensions. As shown in this image, cells are coloured on the basis of their biological features: violet cells are different from green cells, and they are labelled according to previously known biological features. Data reduction shows that the violet cells are mainly separated from the green cells, which in turn are separated in two other groups (that are slightly defined). Cells are seen in 2D, and the 2 groups can be separated according to the transcriptomic data. As an example, the 2 dimensions are reduced to 1 dimension (that can be seen by me)

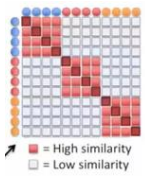


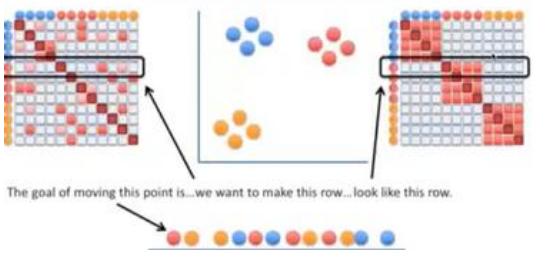
only, to clarify and simply things. I cannot represent the initial structure: the blue, red and yellow points are separated, but I lose the information related to the relative distance among the elements: this representation does not necessary fit the original data. What is the problem when I need to reduce the dimension? If I plot data on the x or y axis, some elements will be separated in a good way, while others will not → how can I obtain a better separation? I start from the situation in which I know what I would like to get, because I know the relationships between cells. then, I take all the cells and put them in one dimension, in a random way. However, as said before, I mathematically know the relation between cells and separation must start from one sample, such as red cells: they have to be close to each other and far from the yellow ones. This separation is not done in one shot, because a red cell is progressively moved towards the other red cells little by little. Now, the yellow cell (which is now the first cell from the left) is investigated: it will move towards the other yellow cells step-by-step. At each step, data re-organize and the final situation resembles the initial one; the separation is maintained according to the space of my dimension. How does this attraction/repulsion work? A cell in the real space is taken as reference (the black one), and its distance from the other cells is measured on the 2D graph: my cell is represented by its distance against all the other cells. To do so, I "create" a normal distribution, whose center is represented by the expression level of my cell. Then, all the other cells will be positioned according to their distance from the black one: cells belonging to the same group of the reference one will be close to it and to the center of the curve, while the far cells will be near the extremities of the curve. [general idea: I project the distance with respect to the cell reference, to calculate it.] At the end I'm going to have a representation in which all the distances between the reference cell and the other ones will reflect the distances between near cells (which stay near to the mean) and the far cells (which distribute on the tails of the curve). Why do I do this procedure? Because I need to normalize data and represent them in a normal distribution, always basing it on the real values (that I don't see but mathematically calculate). What is the problem of this procedure? The blue cluster is very dense, and the final curve will be narrower than the distribution obtained when cells belonging to the same cluster are far from each other. However, this scheme is not good: I have to compare distributions, so I have to normalize data. The easy way to normalize data is the scaling: I take each score representing the distance, and divide it to the sum of the score = scaled score; the sum of all the scaled scores will be equal to 1. After these calculations, all the distributions will be the same and, although the initial values are different, they will equal each other after the normalization. This approach allows to easily compare different groups characterized by cells with a different distance among each other. At the end, I obtain a square similarity matrix, which gives me information about the similarity present among different samples. The diagonal is the area characterized by the highest similarity between elements. However, I still cannot represent data in a smaller space scale → I put cells in the only dimension in a random way and I measure the distances between the 1st and the 2nd, the 2nd and the 3rd and so on. So, I basically do again all the steps I did previously, but in this case I work with randomly distributed cells. Moreover, instead of using a normal distribution, I use a t-distribution (tSNE), which has a narrower tip of the curve and allows similar



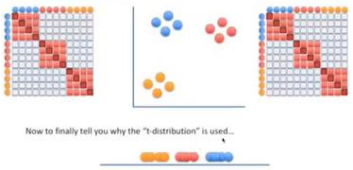








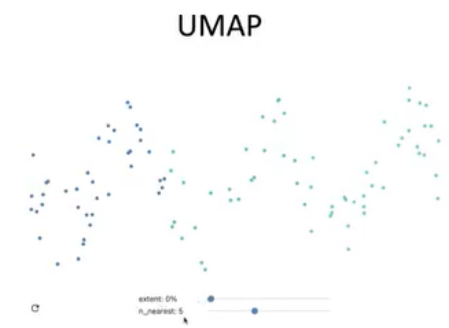
elements to be put closer to each other. So, I project each cell on a t-distribution graph and, at the end, I obtain a correlation matrix. Since the cells have been placed randomly, similarity is not related to the real conditions but only by chance. Now, what does tSNE do? It takes red cells, for example, and moves them closer to the other red cells, progressively checking if the new matrix is similar to the original situation. Then it moves to other cells and does the same step. [overall idea: to make the final configuration as similar as possible to the original one, working step-by-step.]



Dimensionality reduction and data clustering are different concepts. Looking at this condition, cells are divided in 3 groups according to their color; however, this representation is done according to how I see the data, so it is biased. On the other hand, unbiased systems can be used, in order to assign cells to specific groups, characterized by different transcription profiles = clustering. Clustering allows to better appreciate similarities and dissimilarities between cells. At the present time, data reduction is a way to represent data according to **how I SEE them**, but it does not divide cells in clusters. One of the peculiarities of tSNE is that I can assign a preference distance matrix: I decide which kind of distance use to represent data, finally, tSNE will try to reach that representation in the lower dimension.

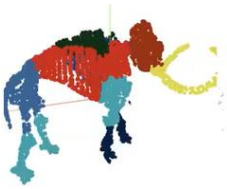
Today there is another method to do so: it is more robust than tSNE and it is UMAP (Uniform Manifold Approximation).

<u>UMAP</u>



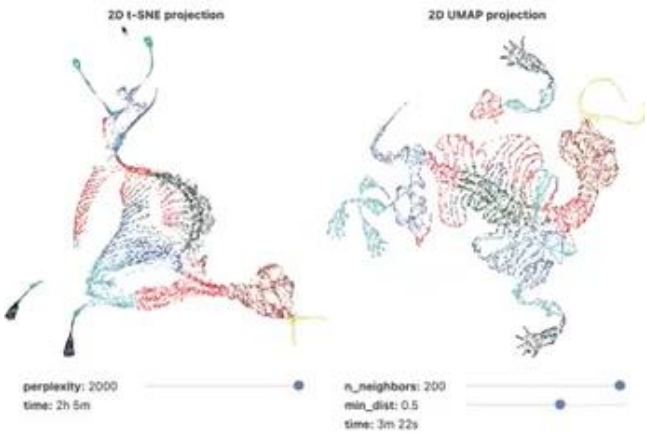
It is another data reduction method, which is preferred because it focuses on “big” pictures (=full representations of data, instead of local ones, which are in contrast investigated by tSNE). UMAP tries to change the structure of data, looking at how they are organized. UMAP defines point aggregations, that are used to reduce the overall single points that I have: if I have many points near to each other, they are aggregated to form a group, called “simplician complex”. This is an example of picture I can get; the lowest cursor (n_nearest) indicates the maximum number of nearest points that can be connected. The “extent” parameter indicates how much I extend the initial point in terms of percentages. Given 4 points, that are all far from each other, I fix a segment that allows me to increase the dimension of the point, until 2 or more points will be connected. On the basis of the distance, I can build a connection network according to how large is the diameter of the circle I build to enlarge the initial point (see next slides to appreciate point connections). In the image, there is a scheme (some wave-like structures are formed) which is progressively lost when the diameter of the circle increases. According to the values of “extent” and “n_neighbour” parameters, I manage to reproduce this connectivity on the basis of the real points; this connectivity is used to generated point aggregations, which are then reproduce in lower dimensions by softwares like tSNE. The main difference between tSNE and UMAP is that tSNE uses distances that I can design a priori, while UMAP does not allow personal measurements. This approach allows to focus on global pictures, and not on local events.

In the exercise, I will have to play with the “n_neighbours” and “min_dist” parameters to see how the plots change.



This is the 3D representation of a mammut; playing with low values of the UMAP parameters, this scheme is lost, while it is seen again when the parameters increase in value. The task of UMAP is to give an overall picture in a lower dimension, with a similar representation of data respect to the initial one. of data are very different; moreover, tSne requires more time and neighbours than UMAP.

UMAP & tSNE



In order to appreciate the mammut picture also in tSNE, I must use high values of perplexity (= amount of complexity that I would like to visualize), which requires 2.05 hours compared to UMAP, which takes only 22 minutes. The reason why I need a lot of time in the first case is because tSNE tries to optimize local data, while UMAP focuses to the full picture.

On Moodle there is some material useful to perform sc data analysis with UMAP and tSNE. Look at the link to read a story of tSNE and UMAP working and comparisons.

Exercise

Work on RStudio or on the docker. There are 2 small samples of sc, which are already imputed. Convert the numerical data in cpm format (I have to find the function that allows me to do so). Once I have the log2 cpm I can run tSNE and UMAP with min_dist = 1 and a perplexity value that allows tSNE to perform a global analysis. Treatments are performed in vivo, so the changes are less “dramatic” than treatments performed in vitro. Aim: analyse how the pictures change and what are the differences between the Ctrl and the treatment.