MATHEMATICAL STATISTICS AND MACHINE LEARNING FOR LIFE SCIENCES

How to tune hyperparameters of tSNE

Three simple rules to make beautiful tSNE plots

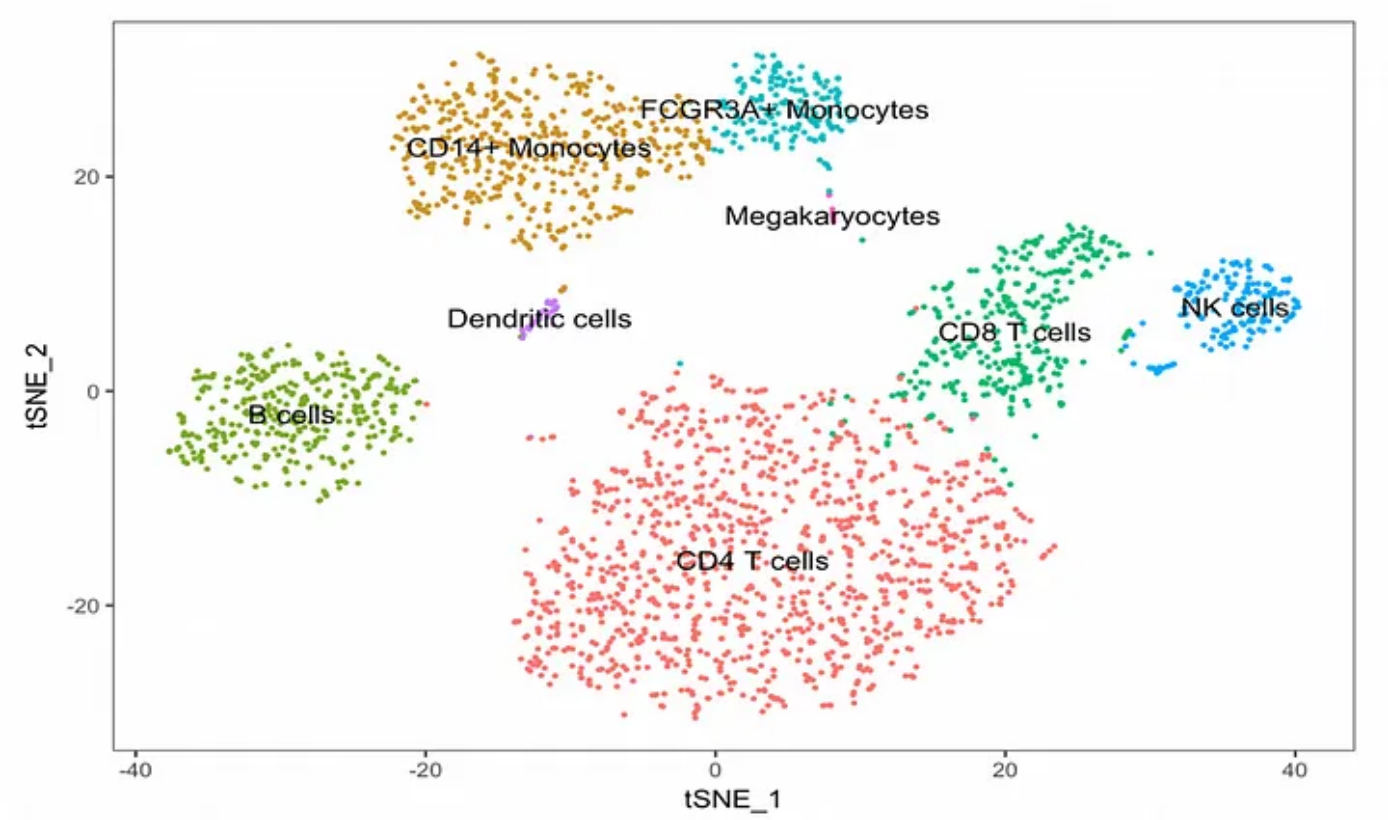


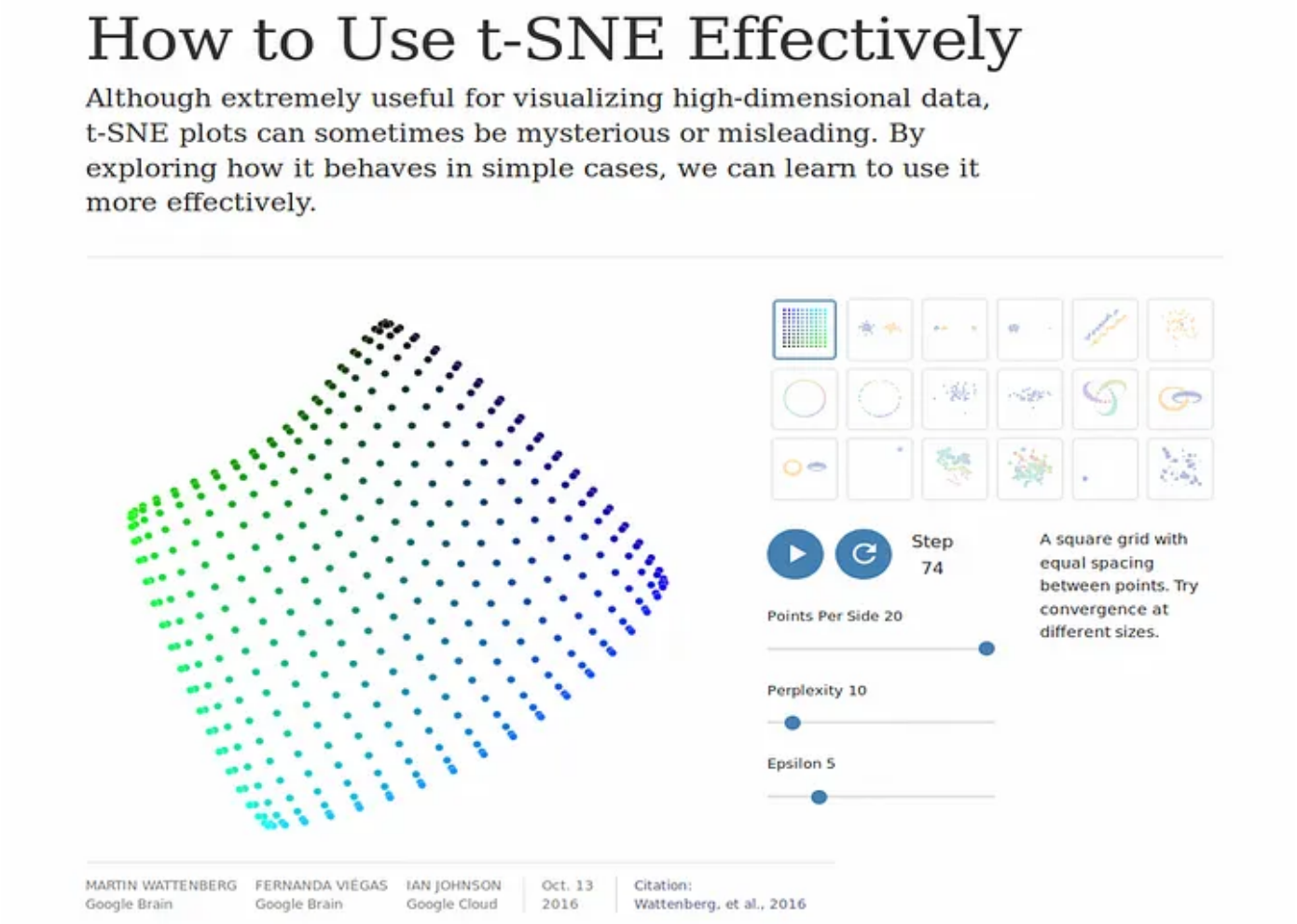
Image source

This is the second post of the column **Mathematical Statistics and Machine Learning for Life Sciences**. In the first post we discussed whether and where in Life Sciences we have Big Data suitable for Machine / Deep Learning, and emphasized that Single Cell is one of the most promising Big Data resources. **t-distributed stochastic neighbor embedding (tSNE)** is a Machine Learning non-linear dimensionality reduction technique which is absolutely central for Single Cell data analysis. However, the choice of hyperparameters for the tSNE might be confusing for beginners.

In this post, I will share my recommendations on selecting optimal values of hyperparameters such as **perplexity**, **number of principal components** to keep, and **number of iterations** for running tSNE.

How to Use tSNE Effectively

When teaching single cell RNA sequencing (scRNAseq) course I keep getting questions about **sensitivity of tSNE with respect to hyperparameters** such as perplexity. The questions are usually inspired by this fantastic post about challenges with interpreting tSNE plots.



A popular tutorial on developing intuition behind tSNE

Despite my great respect for the main message of the post, I think scRNAseq community **should not worry too much about perplexity** and other tSNE hyperparameters based on what they learn from that post because: a) many examples in the post come from abstract mathematical topologies which do not really resemble scRNAseq data, b) the post concentrates on extreme tSNE hyperparameters which are rarely used in the real world scRNAseq analysis.

If you do scRNAseq analysis you will not avoid the popular **Rtsne** function and R package which is based on Barnes-Hut C++ implementation of the original tSNE algorithm. The

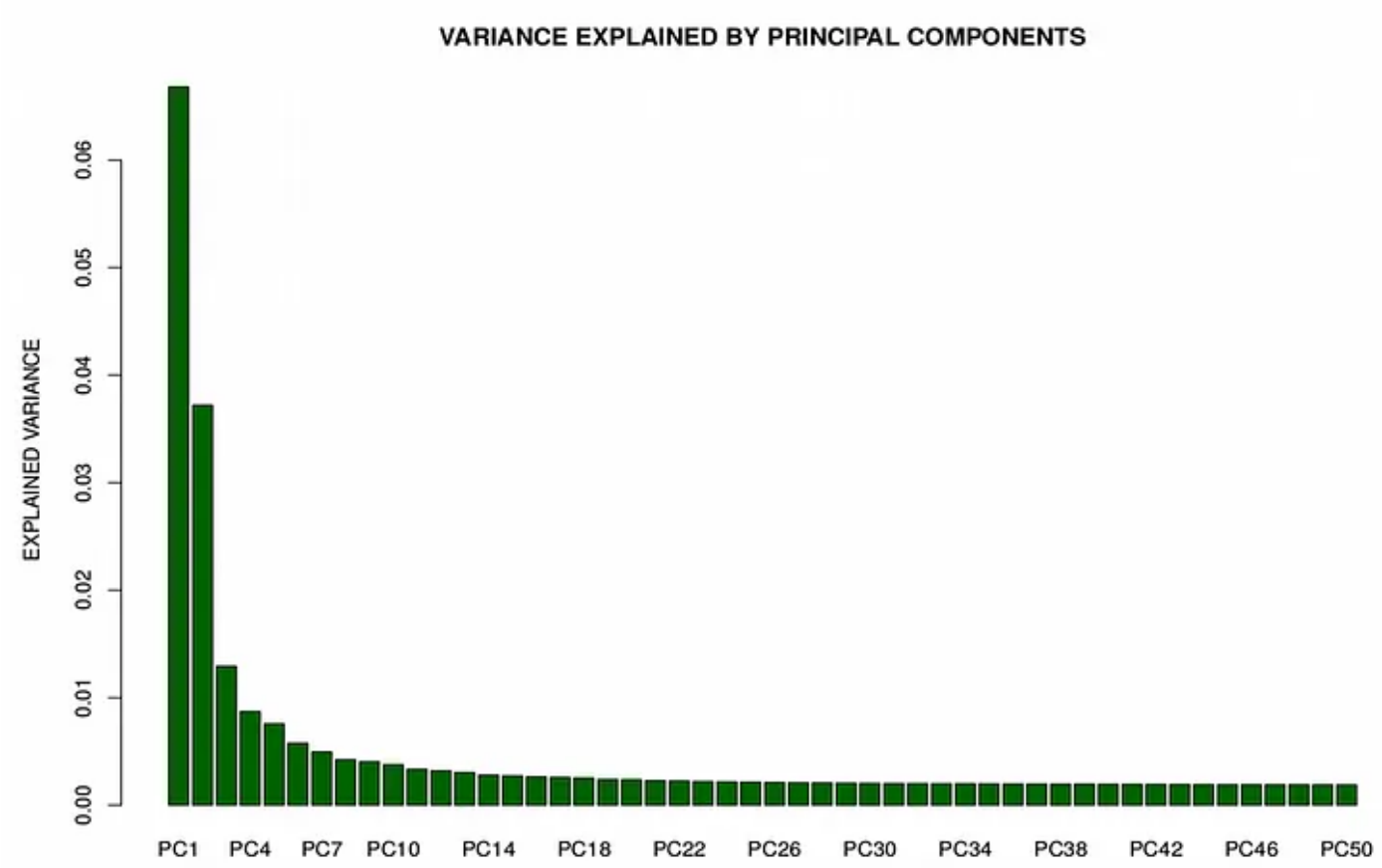
Rtsne function has three main hyperparameters:

1. initial_dims (default 50) providing that pca=TRUE
2. perplexity (default 30)
3. max_iter (default 1000)

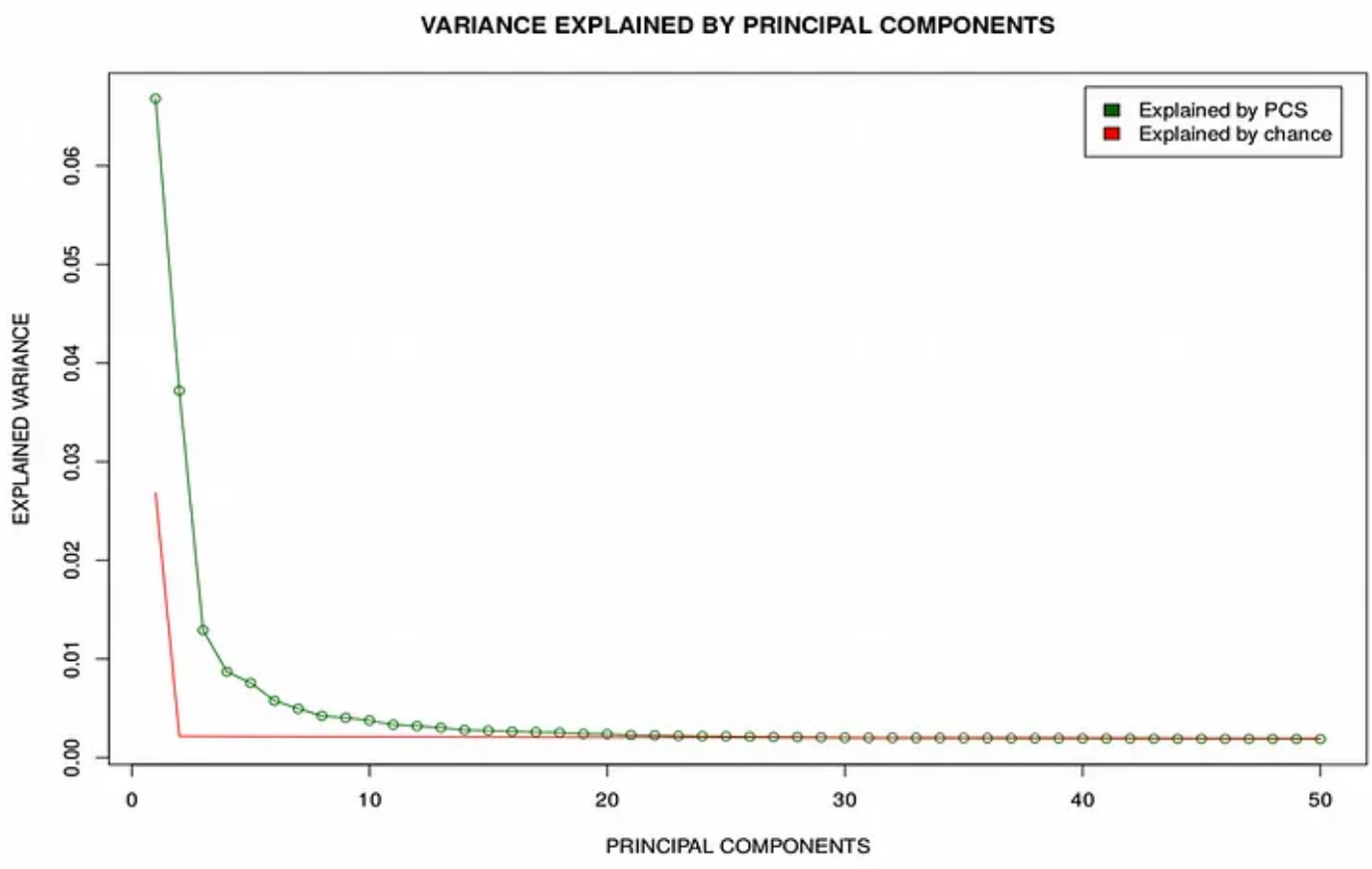
Here, we will go through these hyperparameters and explain what they mean. Obviously, their default values might not work well for arbitrary data. Here, I am going to explain **how to select optimal tSNE hyperparameters for your particular data set** if you are not sure where to start.

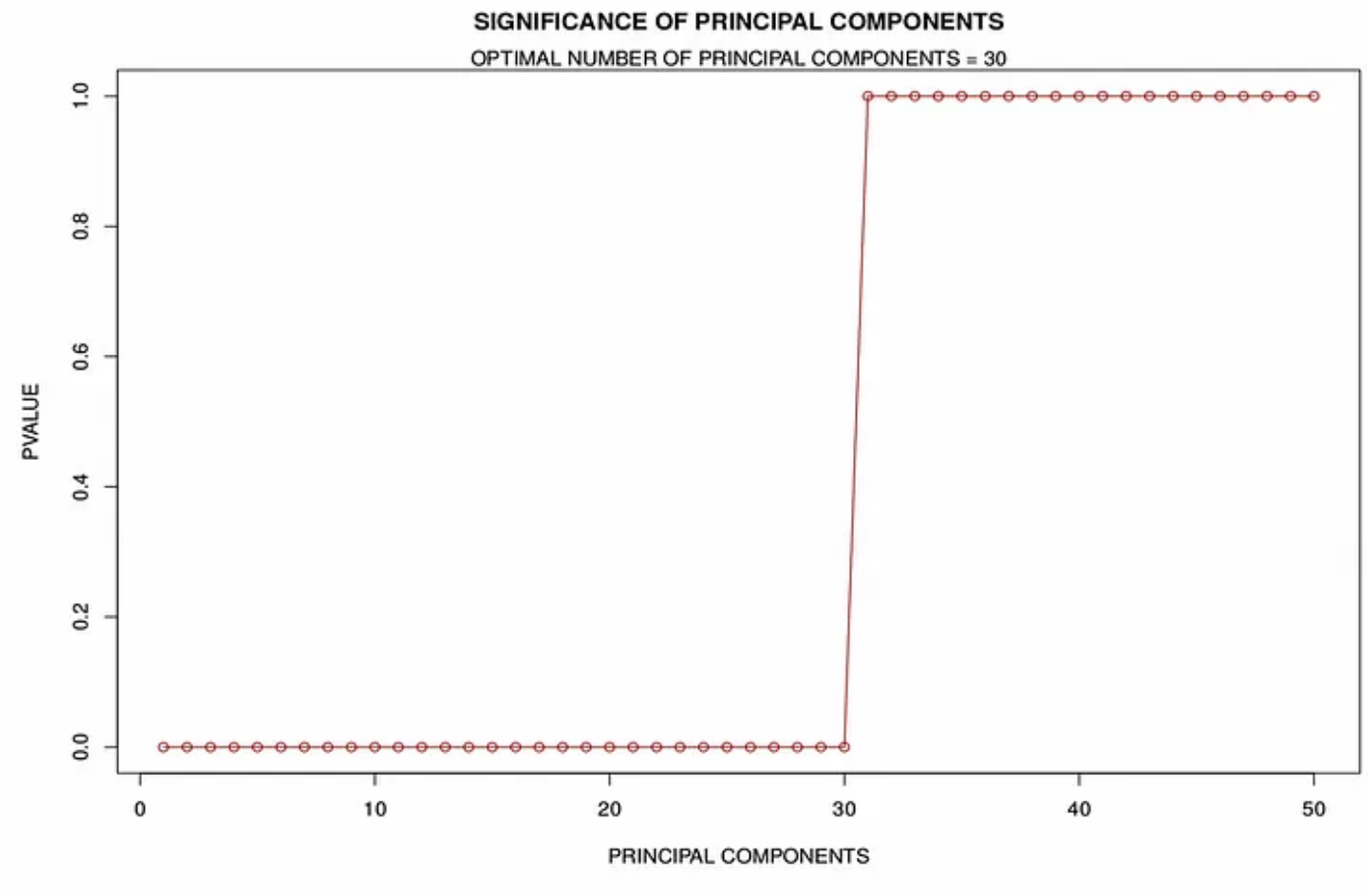
How to select optimal number of PCs?

scRNAseq is a high-dimensional data (~20 000 dimensions / genes) while **tSNE has difficulty dealing with high dimensions**. Therefore, typically you want to reduce initial number of dimensions linearly with e.g. PCA or non-linearly with e.g. Autoencoder (see e.g. here) down to 30 - 50 latent variables (**initial_dims**) and use this as a new data set for feeding into tSNE. Here for simplicity we will use PCA for the pre-dimensionality reduction since PCA is great at separating signal from noise. Let us plot the percentage of variance explained by Principal Components (PCs) using scRNAseq data from Cancer Associated Fibroblasts (CAFs):



Looks familiar, doesn't it? However, how many Principal Components (PCs) should we keep for inputting into tSNE, i.e. what value initial_dims should take: 20, 30, 50 or maybe more? Here we have a dilemma: 1) if we select too many PCs, we will include "noisy" PCs from the tail of the plot, however 2) if we select too few PCs we might loose the signal from the data. To make this decision we recall that randomization is a friend of a Data Scientist and will compare the **observed variance** explained by PCs with **permuted variance**. For this purpose, we shuffle the elements of the expression matrix, perform PCA and check what would the above plot look like for the permuted matrix:


VARIANCE EXPLAINED BY PRINCIPAL COMPONENTS
Explained by PCS
Explained by chance
EXPLAINED VARIANCE
0.00
0.01
0.02
0.03
0.04
0.05
0.06
0
10
20
30
40
50
PRINCIPAL COMPONENTS



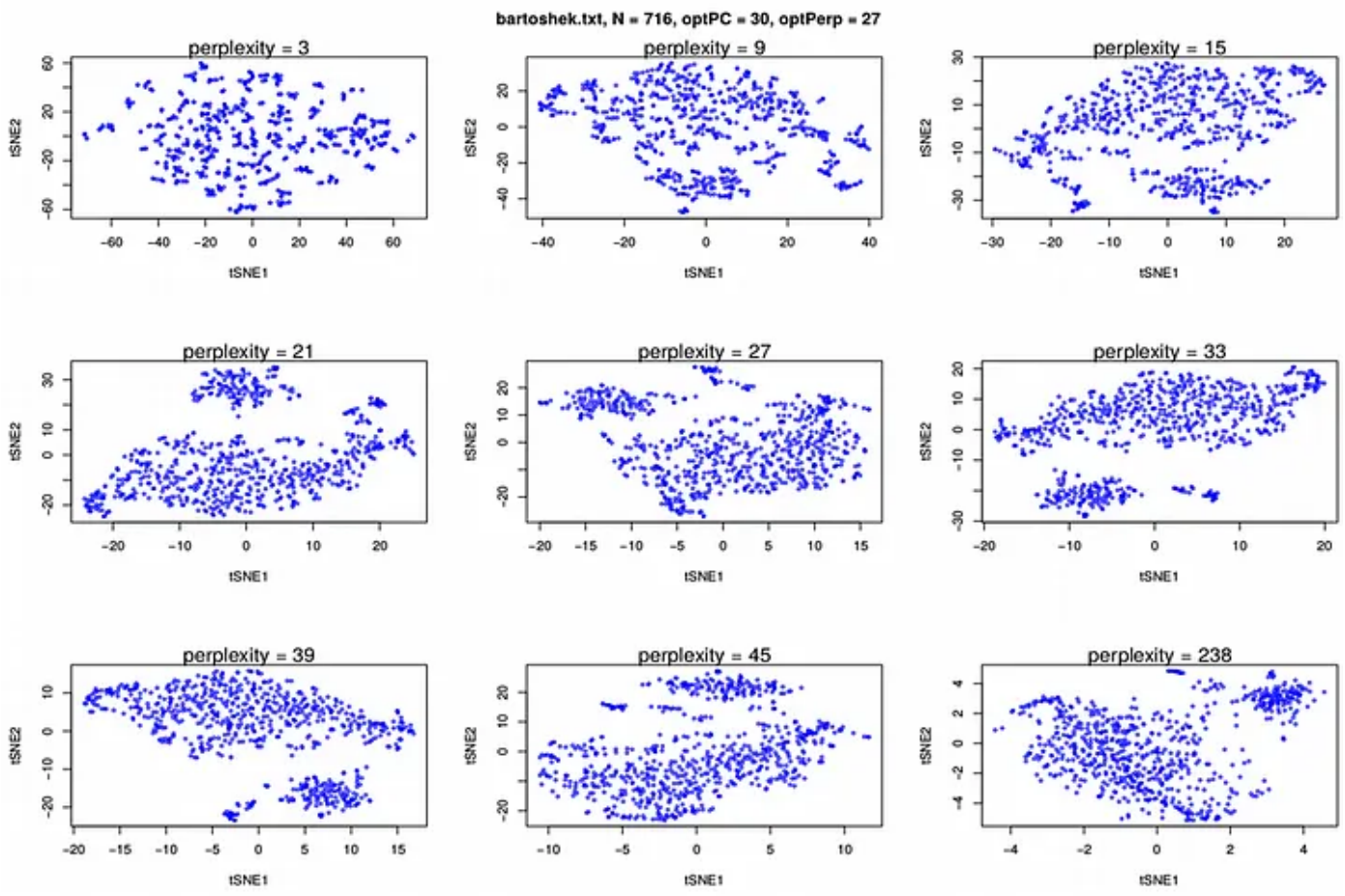
The red curve on the first plot is the mean of the permuted variance explained by PCs, this can be treated as a **"noise zone".** In other words, the point where the observed variance (green curve) hits the permuted variance (red curve) determines how many **informative PCs** we have in our data. Moreover, since we have a vector of permuted variances, it is possible to calculate the p-value of how the observed variance is different from the permuted variance for each PC. For the case of CAFs we conclude that 30 PCs should be kept for the tSNE and the rest should be ignored as their values fall into the "noise zone".

How to select optimal perplexity?

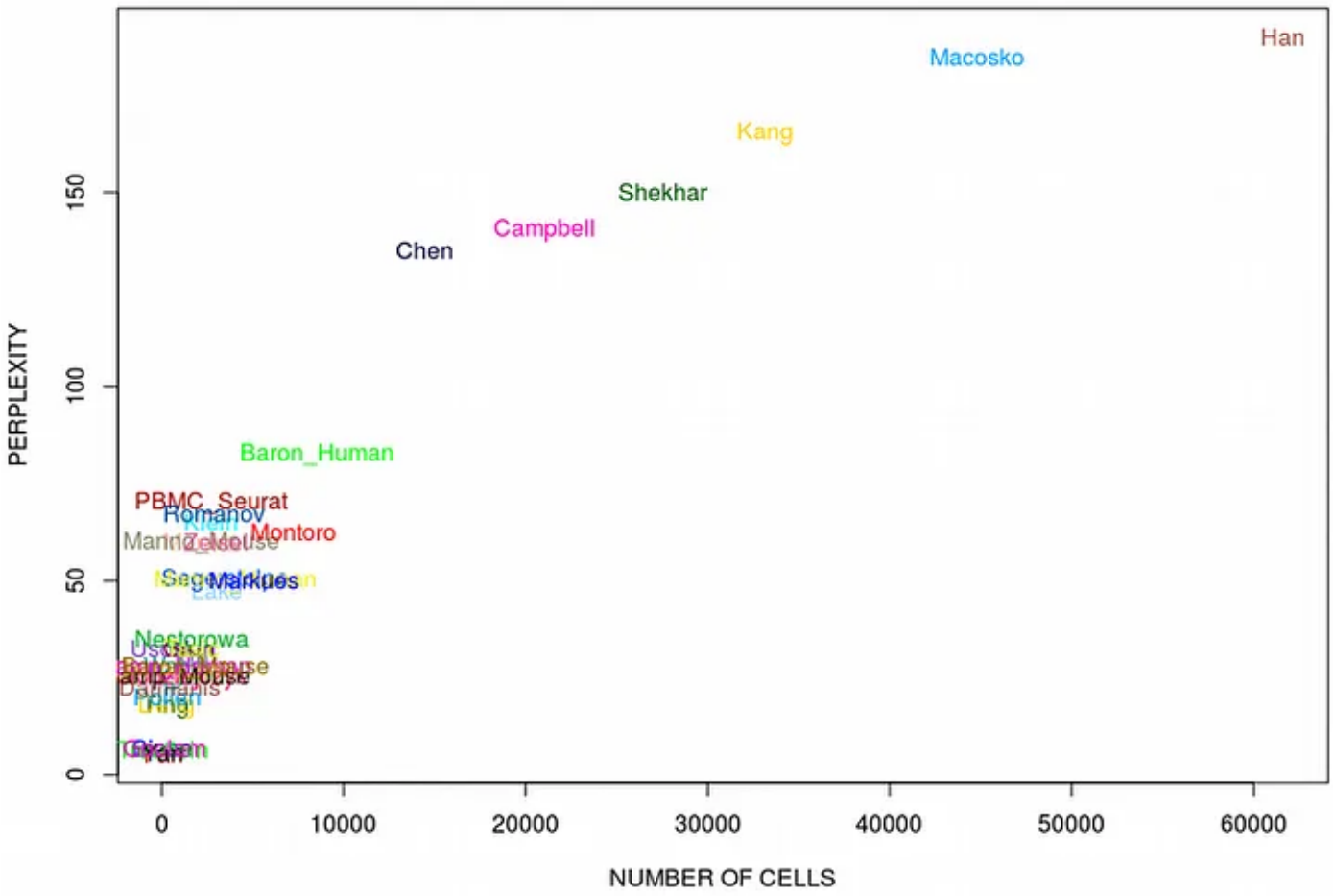
Perplexity is perhaps the most confusing hyperparameter of tSNE. The author of tSNE, **Laurens van der Maaten**, mentions in the FAQ: **Typical values for the perplexity range between 5 and 50**. One obvious questions that comes immediately to my mind: "For how many data points? What if I have 10 000 cells, should I still use perplexity in the range between 5 and 50?". In the next sentence of the FAQ Laurens van der Maaten adds:

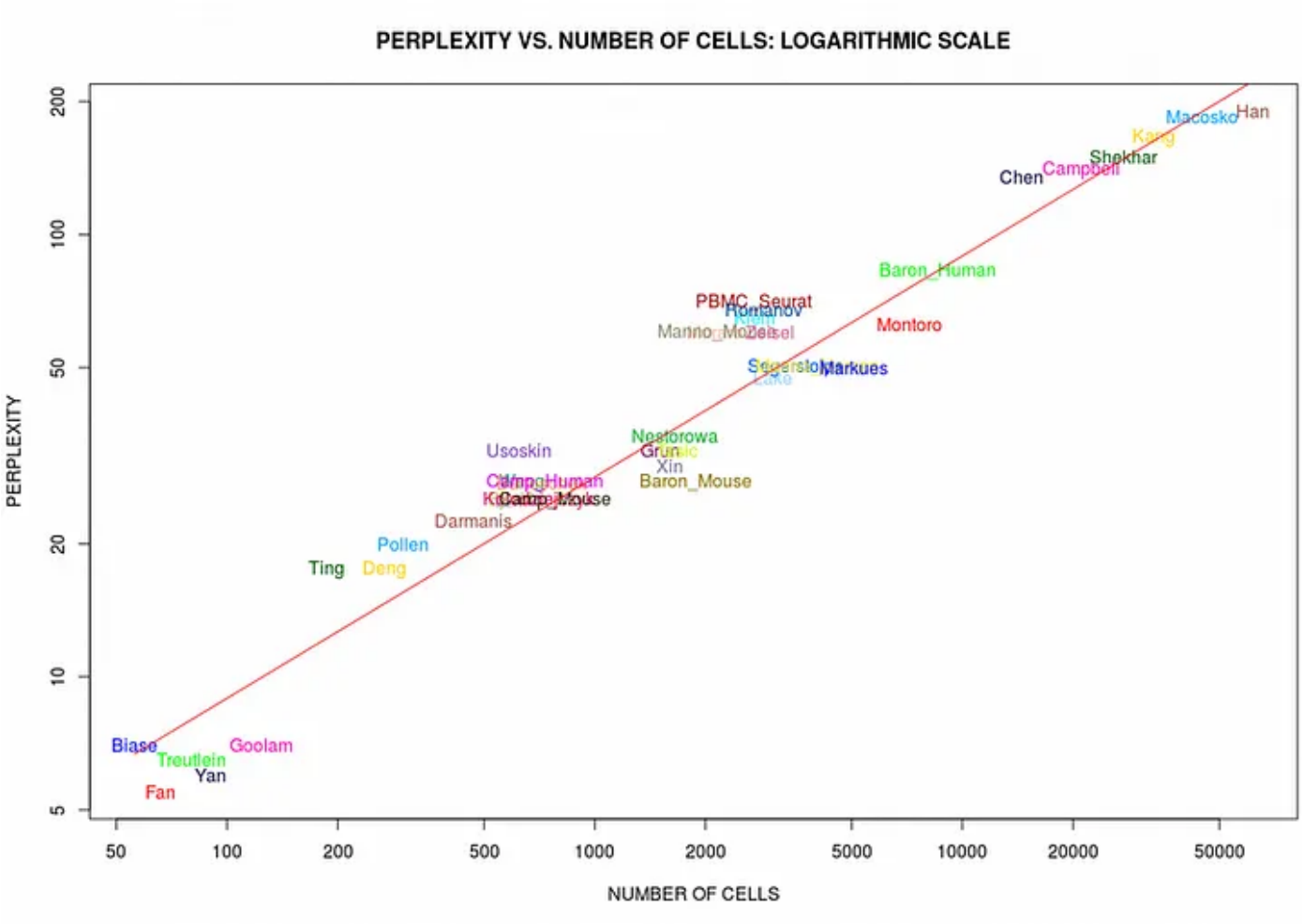
> Loosely speaking, one could say that a larger / denser data set requires a larger perplexity

This sounds reasonable but what should be the functional form of **perplexity vs. number of cells** in order to capture both local and global data structures? To answer this question, one year ago I collected 43 scRNAseq data sets which were publicly available at that time, many of them were downloaded from here. Valentine Svensson has another comprehensive and much more updated list of 500 scRNAseq data sets. For each of the 43 data sets I made a few tSNE plots with perplexities varying from 3 to N/3 (default max perplexity in Rtsne function), N is the number of cells. For the CAFs data set this looked like this:



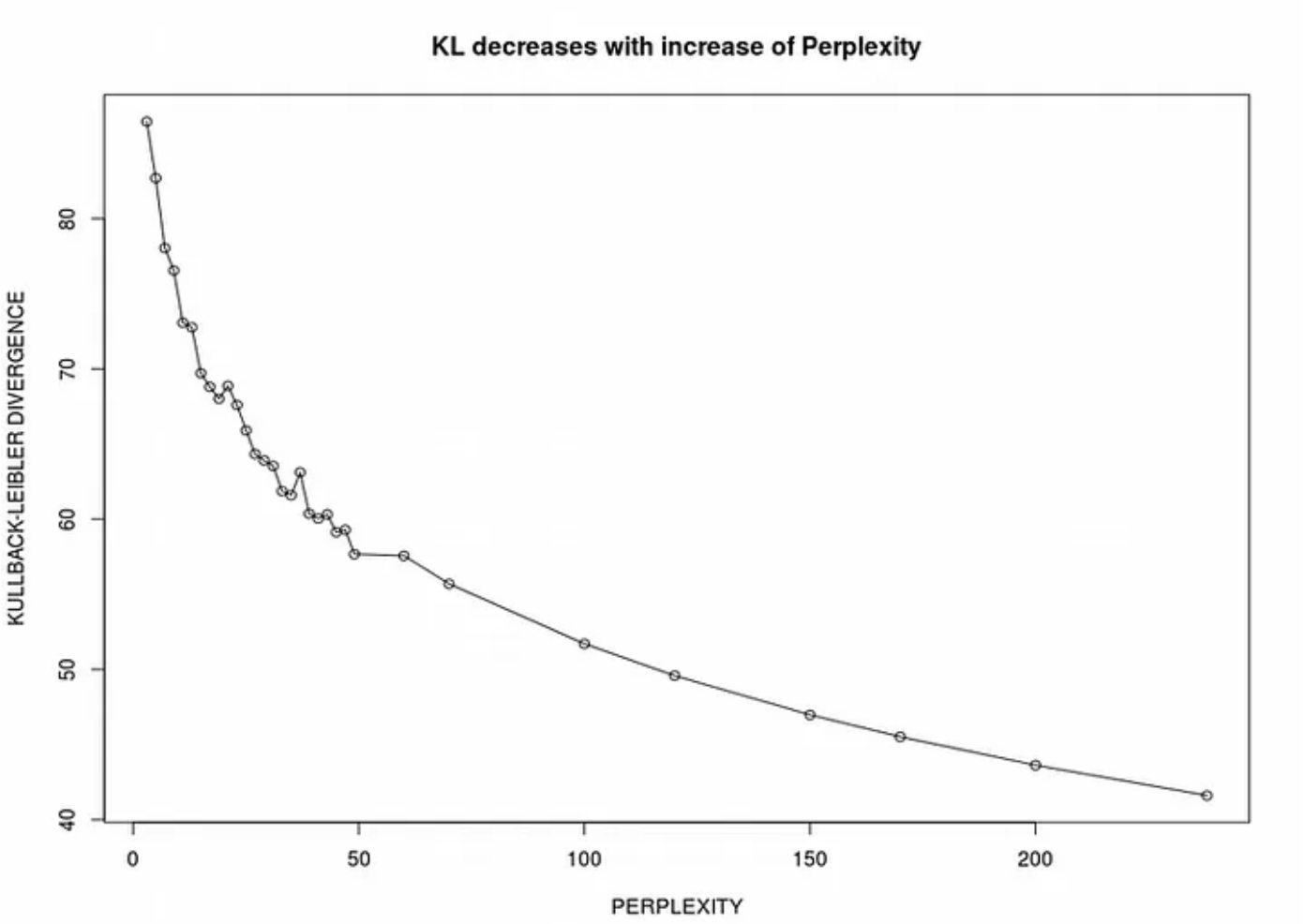
Next, I went through the plots in order to select a range of perplexities where clustering looked most transparent to me, and plotted the mean value of this range of perplexities vs. number of cells for each data set. The curves for the normal (above) and the log-scale (below) looked as follows:


Han
Macosko
Kang
Shekhar
Campbell
Chen
Baron_Human
PBMC_Seurat
Romanov
Montoro
Markus
Nestorowa
Mouse
Yan
PERPLEXITY
0
50
100
150
0
10000
20000
30000
40000
50000
60000
NUMBER OF CELLS



The dependence on the log-scale looks linear, fitting linear model I obtained **log(Perp) = -0.179 + 0.51*log(N)**. Please note the coefficient **1/2** in front of log(N), this implies that the perplexity grows as **Perp ~ N^(1/2)**. Later I realized that this power law is very similar to the rule of thumb for selecting optimal K in the K - Nearest Neighbors (KNN) algorithm. Indeed, in the KNN Machine Learning it is widely accepted (see for example here) that optimal **K~N^(1/2)**. Since the intuition behind perplexity is how many neighbors each data point can "sense", this confirms the power law obtained above.

Now let us try to **analytically derive this power law**. Since tSNE is based on minimization of the Kullback-Leibler (KL) divergence, so maybe the optimal perplexity can be found from the minimum of KL? However, if we plot KL as a function of perplexity at other parameters fixed, it decreases monotonically.



So there is no minimum of KL with respect to perplexity, and KL will always prefer higher perplexities despite we know that too large perplexities will lead to one big clump of points without any clustering. Therefore we need to build another **Score function** that includes the KL and an additional contribution that penalizes KL for too large perplexities. Assuming that the KL behaves as 1/Perplexity, a simple function which always has a minimum would be **Score ~ 1/Perplexity + Perplexity**. However, Perplexity is usually a large number, therefore the Score function will be dominated by the second term. A simple trick to make both contributions to be on the same order of magnitude is to normalize the second term by the number of cells N. Finally, in order to find the minimum of Score we calculate its derivative with respect to Perplexity and equate it to zero. Solving this equation leads to Perplexity ~ N^(1/2).

$$\text{Score} \sim \text{KL} + \text{Penalty}$$

$$\text{Score} \sim \frac{1}{\text{Perplexity}} + \text{Perplexity} \qquad \text{- always has a minimum}$$

$$\text{Score} \sim \frac{1}{\text{Perplexity}} + \frac{\text{Perplexity}}{N} \qquad \text{- apply normalization}$$

$$\frac{d\text{Score}}{d\text{Perplexity}} = -\frac{1}{\text{Perplexity}^2} + \frac{1}{N} = 0$$

$$\implies \text{Perplexity} \sim \text{N}^{\frac{1}{2}}$$

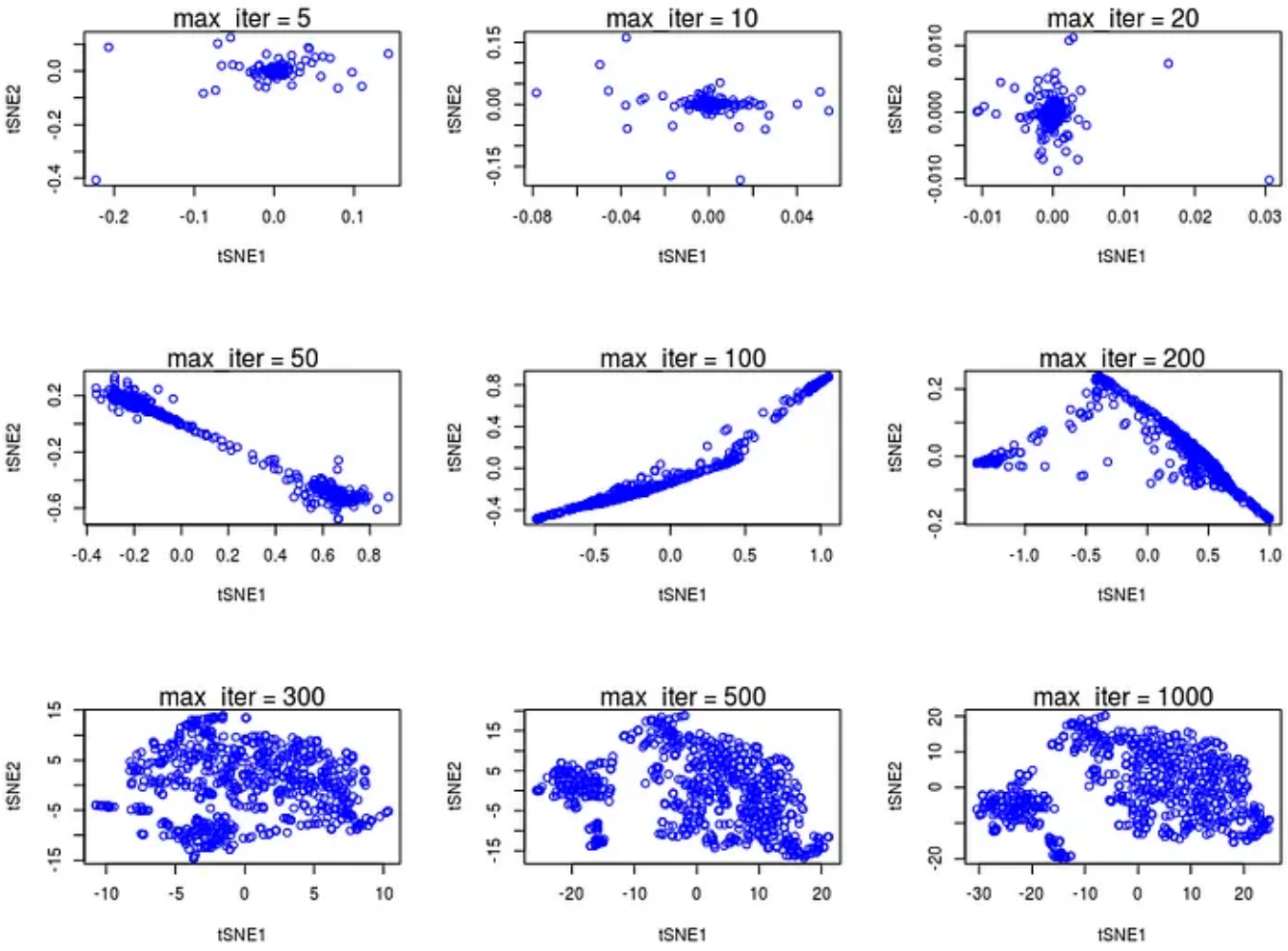
Simple scaling derivation of the power law Perplexity ~ N^(1/2)

Despite the empirical way of deriving the power law, Perplexity ~ N^(1/2), can not be considered as a proper research, it helps **developing an intuition** about the concept of perplexity and its relation to the number of cells. On the other hand, many things about tSNE are based on pure intuition and rules of thumb as tSNE does not have a solid mathematical background in contrast to UMAP. Therefore if you are not sure what perplexity to use for your particular data set, try N^(1/2) and you will not be too off.

How to select optimal number of iterations?

When it comes to the number of iterations needed for tSNE to converge, the simplest recommendation can be **the more iterations the better**. However, practically this is not feasible for big data sets as one might have to wait for days to reach e.g. 10 000 iterations. In contrast, if you use too few iterations the clusters might not be visible and you typically discover a huge clump of data points in the center of your tSNE plot. What to do in this case? Well, if you look carefully at the tSNE plots available in literature, you notice that the **largest distance between data points is on the order of ~100**. This simple rule of thumb indicates that the algorithm reached convergence and further increasing the number of iterations will only marginally change the plot. For the CAFs data set, we can observe how the scale spans only a few units at the beginning of the training and grows up

to ~60 units for larger numbers of iterations such as max_iter = 1000, this also leads to more distinct clustering.



Changes in tSNE plot when increasing the number of iterations

Summary

In this post we have learnt that despite tSNE can be sensitive with respect to its hyperparameters, there are simple rules for obtaining good looking tSNE plots for scRNAseq data. The optimal number of PCs for inputting into tSNE can be found through **randomization** of the expression matrix. The optimal perplexity can be calculated from the number of cells according to the simple power law **Perplexity ~ N^(1/2)**. Finally, the optimal number of iterations should provide the largest distance between the data points of **~100 units**.