The art of using t-SNE for single-cell transcriptomics

Dmitry Kobak[1] and Philipp Berens[1]

[1] *Institute for Ophthalmic Research, University of Tübingen, Germany*

October 25, 2018

Abstract

Single-cell transcriptomics yields ever growing data sets containing RNA expression levels for thousands of genes from up to hundreds of thousands of cells. Common data analysis pipelines include a dimensionality reduction step for visualising the data in two dimensions, most frequently performed using t-distributed stochastic neighbour embedding (t-SNE). It excels at revealing local structure in high-dimensional data, but naive applications often suffer from severe shortcomings, e.g. the global structure of the data is not represented accurately. Here we describe how to circumvent such pitfalls, and explain a protocol for successful exploratory data analysis using t-SNE. They include PCA initialisation, multi-scale similarity kernels, exaggeration, and downsampling-based initialisation for very large data sets. We use published single-cell RNA-seq data sets to demonstrate that this protocol yields superior results compared to the naive application of t-SNE.

1 Introduction

Recent years have seen a rapid growth of interest in single-cell RNA sequencing (scRNA-seq), or single-cell transcriptomics (Sandberg, 2014; Poulin et al., 2016). Through improved experimental techniques it has become possible to obtain gene expression data from tens of thousands of cells using full-length sequencing (Tasic et al., 2017; The Tabula Muris Consortium, 2018) and from hundreds of thousands of cells using tag-based protocols (Zeisel et al., 2018; Han et al., 2018; Saunders et al., 2018). Computational analysis of such data sets often entails unsupervised, exploratory steps including dimensionality reduction for visualisation. To this end, almost every study today is using t-distributed stochastic neighbour embedding, or t-SNE (van der Maaten and Hinton, 2008).

This technique maps a set of high-dimensional points to two dimensions, such that ideally, close neighbours remain close to each other and distant points remain distant. Informally, the algorithm places all points on the 2D plane, initially at random positions, and lets them interact as if they were physical particles. The interaction is governed by two "laws": first, all points are repelled from each other; second, each point is attracted to its nearest neighbours (see Box 1 for a mathematical description). The most important parameter of t-SNE, called *perplexity*, governs how many of its nearest neighbours each point is effectively attracted to. The default value of perplexity in different existing implementations is 30 or 50 and the common wisdom is that "the performance of t-SNE is fairly robust to changes in the perplexity, and typical values are between 5 and 50" (van der Maaten and Hinton, 2008).

Strikingly, when applied to high-dimensional but well-clustered data, t-SNE tends to produce a two-dimensional visualisation with distinctly isolated clusters, which in many cases are in good agreement with the clusters produced by a dedicated clustering algorithm. This attractive property as well as the lack of serious competitors until very recently (see *Comparison with other methods*) made t-SNE the de facto standard for visual exploration of scRNA-seq data. At the same time, t-SNE has well known, but frequently overlooked weaknesses (Wattenberg et al., 2016). Most importantly, it often fails to preserve the global geometry of the data. This means that when t-SNE places cells into several distinct clusters, the relative position of these clusters on the 2D plot is almost arbitrary and depends on random initialisation more than on anything else. While this may not be a problem in some scenarios, scRNA-seq data sets often exhibit biologically meaningful hierarchical structure, e.g. encompass several very different cell classes, each further divided into various types. Typical t-SNE plots do not capture such

global structure, yielding a suboptimal and potentially misleading visualisation. In our experience, the larger the data set, the more severe this problem becomes. Other notable challenges include performing t-SNE visualisations for very large data sets (e.g. a million of cells or more), or mapping cells collected in follow-up experiments onto an existing t-SNE visualisation.

1.1 Development of the protocol

We developed this protocol to guide researchers in how to achieve improved t-SNE visualisations that preserve the global geometry of the data. Our method relies on modifications of the standard pipeline such as providing PCA initialisation, employing multi-scale similarity kernels, or using so called exaggeration, which have been scattered throughout the literature (e.g. Lee et al., 2015; Linderman et al., 2017) and remain poorly known in the transcriptomic community. To demonstrate our protocol we use a published Smart-seq2 data set with 24 000 cells and several UMI-based data sets with up to 1 300 000 cells.

In many challenging cases our pipeline yields visualisations that are better than state of the art. We also describe how to position new cells on an existing t-SNE reference atlas and how to visualise multiple related data sets side by side in a consistent fashion. We focused on single-cell transcriptomics but our recommendations are more generally applicable to any data set that has *hierarchical organisation*, which is often the case e.g. in single-cell flow or mass cytometry (Amir et al., 2013; Unen et al., 2017) or whole-genome sequencing (Li et al., 2017; Diaz-Papkovich et al., 2018).

1.2 Data sets for presentation

We make use of the following publicly available data sets to demonstrate our protocol. All data come from the mouse brain. In all cases, we rely on quality control and clustering performed by the original authors. All cell numbers reported below are after quality control (i.e. after excluding low-quality cells, cell doublets, etc.). We use cluster names and cluster colours from the original publications. The dimensionality in all cases is ∼25 000 (the exact number of genes depends on the used genome annotation), apart from the first data set where it is ∼45 000 (it includes non-coding genes and pseudogenes).

1. Tasic et al. (2017): $N = 23\,822$ cells from primary visual cortex (VISp, 60%) and anterior lateral motor cortex (ALM, 40%). Sequencing protocol: Smart-seq2. Number of clusters: 133.

2. Cadwell et al. (2016): $N = 46$ inhibitory neurons from Layer 1 of visual cortex. Sequencing protocol: Smart-seq2. Two cell classes, identified based on electrophysiology.

3. Tasic et al. (2016): $N = 1\,679$ cells from visual cortex. Sequencing protocol: SMARTer. Number of clusters: 49.

4. Macosko et al. (2015): $N = 44\,808$ from the retina. Sequencing protocol: Drop-seq. Number of clusters: 39.

5. Shekhar et al. (2016): $N = 27\,499$ cells from the retina, predominantly bipolar cells. Sequencing protocol: Drop-seq. Number of clusters: 26 (18 biologically meaningful ones and 8 clusters of cell doublets/contaminants).

6. Harris et al. (2018): $N = 3\,663$ interneurons from area CA1 in hippocampus. Sequencing protocol: 10x Genomics Chromium. Number of clusters: 49.

7. **10x Genomics:** $N = 1\,306\,127$ brain cells from mouse embryo, available at `http://10xgenomics.com`. Sequencing protocol: 10x Genomics Chromium. We use clustering into 39 clusters performed in Wolf et al. (2018).

1.3 Overview of the procedure

Our pipeline for performing t-SNE while preserving global geometry consists of several steps. First, we perform data pre-processing consisting of library size normalisation, feature selection, nonlinear transformation, and dimensionality reduction using principal component analysis (PCA). Next, we inspect the

Box 1: The t-SNE algorithm The t-SNE algorithm (van der Maaten and Hinton, 2008) is based on the SNE framework (Hinton and Roweis, 2003). For any given point i, SNE introduces a notion of directional similarity of point j to point i,

$$p_{j|i} = \frac{\exp(-\|\mathbf{x}_i - \mathbf{x}_j\|^2/2\sigma_i^2)}{\sum_{k\neq i}\exp(-\|\mathbf{x}_i - \mathbf{x}_k\|^2/2\sigma_i^2)},$$

defining a probability distribution over all points $j \neq i$ (all $p_{i|i}$ are set to zero). The variance of the Gaussian kernel σ_i^2 is chosen such that the perplexity of this probability distribution

$$\exp\Big(-\ln(2)\cdot\sum_{j\neq i} p_{j|i}\log_2 p_{j|i}\Big)$$

has some pre-specified value. The larger the perplexity, the larger the variance of the kernel, with the largest possible perplexity value equal to $N-1$ corresponding to $\sigma_i^2=\infty$ and the uniform probability distribution (N is the number of points in the data set). Importantly, for any given perplexity value P, all but $\sim P$ nearest neighbours of point i will have $p_{j|i}$ very close to zero. For mathematical and computational convenience, *symmetric SNE* defines undirectional similarities

$$p_{ij} = \frac{p_{i|j}+p_{j|i}}{2N}$$

such that $\sum_{i,j} p_{ij} = 1$, i.e. this is a valid probability distribution on the set of all pairs (i,j).

The main idea of SNE and its modifications is to arrange the N points in a low-dimensional space such that the similarities q_{ij} between low-dimensional points match p_{ij} as close as possible in terms of the Kullback-Leibler divergence. The loss function is thus

$$\mathcal{L} = \sum_{i,j} p_{ij}\log\frac{p_{ij}}{q_{ij}}.$$

The main idea of t-SNE was to use a t-distribution with one degree of freedom (also known as Cauchy distribution) as the low-dimensional similarity kernel:

$$q_{ij} = \frac{w_{ij}}{Z}, \quad w_{ij} = \frac{1}{1+\|\mathbf{y}_i - \mathbf{y}_j\|^2}, \quad Z = \sum_{k\neq l} w_{kl},$$

where $\mathbf{y}_i$ are low-dimensional coordinates (and $q_{ii}=0$). As a matter of definition, we consider any method that uses the t-distribution as the output kernel and Kullback-Leibler divergence as the loss function to be "t-SNE"; similarities $p_{j|i}$ can in principle be computed using non-Euclidean distances instead of $\|\mathbf{x}_i - \mathbf{x}_j\|$ or can use non-perplexity-based calibrations.

To justify our intuitive explanation in terms of attractive and repulsive forces, we can rewrite the loss function as follows:

$$\mathcal{L} = \sum_{i,j} p_{ij}\log\frac{p_{ij}}{q_{ij}} = \text{const} - \sum_{i,j} p_{ij}\log q_{ij},$$

and dropping the constant,

$$-\sum_{i,j} p_{ij}\log\frac{w_{ij}}{Z} = -\sum_{i,j} p_{ij}\log w_{ij} + \sum_{i,j} p_{ij}\log Z = -\sum_{i,j} p_{ij}\log w_{ij} + \log\sum_{i,j} w_{ij}.$$

To minimise $\mathcal{L}$, the first sum should be as large possible, which means large w_{ij}, i.e. small $\|\mathbf{y}_i - \mathbf{y}_j\|$, meaning an attractive force between points i and j whenever $p_{ij}\neq 0$. At the same time, the second term should be as small as possible, meaning small w_{ij} and a repulsive force between any two points i and j, independent of the value of p_{ij}.

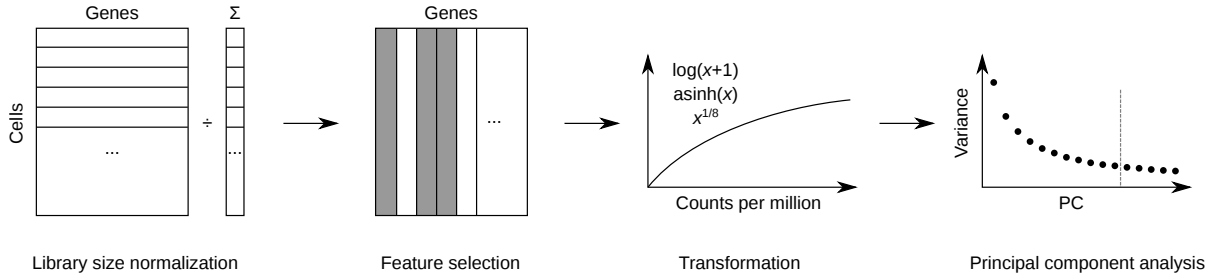

Figure 1: Our pre-processing pipeline for single-cell transcriptomics: library size normalisation, followed by feature selection, followed by log-like transformation, followed by principal component analysis (PCA). See text for details.

leading principal components for evidence of global structure, and perform t-SNE with various perplexity settings to probe the structure at different scales. If clustering results are available, we recommend further exploration via PCA or multidimensional scaling (MDS) on cluster means. We then initialise t-SNE with PCA coordinates and use a multi-scale approach combining different perplexities to obtain a visualisation yielding a good representation of both, the global and the local structure. If the data set is very large, additional modifications such as initial downsampling and exaggeration may be required. Optionally, we show how to create aligned visualisations of related data sets, or how to map additional cells onto an existing visualisation. We first describe and validate each of these steps using example data sets, before providing a step-by-step protocol.

1.4 Pre-processing of the data

We used a standard pre-processing pipeline consisting of the following steps (Figure 1, see Methods for details): (i) library size normalisation; (ii) feature selection; (iii) log-transformation; (iv) PCA. Specifically, we normalised the counts to counts per million (CPM), selected 1000–3000 most variable genes using dropout-based feature selection similar to the one suggested in Andrews and Hemberg (2018), applied $\log_2(x+1)$ transform, and finally did PCA retaining the 50 leading PCs. We experimented with modifying and omitting any of these steps. Our experiments showed that log-transformation (or a similar nonlinear transformation) and feature selection are the two most important steps for good t-SNE results (Figure 9). Library size normalisation was less important. PCA mainly improves computational efficiency as it reduces the dimensionality and size of the data set before running t-SNE (see Methods for further discussion).

In general, inspecting initial t-SNE results often allows to discover small outlying clusters of cell doublets or other experimental artefacts. We recommend discarding such cells and re-running the analysis: otherwise such artefact clusters can create "bridges" between unrelated t-SNE clusters and spoil the visualisation. All data sets shown in this Protocol have already passed quality control by the original authors, so we omit this step.

1.5 Preserving global geometry with t-SNE for medium sized data sets

To explore the global structure of the Tasic et al. (2017) data set, we performed PCA. In a plot of its first and second principal components, three well-separated clusters are apparent (Figure 2a), corresponding to excitatory neurons (cold colours), inhibitory neurons (warm colours), and non-neural cells such as astrocytes or microglia (grey/brown colours). Iteratively performing PCA of these three clusters separately revealed further structure inside each of them: e.g. inhibitory neurons are well separated into two groups, *Pvalb/SSt*-expressing and *Vip*-expressing (Figure 2a, inset). This demonstrates the hierarchical organisation of the data that any successful visualisation should capture.

Another way to look at the global geometry of this data set is to perform multidimensional scaling (MDS) on the set of 133 cluster means (Figure 2b). Here again we see the same three groups (interestingly, MDS highlights that non-neural clusters are very different from each other). Unlike PCA on the full data set, PCA or MDS on the cluster means are not influenced by the relative abundances of different clusters and thus can more faithfully represent the relationships between the clusters. However, this

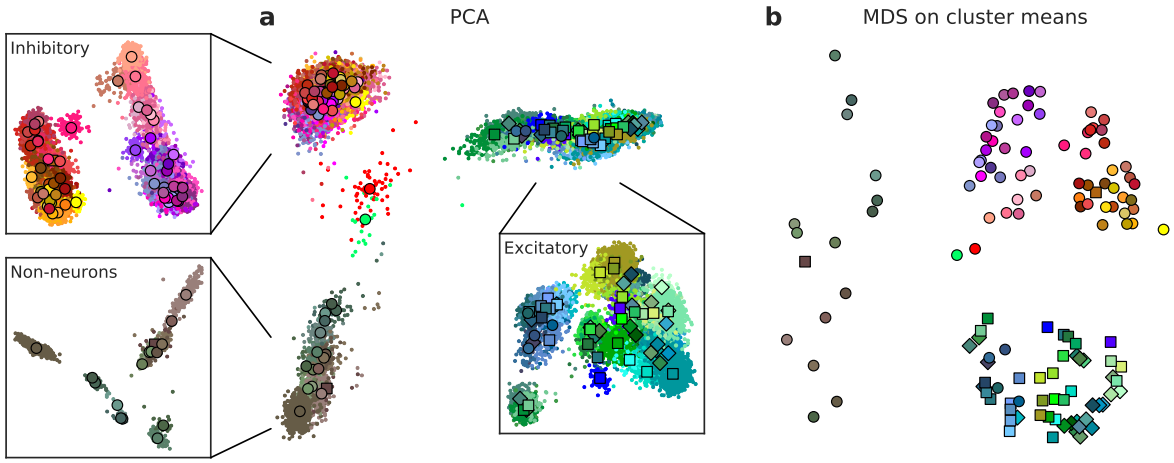

Figure 2: **(a)** PC1 vs PC2 of the Tasic et al. data set. Points show individual cells, colours correspond to clusters. Same clusters and cluster colours as in Tasic et al. (2017). Squares/rhombi/circles show cluster centroids for clusters having mostly VISp cells (squares), mostly ALM cells (rhombi), and cells from both areas (circles). Warm colours correspond to inhibitory neurons, cold colours correspond to excitatory neurons, brown/grey colours correspond to non-neural cells. Insets show first two PCs of these three groups of cells. **(b)** Multidimensional scaling (MDS) on all 133 cluster means.

requires knowledge of the cluster structure. As we are primarily interested in exploratory analysis that does not impose a specific clustering result on the visualisation, we will, from now on, not use cluster labels for any analysis.

The observed global structure is clearly missing from a standard t-SNE visualisation of the same data set, performed with the default perplexity 50 (Figure 3a). This figure clearly illustrates how t-SNE is able to find the local structure (cells are split into multiple isolated groups that correspond well to the clustering results), but fails to represent the global structure: excitatory neurons, inhibitory neurons, and non-neural cells are each split into multiple groups which are shuffled among each other. For example, the group of purple clusters is separated from a group of salmon clusters (both inhibitory neurons) by some excitatory clusters, which clearly misrepresents the global topology of cell types. This outcome is not a strawman: even though Tasic et al. (2017) do not show a t-SNE visualisation in their paper, the accompanying section of the Allen brain atlas (`http://celltypes.brain-map.org/rnaseq/mouse`) features a t-SNE figure qualitatively similar to our visualisation (Figure 3a). Perplexity values in the common range (e.g. 20, 50, 80) yield similar results, confirming that t-SNE is not very sensitive to the exact value of perplexity.

In contrast, extreme values of perplexity yield substantially different output (Figures 3b and c for perplexity 5 and 500, respectively; obtained using the same random initialisation as in Figure 3a). Small perplexity means weaker attractive forces; this lets the repulsive forces dominate allowing all clusters to "blow up" from inside and coalesce together into a group of adjacent soap bubbles. This further exacerbates the problem of misrepresenting global structure: some groups of related clusters that were placed close together with perplexity 50 (Figure 3a) get separated into different regions with perplexity 5 (Figure 3b, black arrows highlight one example). In contrast, large perplexity means stronger and more long-ranging attractive forces; this loses finer detail but pulls larger structures together (Figure 3c). In fact, perplexity 500 yields a surprisingly good visualisation.

These examples show that smaller perplexity uncovers more local structure while larger perplexity shows more global structure. This observation suggests several ways to obtain the desired visualisation with accurate local and correct global structure. First, we can use the t-SNE output with large perplexity as initialisation for t-SNE with smaller perplexity (this process can be called *perplexity annealing*) (Lee et al., 2015). For example, we performed t-SNE with perplexity 50 using the t-SNE result with perplexity 500 (Figure 3c) as initialisation. It differs from Figure 3a only in initialisation, but preserves the global structure substantially better (Figure 3d): each of the three major groups of cells occupies a separate region without intermixing.

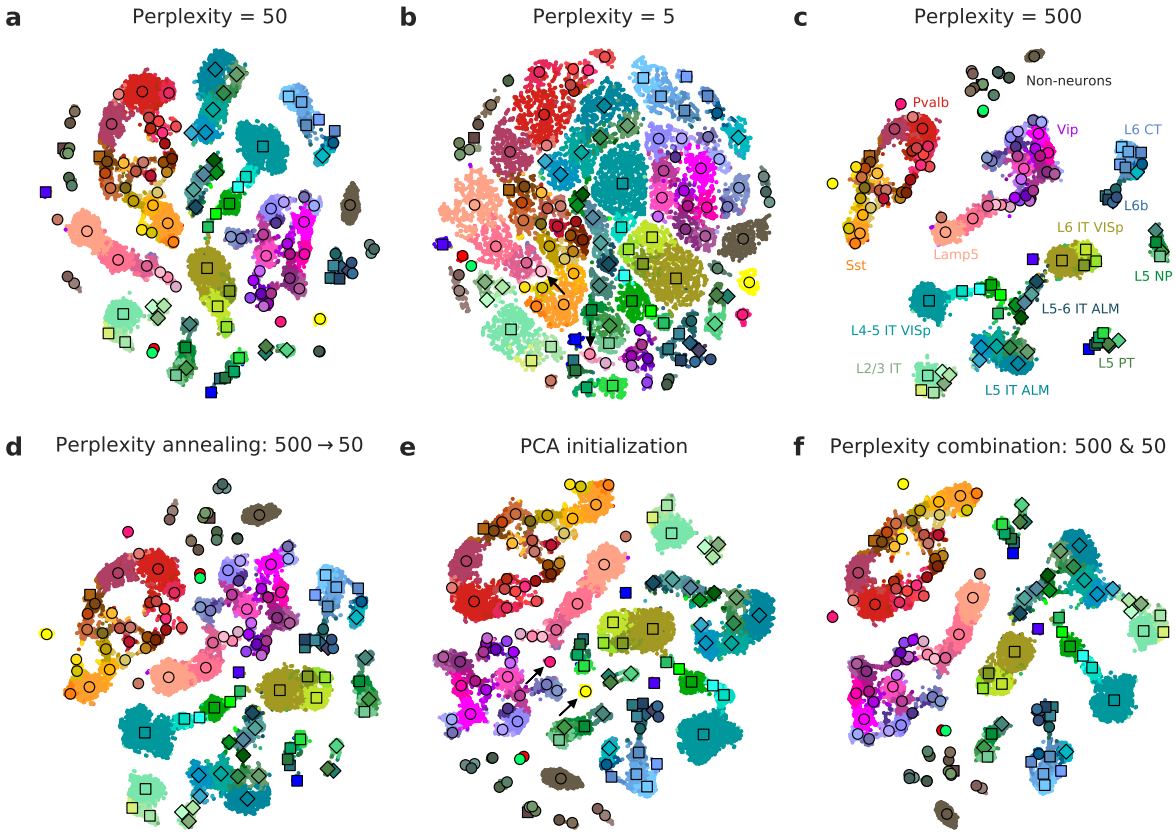

Figure 3: **(a)** t-SNE visualisation of the Tasic et al. data set, colours and shapes as in Figure 2a. Random initialisation, perplexity 50. **(b)** Same initialisation as in (a), perplexity 5. Black arrows highlight two clusters that were close in panel (a) but got separated in (b). **(c)** Same initialisation as in (a), perplexity 500. **(d)** Perplexity 50, but panel (c) used as initialisation. **(e)** Perplexity 50, but PCA as shown in Figure 2a used as initialisation. Black arrows highlight two clusters that belong to the *Pvalb/Sst* group, as can be clearly seen in (c), but end up in a wrong place here. **(f)** Multi-scale approach: similarities between cells are computed with perplexity 50 and 500 and then averaged. PCA used as initialisation.

An alternative strategy is to use the first two PCs (after appropriate scaling, see Methods) as initialisation. We saw above that the first two PCs provide a reasonable overview of the global geometry, which is preserved during the course of t-SNE optimisation (Figure 3e). Apart from maintaining the global geometry, PCA initialisation is convenient because it makes the t-SNE outcome reproducible and not dependent on a random seed. For this data set, these two approaches yielded qualitatively similar results (Figures 3d, e), yet PCA initialisation in this particular case does a bit worse: e.g. two well-isolated clusters from the *Pvalb/Sst* group (*Sst Chodl* and *Pvalb Vipr2*, see black arrows in Figure 3e) end up far away from the main *Pvalb/Sst* continent.

Another way to capture global and local structure together is to use a multi-scale approach suggested in Lee et al. (2015). Here, several perplexity values are used together at the same time. For example, using perplexities 50 and 500, each point is strongly attracted to its ∼50 closest neighbours and weakly attracted to its ∼500 neighbours. Mathematically, this corresponds to using a similarity kernel that is an average of a narrow and a broad Gaussian (see Methods), instead of a Gaussian similarity kernel used in t-SNE. The result, when initialised with PCA, is arguably the best t-SNE visualisation of this data set that we were able to obtain (Figure 3f). The local structure is captured thanks to perplexity 50, the global structure is captured due to perplexity 500, and the overall result is deterministic due to using PCA as initialisation. Compared to the Figure 3d, the clusters are less tightly packed. A very similar result can be obtained by combining perplexities 5, 50, and 500.

An important caveat is that the success of this multi-scale approach relies on using perplexity values

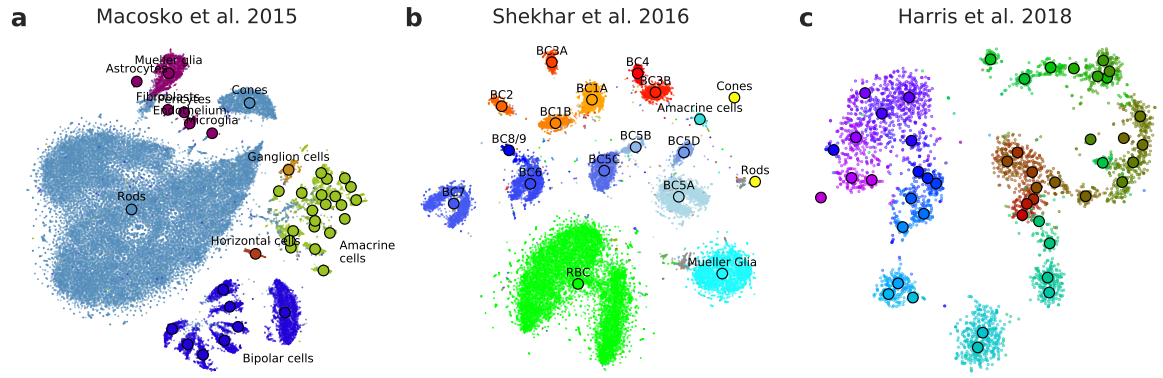

Figure 4: **(a)** t-SNE visualisation of the Macosko et al. (2015) data set, comprising $N = 44\,808$ cells from the mouse retina. Clusters and cluster colours are taken from the original publication. Perplexity combination of 50 and 500; PCA initialisation. **(b)** t-SNE visualisation of the Shekhar et al. (2016) data set, comprising $N = 27\,499$ cells from the mouse retina, mostly bipolar cells. Clusters and cluster colours are taken from the original publication. BC: bipolar cell, RBC: rod bipolar cell. Putative doublets/contamitants shown in grey. Perplexity 50; see text for the initialisation details. Some clusters appear to consist of two parts; this is due to an experimental batch effect that we did not remove. **(c)** t-SNE visualisation of the Harris et al. (2018) data set, comprising $N = 3\,663$ interneurons from the area CA1 in mouse hippocampus. Clusters and cluster colours taken from the original publication. The mean of one cluster (*Sst Cryab*) is not shown because its cells were scattered all over the figure (as they did in the original publication as well). Cluster labels not shown for visual clarity. Perplexity 50 and PCA initialisation.

that are large enough to capture the global geometry. Here we had to use $500 \approx N/50$ as the largest perplexity. If the data set had 10 times more cells, then the necessary perplexity would likely also grow 10 times. At the same time, t-SNE optimisation becomes much slower as the perplexity grows. For $N \approx 25\,000$ and perplexity 500, the algorithm takes several minutes (compared to several seconds with perplexity 50), but $N \approx 250\,000$ and perplexity 5000 would be computationally very expensive, and for even larger data sets eventually unfeasible. We discuss an alternative strategy for large N in a later section.

1.6 t-SNE analysis of UMI-based RNA-seq data

To demonstrate that our approach to t-SNE can handle UMI-based transcriptomic data as well, we consider three further data sets. First, we re-analysed the data from Macosko et al. (2015) who sequenced 44 808 cells from mouse retina. Our t-SNE result (Figure 4a) preserved much of the global geometry: e.g. multiple amacrine cell clusters ended up in one part of the figure, bipolar cell clusters in another part, and non-neural clusters in yet another part. The t-SNE analysis performed by the authors in the original publication relied on downsampling and had a much worse representation of the global geometry.

Second, we analysed the data from Shekhar et al. (2016) who sequenced $N = 27\,499$ cells from mouse retina targeting bipolar neurons. Here again, our t-SNE result (Figure 4b) is consistent with the global structure of the data: for example, OFF bipolar cells (type 1 to type 4, warm colours) and ON bipolar cells (type 5 to type 9, cold colours) were placed close to each other, and four subtypes of type 5 are next to each other as well. This was not true for the t-SNE shown in the original publication. This data set presents an additional complication because it contains several very distinct but very small clusters. In this situation, large perplexity can fail to yield a good representation of the global geometry. For example, in this data set, cone and rod photoreceptor clusters contain only ∼50 cells each, and with perplexity 500 will be strongly attracted to other, unrelated, clusters. This leads to small clusters getting "sucked" into the middle of the figure even if they are initialised on a periphery. An alternative approach is to use t-SNE without a perplexity parameter altogether, letting all cells be attracted to their neighbours in a specified vicinity, irrespective of how many such neighbours there are (see Methods). Here we use the result of such perplexity-free t-SNE as an initialisation for standard t-SNE with perplexity 50.

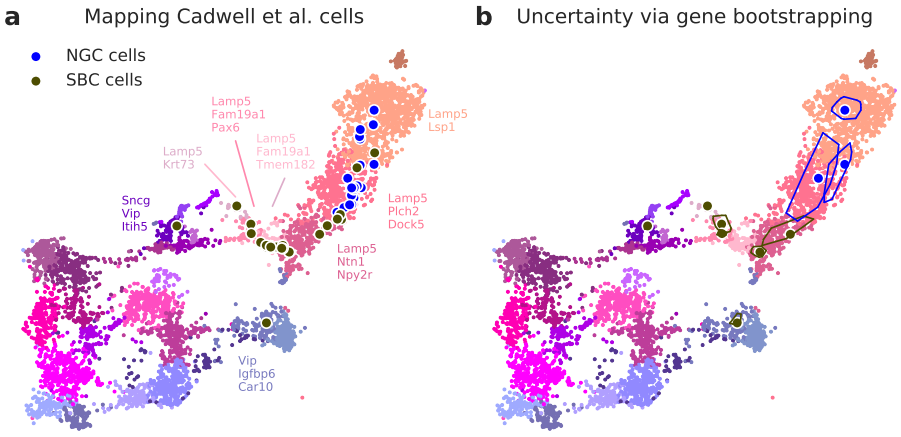

Figure 5: (a) All interneurons from Cadwell et al. (2016) positioned on the reference t-SNE atlas from Figure 3f. Only *Vip/Lamp5* continent shown here, as no cells mapped elsewhere. Cluster labels are given only for clusters where at least one cell maps to. NGC: neurogliaform cells; SBC: single bouquet cells. **(b)** Uncertainty of the mapping of 10 exemplary cells from panel (a). Polygons are convex hulls of bootstrap repetitions, see text.

Finally, we analysed a smaller data set from Harris et al. (2018) who obtained transcriptomes of $N = 3\,663$ inhibitory neurons from the area CA1 of mouse hippocampus. The original publication introduced a novel clustering and feature selection method based on the negative binomial distribution, and used a modified negative binomial t-SNE procedure. Our t-SNE visualisation (Figure 4c) did not use any of that but was nevertheless at least as good as the original one. As always, we used PCA initialisation, but in this case perplexity 50 yielded good results even with random initialisation (probably because of the relatively small size of the data set). In good agreement with our recommendations, Harris et al. (2018) also observed that performing standard t-SNE without log-transformation and/or without feature selection leads to visibly worse results.

1.7 Positioning new points on an existing t-SNE atlas

A common task in single-cell transcriptomics is to match a given cell to an existing reference data set. For example, introducing a protocol called *patch-seq*, Cadwell et al. (2016) performed patch-clamp electrophysiological recordings followed by RNA sequencing of ∼50 inhibitory cells in layer 1 of mouse VISp. Given the existence of the much larger Tasic et al. data set described above, it is natural to ask where on the Figure 3d, taken as a reference atlas, should Cadwell et al. cells be positioned.

It is often claimed that t-SNE does not allow out-of-sample mapping, i.e. no new points can be put on a t-SNE atlas after it is constructed. What this means is that t-SNE is a nonparametric method that does not construct any mapping $f(\mathbf{x})$ from a high-dimensional to the low-dimensional space (parametric t-SNE is possible, but is out of scope of this paper, see *Comparison with other methods*). However, despite the absence of such an $f(\mathbf{x})$, there is a straightforward way to position a new $\mathbf{x}$ on the existing t-SNE atlas (e.g. Berman et al., 2014).

It can be done as follows: for each Cadwell et al. cell, we find its $k = 25$ nearest neighbours among the Tasic et al. reference cells, using Pearson correlation across the most variable Tasic et al. genes as distance. Then we position the cell at the median t-SNE location of these k reference cells (Figure 5a). The result agrees very well with the assignment of Cadwell et al. cells to the Tasic et al. clusters done with a different method (Tasic et al., 2017, Figure S10).

This method assumes that for each new cell there are cells of the same type in the reference data set. Cells that do not have a good match in the reference data can end up positioned in a misleading way. However, in our case this assumption is justified because the Tasic et al. data set is much larger and includes neurons from the same layer of the same brain area. The same procedure worked very well to map the bipolar cells from the Macosko et al. (2015) data set to the t-SNE atlas of the Shekhar et al. (2016) data (not shown).

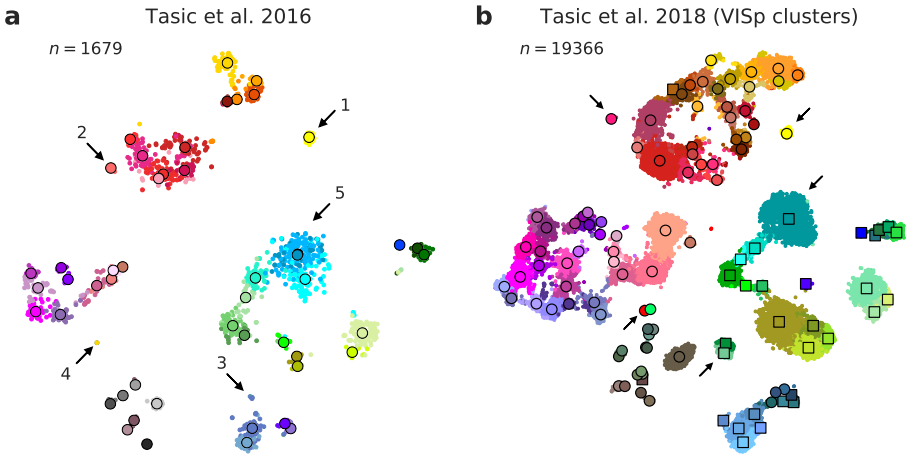

Figure 6: **(a)** t-SNE visualisation of the Tasic et al. (2016) data set. Clusters and cluster colours as in (Tasic et al., 2016). Arrows show some noteworthy cases, see text. **(b)** t-SNE visualisation of the Tasic et al. (2017) data set after excluding all clusters that mostly consist of ALM cells. This t-SNE analysis was initialised by positioning all cells on the reference atlas from panel (a), ensuring that the two panels are aligned with each other.

Berman et al. (2014) used a more sophisticated approach, where each new cell is initially positioned as outlined above (or in a similar way) but then its position is optimised using a t-SNE-like method: the cell is attracted to its nearest neighbours in the reference set and repelled from all points in the reference set, with the effective number of nearest neighbours determined by the perplexity parameter. We found that the simpler procedure without this additional optimisation step worked well for our data.

The proposed procedure does not convey any information on how trustworthy the position of each new cell is. However, the uncertainty can be estimated via bootstrapping across genes (our procedure is inspired by Tasic et al., 2017). We repeatedly selected a bootstrapped sample from the set of highly variable genes and repeated the positioning procedure (100 times). This yielded a set of bootstrapped mapping locations; the larger the variability in this set, the larger the uncertainty. To visualise the uncertainty, we draw a polygon around each cell as a convex hull of the bootstrap repetitions (Figure 5b; after excluding 5% of most outlying locations, see Methods). A large polygon means high uncertainty; small polygons mean high precision. For some cells the polygons are so small that they are not visible behind the main dot. For some other cells the polygons are large and sometimes even spread across the border of two clusters. This means that not only the exact position of this cell on the t-SNE atlas is not known, but even the cluster assignment is not certain.

1.8 Aligning two t-SNE visualisations

The data set of Tasic et al. (2017) is a follow-up to Tasic et al. (2016) that analysed $N = 1\,679$ cells from mouse VISp using a previous version of the sequencing protocol. If one excludes from the new data set all clusters that have mostly ALM cells, then the remaining data set has $N = 19\,366$ cells. How similar is the cluster structure of this newer and larger data set compared to the older and smaller one? One way to approach this question is through aligned t-SNE visualisations.

To obtain an aligned t-SNE visualisation, we perform t-SNE analysis on the Tasic et al. (2016) data set using PCA initialisation with perplexity 50, which already achieves good results (probably because of the small size of the data set; Figure 6a). Now we can position each cell of the reduced Tasic et al. (2017) data set on this reference using the procedure described above and use the resulting arrangement as initialisation for t-SNE with the perplexity combination of 50 and 500. The result is shown in Figure 6b. Of course the larger data set has higher resolution and gets clustered into more clusters. But the paired and aligned t-SNE analyses allow to make much more detailed comparisons.

The arrows in Figure 6 highlight several noteworthy findings. (1) and (2) are examples of well-isolated

clusters in the 2016 data that remain to be well-isolated in the 2017 data (*Sst Chodl* and *Pvalb Vipr2*, here and below we use the 2017 nomenclature). (3) and (4) are examples of small groups of cells that were not assigned to separate clusters back in 2016, became separate clusters on the basis of the 2017 data, but in retrospect appear well-isolated on the 2016 t-SNE plot (two *L5 LP VISp Trhr* clusters and two *Meis2/CR* clusters). Finally, (5) shows an example of several clusters in the 2016 data merging together into one cluster based on the 2017 data (*L4 IT VISp Rspo1*). All of this corresponds well to the conclusions of Tasic et al. (2017) made with other methods, but we find that t-SNE adds a valuable perspective and allows for an intuitive comparison.

1.9 Preserving global geometry with t-SNE for large data sets

Performing t-SNE on large data sets with $N \gg 100\,000$ presents three additional challenges to those already adressed above:

1. Standard t-SNE is slow for $N \gg 1000$ and computationally unfeasible for $N \gg 10\,000$. A widely used approximation called Barnes-Hut t-SNE (van der Maaten, 2014) in turn becomes very slow for $N \gg 100\,000$ (see Box 2). For larger data sets one needs a faster approximation scheme.
2. For $N \gg 100\,000$, t-SNE suffers from "overcrowding" (Linderman et al., 2017; Unen et al., 2017): clusters tend to blow up and get squeezed together, similar to Figure 3b. The exact reason for this is mathematically not well understood.
3. Our approach, described above, to preserve global geometry relies on using large ($\sim N/50$) perplexity values and becomes computationally infeasible for $N \gg 100\,000$. For such Ns one has to use perplexity values in the standard range 10–100.

The first challenge has been effectively solved by Linderman et al. (2017) who developed a novel t-SNE approximation which uses an interpolation scheme accelerated by the fast Fourier transform (see Box 2). Using this approximation, a data set with 1 million cells can be processed in ∼40 minutes on a computer with 4 double-threaded cores, 3.4 GHz each (and in ∼15 minutes using a computer with 20 double-threaded cores, 2.2 GHz each).

For the second challenge, there is so far no principled solution in the t-SNE framework (but see *Comparison with other methods*), but a very practical trick suggested in Linderman et al. (2017) is to increase the strength of all attractive forces by a small constant "exaggeration" factor between 1 and ∼10 (see Methods). This counteracts the expansion of the clusters.

Our approach to the third challenge is based on an assumption that global geometry should be detectable even after strong downsampling of the data set. This suggests the following pipeline: (i) downsample a large data set down to some manageable size; (ii) run t-SNE using large perplexity to preserve global geometry; (iii) position all the remaining points on the resulting t-SNE plot using k nearest neighbours; (iv) use it as initialisation to run t-SNE on the whole data set.

We demonstrate these ideas using the 10x Genomics data set with $N = 1\,306\,127$, which is the largest existing single-cell RNA-seq data set to date. We first create a t-SNE plot of a randomly selected subset of 25 000 cells (Figure 7a). We use PCA initialisation and perplexity 500 to capture the global geometry (combining perplexity 500 with smaller values like 50 did not seem to improve the result in this case). We then positioned all N cells on this t-SNE plot using the approach described above with $k = 1$. Despite the large size of the data set, this can be done fairly quickly (in under 10 minutes). Finally, we use the result as initialisation to run t-SNE on all N points using perplexity 30 and exaggeration coefficient 4 (Figure 7b). In addition, we had to modify two optimisation parameters from their default values: we increased the learning rate from 200 to 1000 and the length of the "early exaggeration" phase from 250 to 500 (see Methods for further details).

Validating this procedure is not straightforward, as there are no annotations available for the clusters. To identify meaningful biological structure, we looked at developmental marker genes (Englund et al., 2005; Yuzwa et al., 2017; Iacono et al., 2018). The left of the main continent is composed of radial glial cells (neural stem cells) expressing *Aldoc* and *Slc1a3* (Figure 8a). The neighbouring areas consist of neural progenitors (neuroblasts) expressing *Eomes* (previously known as *Tbr2*) (Figure 8b). The lower right parts consist of mature excitatory neurons expressing pan-neuronal markers such as *Stmn2* and *Tubb3* (Figure 8c) but not expressing inhibitory neuron markers *Gad1* or *Gad2* (Figure 8d), whereas the

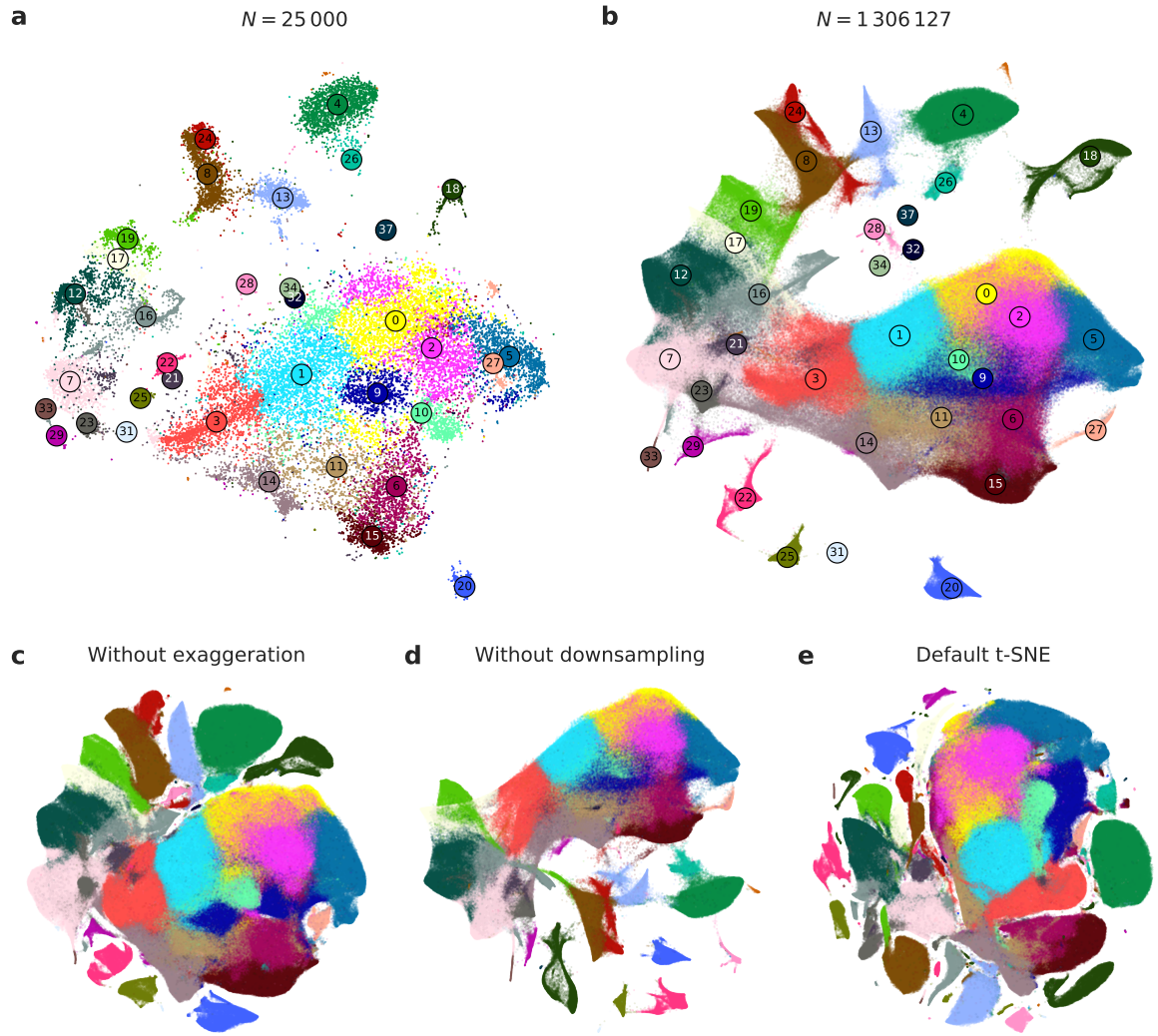

Figure 7: **(a)** t-SNE visualiation of the 10x Genomics data set, subsampled to 25 000 cells. PCA initialisation, perplexity 500. Colours and numbers correspond to clusters taken from Wolf et al. (2018). Cluster labels for several small clusters (30, 35, 36, and 38) are not shown here and in (b) because these clusters were very dispersed on our t-SNE figures. **(b)** t-SNE visualisation of the full 10x Genomics data set (1 306 127 cells). All cells were positioned on panel (a) and this was used as initialisation. Perplexity 30, exaggeration 4. **(c)** The same as in (b) but without exaggeration. **(d)** The same as in (b) but with PCA initialisation, i.e. without using the downsampling step. **(e)** Default t-SNE: random initialisation, no exaggeration.

upper parts are occupied by several inhibitory neuron clusters (Figure 8d). These marker gene gradients confirm that our t-SNE visualisation shows meaningful topology and is able to capture the developmental trajectories: from radial glia, to excitatory/inhibitory neural progenitors, to excitatory/inhibitory mature neurons.

The effectiveness of several of the components of our pipeline can be easily seen from a series of control experiments. Omitting exaggeration yields over-expanded clusters and a plot with less obvious global structure (issue #2 is not addressed, Figure 7c). Without downsampling, the global geometry is preserved worse (Figure 7d): e.g. most of the interneuron clusters are in the lower part of the figure, but clusters 17 and 19 (developing interneurons), are located in the upper part (issue #3 is not addressed). Finally, the outcome of the default t-SNE pipeline with random initialisation and no exaggeration leads to poor result that not only suffers from issues #2 and #3 but also has some clusters fragmented into several pieces (Figure 7e). Again, this is not a strawman: Figure 7e is qualitatively similar to the t-SNE

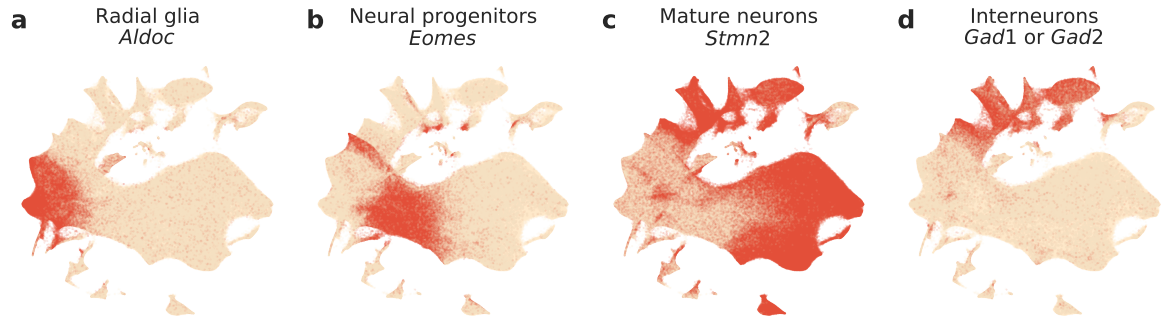

Figure 8: **(a)** Expression of *Aldoc* gene (marker of radial glia, i.e. neural stem cells) on the t-SNE map from Figure 7b. Any cell with *Aldoc* detected (UMI count above zero) was coloured in red. Another radial glia marker, *Slc1a3*, had similar but a bit broader expression. **(b)** Expression of *Eomes*, marker of neural progenitors (neuroblasts). **(c)** Expression of *Stmn2*, marker of mature neurons. A pan-neuronal marker *Tubb3* had similar but a bit broader expression. The isolated island in the bottom of the plot (cluster 20) consists of Cajal-Retzius neurons (expressing *Reln*). **(d)** Expression of *Gad1* and *Gad2* (either of them), markers of inhibitory neurons.

visualisations shown in Wolf et al. (2018) and Bhaduri et al. (2018). In additional experiments, we found that the problems in Figure 7d can be substantially improved by using more aggressive early exaggeration (Linderman and Steinerberger, 2017; Arora et al., 2018) (e.g. early exaggeration 120 for 1000 iterations). For some data sets this can be a promising alternative to the downsampling approach presented above (but see Methods for further discussion).

1.10 Advantages and disadvantages of the proposed procedure

Our protocol describes a set of tools which can be used to create accurate low-dimensional visualisations of high-dimensional scRNA-seq data sets that preserve global, hierarchical structure. Accordingly, the resulting visualisations are biologically more interpretable. The procedure works well even for large data sets and — due to the PCA initialisation — the visualisation is deterministic and not random. We provide means to create aligned presentations of two data sets or map additionally collected cells onto existing visualisations. Nevertheless, in our experience the procedure requires a certain degree of experimentation to be adapted to a different data set. In data vizualisation, as a matter of principle, there is just no "one size fits all" solution — the reason we call this "the art of using t-SNE".

1.11 Comparison with other methods

A very promising method called UMAP (McInnes and Healy, 2018) has recently attracted considerable attention from the transcriptomics community (Becht et al., 2018). Technically, UMAP is very similar to a method called largeVis (Tang et al., 2016), but McInnes and Healy (2018) provided a mathematical foundation and a convenient Python package. LargeVis/UMAP use the same attractive forces as t-SNE, but change the nature of the repulsive forces and use a different, sampling-based approach to optimisation. While UMAP is much faster than Barnes-Hut t-SNE (see Box 2), the work of Linderman et al. (2017) made t-SNE at least as fast as UMAP. One big advantage of UMAP is that, unlike t-SNE, it does not seem to suffer from "overcrowding" for large N (issue #2 discussed above). In addition, UMAP is claimed to preserve the global structure of the data better than t-SNE.

We tested this using the Tasic et al. (2017) data set. With default settings, UMAP produced a visualisation similar to Figure 3a in that the global geometry was lost, but with more fragmented and much more compressed clusters (not shown). We then modified the two key parameters (number of neighbouring points and tightness of the embedding) in their recommended ranges on a two-dimensional grid (see Methods). For some combinations the result looked good, but not quite as satisfactory as Figure 3f. We also ran UMAP with default settings on the 10x data set with 1.3 million cells; the outcome was similar to Figure 7b in terms of the main "fish" shape, but the overall geometry was worse because interneuron clusters were located in various places around the fish (not shown). Also, for this

data set, UMAP was around 3 times slower than t-SNE. On the other hand, UMAP was able to isolate some (but not all) small clusters that remained scattered on the t-SNE figure (see caption to Figure 7b). This analysis shows that UMAP does not solve the t-SNE's problems out of the box and seems to require as many careful choices as t-SNE does. Many recommendations for running t-SNE that we made in this manuscript can likely be adapted for UMAP.

Additionally, several variants of t-SNE have been proposed in the literature. One important example is a parametric version of t-SNE, where a neural network is used to create a mapping $f(\mathbf{x})$ from high-dimensional input space to two dimensions (i.e. the output layer of the network contains two neurons) and is then trained using standard deep learning techniques to yield an optimal t-SNE result (van der Maaten, 2009). One potential advantage of this approach is that the "optimal" perplexity does not need to grow with the sample size, as long as the mini-batch size remains constant (cf. issues #2 and #3 above). Parametric t-SNE has been recently combined with a variational autoencoder and applied to transcriptomic data under the name of scvis (Ding et al., 2018). The authors claim that their algorithm leads to more "interpretable" visualisations than standard t-SNE because it does a better job at preserving the global structure. Indeed, the network architecture limits the form that the mapping $f(\mathbf{x})$ can take; this implicit constraint on the complexity of the mapping prevents similar clusters from ending up in very different parts of the resulting visualisation. To demonstrate the benefits of scvis over standard t-SNE, Ding et al. (2018) re-analyzed the retinal data from Shekhar et al. (2016) and achieved a visualisation with more meaningful global structure, similar to our Figure 4b.

Another important development is the work of Pezzotti et al. (2016) on hierarchical t-SNE (HSNE). The key idea is to use random walks on the k-nearest-neighbours graph of the data to find a smaller set of "landmarks", which are points that can serve as representatives of the surrounding points. In the next round, the k-nearest-neighbours graph on the level of landmarks is constructed. This operation can reduce the size of the data set by an order of magnitude, and can be repeated until the data set becomes small enough to be analysed with t-SNE. Each level of the landmarks hierarchy can be explored separately. Unen et al. (2017) successfully applied this method to mass cytometry data sets with up to 15 million cells.

All of these methods can potentially become important tools for the transcriptomic data analysis. Our goal in this manuscript was not to argue that t-SNE is better or worse than any of them, but to demonstrate what t-SNE itself is capable of when applied with care and to provide a hands-on protocol for doing so.

2 Methods

2.1 Pre-processing

Let $\mathbf{X}$ be a $N \times p$ matrix of gene counts, with N and p being the number of cells and the number of genes respectively. We assume that zero columns (if any) have been removed.

Library size normalisation For each cell, we normalise the counts by the cell's library size $\sum_k X_{ik}$, and multiply by 1 million, to obtain counts per million (CPM):

$$\frac{X_{ij}}{\sum_k X_{ik}} \cdot 10^6.$$

Some studies prefer to multiply by the median library size across all N cells instead of using 1 million. In our experience, this does not make a difference. In fact, for the Tasic et al. (2017) data, library size normalisation did not make much of a difference at all. Using RPKM or TPM data instead of read counts did not change the results much either.

Feature selection Most studies use the mean-variance relationship to perform feature selection (e.g. Zheng et al., 2017): they select genes that have large variance given their mean. We adapt the approach of Andrews and Hemberg (2018) who exploit the mean-dropout relationship: the idea is to select genes that have high dropout (i.e. zero count) frequency given their mean expression level (Figure 9a). Any gene that has high dropout rate and high mean expression could potentially be a marker of some particular

Box 2: The t-SNE optimisation The original publication (van der Maaten and Hinton, 2008) suggested optimising $\mathcal{L}$ using adaptive gradient descent with momentum. They initialised $\mathbf{y}_i$ with a standard Gaussian distribution with standard deviation 0.0001. It is important that initial values have small magnitude: otherwise optimisation fails to converge to a good solution.

To escape bad local minima, van der Maaten and Hinton (2008) suggested an "early exaggeration" trick: during initial iterations they multiply all attractive forces by $\alpha > 1$. Later work (van der Maaten, 2014) used $\alpha = 12$ for 250 iterations, which became the default since then.

The exact t-SNE computes N^2 similarities p_{ij} and N^2 pairwise attractive and repulsive forces on each gradient descent step. This becomes infeasible for $N \gg 10\,000$. In the follow-up paper, van der Maaten (2014) suggested two approximations in order to speed up the computations. First, he noticed that for any perplexity value P all but $\mathcal{O}(P)$ nearest neighbours of any given point i will have nearly zero values $p_{j|i}$. He suggested to only find $k = 3P$ nearest neighbours of each point and set $p_{j|i} = 0$ for the remaining $N - 3P$ points. This relied on finding the exact $3P$ nearest neighbours, but in later work various authors (Pezzotti et al., 2017; Tang et al., 2016; Linderman et al., 2017; McInnes and Healy, 2018) started using approximate nearest neighbour algorithms which is much faster and does not seem to make t-SNE results any worse.

Using $3P$ nearest neighbours accelerates computation of the attractive forces. To accelerate the repulsive force computations, van der Maaten (2014) used the Barnes-Hut approximation, originally developed for N-body simulations in physics. This reduces computational complexity from $\mathcal{O}(N^2)$ to $\mathcal{O}(N \log N)$, works reasonably fast for N up to $\sim$100 000, but becomes too slow for much larger sample sizes. Inspired by the fast multipole method (FMM), another technique originally developed for N-body simulations, Linderman et al. (2017) suggested using a fast Fourier transform (FFT) to accelerate the interpolation of the repulsive forces on an equispaced grid. This lowers computational complexity to $\mathcal{O}(N)$ and works very fast for N into millions. Throughout the paper, we use their C++ implementation available at `https://github.com/klugerlab/FIt-SNE`.

There is some recent work on how to accelerate t-SNE computations by using GPUs (Chan et al., 2018; Pezzotti et al., 2018), but we did not explore these implementations here.

subpopulation. We found it more intuitive to use the mean across non-zero counts instead of the overall mean, because it is computed independently of the fraction of zero counts.

To be precise, for each gene g, we compute the fraction of "near-zero" counts

$$d_g = \frac{1}{N} \sum_i I(X_{ig} \le t)$$

and the mean log "non-zero" expression

$$m_g = \big\langle \log_2 X_{ig} \mid X_{ij} > t \big\rangle.$$

For all UMI-based data sets we used $t = 0$ but for Smart-seq2 data sets we found $t = 32$ to perform better (some known marker genes were not getting selected with $t = 0$). We discard all genes that have "non-zero" expression in less than $n_{\min} = 10$ cells. We then plot d_g versus m_g for all remaining genes (Figure 9a) and inspect the relationship. As we are not interested in conducting formal significance tests for whether a gene has higher than chance dropout rate, but rather want to select a pre-specified number M of genes (usually $M = 1000$ or $M = 3000$), we use a heuristic approach of finding a value b such that

$$d_g > \exp\big[-a(m_g - b)\big] + 0.02$$

was true for exactly M genes. This can be done with a simple binary search. In Figure 9a this corresponds to moving the red exponent line horizontally until there are exactly M genes to the upper-right of it. These M genes were then selected. We used $a = 1$ for most data sets, but changed it to $a = 1.5$ for some data sets to provide a better fit for the distribution.

Figure 9b shows the effect on t-SNE when feature selection step is omitted (i.e. PCA is performed on the full-sized $N \times p$ matrix). For the Tasic et al. (2017) data set, this clumps multiple clusters together. The same effect was observed by Harris et al. (2018, Figure S4) in their UMI-based data set.

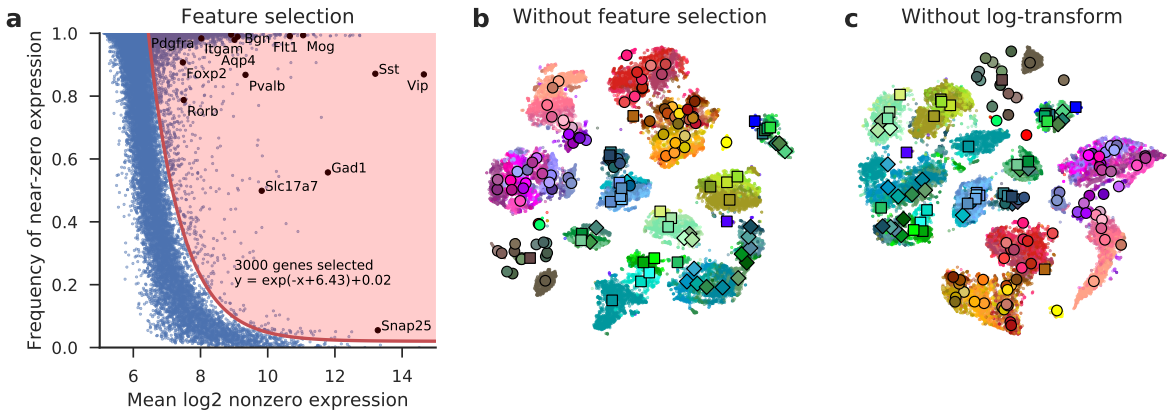

Figure 9: **(a)** Our feature selection procedure for the Tasic et al. (2017) data set. Black dots show well-known marker genes, taken from Tasic et al. (2016, Figure 1c). Any good feature selection procedure should confidently select all those. **(b)** The effect of omitting the feature selection step on t-SNE of the Tasic et al. (2017) data set. All other settings as in Figure 3f. **(c)** The effect of omitting the log-transformation on t-SNE of the Tasic et al. (2017) data set. All other settings as in Figure 3f.

We performed feature selection using the raw counts (before library size normalisation). Then we used library-size normalised values (counts per million) for the selected M genes.

Nonlinear transformation We transform all values in the $N \times M$ count matrix after feature selection with a $\log_2(x+1)$ transformation. Figure 9c shows the effect on t-SNE when the log-transformation step is omitted. For the Tasic et al. (2017) data set, this clumps multiple clusters together. The same effect was observed by Harris et al. (2018, Figure S4) in their UMI-based data set.

The $\log_2(x+1)$ transformation is standard in transcriptomics literature. It is convenient because all zeros remain zeros, and at the same time the expression counts of all genes become roughly comparable. Without this transformation, the Euclidean distances between cells are dominated by a handful of genes with very high counts. However, there are other transformations that can perform similarly well. In the cytometric literature, the inverse hyperbolic sine $\operatorname{arsinh}(x) = \ln\left(x + \sqrt{x^2+1}\right)$ is often used, sometimes as $\operatorname{arsinh}(x/r)$ with $r = 5$ or a similar value (Amir et al., 2013). Note that $\operatorname{arsinh}(x/r)$ is the variance-stabilizing transformation for the negative binomial distribution with parameter r, which is often taken to model UMI counts well.

Another possibility is to use the $x^{1/8}$ transformation (or another small power of x). It looks very similar to $\log(x)$ for a large range of x values, which are typical for expression counts. In our experiments, all these transformations performed similarly well.

Standardisation Many studies standardise the $N \times M$ matrix after the log-transformation, i.e. center and scale all columns to have mean zero and unit variance. We prefer not to do this by default. In general, standardisation is recommended when different features are on different scale, but the log-transformed counts of different genes are arguably *already* on the same scale.

From a more theoretical point of view, if one could assume that the expression counts of each gene for cells of the same type are distributed log-normally, then Euclidean distance after log-transformation would exactly correspond to the log-likelihood. For the UMI-based data, the common assumption is that the expression counts are distributed according to the negative binomial distribution. For large counts, the negative binomial distribution behaves qualitatively similarly to the log-normal (for example, its variance function is $\mu + \mu^2/r$ whereas the log-normal has variance function proportional to μ^2), so the Euclidean distance after the log-transformation can be thought of as approximating the negative binomial log-likelihood (Harris et al., 2018). Standardising all the genes after log-transformation would destroy this relationship.

At the same time, in some data sets we observed a stronger separation between some of the classes if one does perform the standardisation step. We recommend trying to do the analysis both ways. Here

we apply standardisation for those data sets in which it was used by the original authors (see below).

Principal component analysis We used PCA to reduce the size of the data matrix from $N \times M$ to $N \times 50$ prior to running t-SNE. In our experiments, this does not have much influence on the t-SNE results but is computationally convenient. Some studies estimate the number of "significant" principal components via shuffling (e.g. Shekhar et al., 2016). In our experiments, for the data sets with tens of thousands of cells, the number of significant PCs is close to 50 (for example, for the Tasic et al. (2017) data set it is 37, according to the shuffling test). Given that PCA does not have much influence on the t-SNE results, we prefer to use a fixed value of 50. Note that using many fewer PCs would be detrimental because it could substantially distort between-cell distances.

When we use PCA for initialisation of t-SNE, we always divide the first two principal components by the standard deviation of PC1 and multiply them by 0.0001 (which is the default standard deviation of the random initialisation). This is very important: values used for initialisation should be close to zero, otherwise the algorithm will have problems with convergence.

The sign of the principal components is arbitrary. To increase reproducibility of the figures, we always fix the sign of the first two PCs such that for each PCA eigenvector the sum of its values was positive (i.e. if it is negative, we flip the sign).

2.2 Details of the t-SNE analysis

This section gives additional details of our t-SNE analysis. We used a C++ implementation of t-SNE by Linderman et al. (2017), available at `https://github.com/klugerlab/FIt-SNE`. While working on this paper, we contributed to this package several additional features that were crucial for our pipeline. Our analysis code in Python is available at `https://github.com/berenslab/rna-seq-tsne`.

Default parameters for t-SNE optimisation Unless explicitly stated, we used the default optimisation parameters of the `FIt-SNE` package. Following (van der Maaten, 2014), the defaults are 1000 iterations with learning rate $\eta = 200$; momentum .5 for the first 250 iterations and .8 afterwards; early exaggeration $\alpha = 12$ for the first 250 iterations; initialisation of the points in the 2D embedding space with coordinates drawn from a standard Gaussian distribution with standard deviation 0.0001. Further input parameters for the nearest neighbour search using the `Annoy` library (number of trees: 50, number of query nodes: $3P \cdot 50$) and for the grid interpolation (number of approximating polynomials: 3, maximum grid spacing: 1, minimum grid size: 50) were always left at the default values.

Multi-scale similarities We follow Lee et al. (2015) in their definition of multi-scale similarities. For example, to combine perplexities 50 and 500, the values $p_{j|i}$ (see Box 1) are computed with perplexity 50 and with perplexity 500 for each cell i and then averaged. This is approximately equivalent to saying that we are using a different similarity kernel: instead of the Gaussian kernel $\exp(d^2/2\sigma_i^2)$ where d is Euclidean distance, a multi-scale kernel

$$\frac{1}{\sigma_i}\exp\left(\frac{d^2}{2\sigma_i^2}\right) + \frac{1}{\tau_i}\exp\left(\frac{d^2}{2\tau_i^2}\right)$$

is used. The variances σ_i^2 and τ_i^2 are selected such that the perplexity of the first Gaussian component was 50 and the perplexity of the second Gaussian component was 500. In other words, this kernel has two adaptive parameters.

Exaggeration Early exaggeration (see Box 2) means multiplying the attractive term in the loss function (see last equation in Box 1) by a constant $\alpha > 1$ during the initial iterations (the default is $\alpha = 12$ for 250 iterations). Linderman et al. (2017) suggested to use "late exaggeration": for an example data set with 1 million points they used $\alpha = 12$ to increase the attractive forces during the last 250 iterations. Thus, their approach used three stages: early exaggeration, followed by no exaggeration, followed by late exaggeration. For simplicity, we prefer to use two stages only: we keep α constant after the early exaggeration is turned off. This is why we call it "exaggeration" and not "late exaggeration".

Tasic et al. 2017 We selected the $M = 3000$ most variables genes using our procedure with $t = 32$ and $a = 1$ (resulting in $b = 6.43$). We used t-SNE with random initialisation (random seed set to 42) for Figures 3a–c with perplexities 50, 5, and 500 respectively. For Figure 3d, we used Figure 3c as initialisation for running t-SNE with perplexity 50. For Figure 3e, we used scaled PCA initialisation. For Figure 3f, we used scaled PCA initialisation and a perplexity combination of 50 and 500. Increasing the learning rate and/or the number of iterations decreased the loss but did not correspond to any noticeable visual improvement.

Visualisation In all figures, we use the median across all points in a cluster to draw a cluster centroid. For the Tasic et al. (2017) data set, we use three different symbols for cluster centroids: squares if more than 95% of cells in a cluster come from VISp, rhombi if more than 95% of cells in a cluster come from ALM, and circles otherwise.

Cadwell et al. 2016 Out of the 3000 most variables genes in the Tasic et al. (2017) data set, 2547 were present in the Cadwell et al. (2016) data. This was the gene set we used for mapping (Figure 5a). Each cell was positioned at the median t-SNE location of its $k = 25$ nearest neighbours (in terms of correlation distance) among the Tasic et al. (2017) cells.

Bootstrapping over genes We use bootstrapping to estimate the uncertainty of the mapping of new cells to the existing t-SNE visualisation. Given a set of L genes that are used for mapping (e.g. for Cadwell et al. data we used $L = 2547$ genes as stated above), we select a bootstrap sample of L genes out of L with repetition and performing the positioning procedure using this sample of genes. This constitutes one bootstrap iteration. We do 100 iterations and, for each cell, obtain 100 positions on the t-SNE atlas. The larger the spread of these positions, the larger the mapping uncertainty.

For Figure 5b, we computed the distances from the original mapping position to the 100 bootstrapped positions and discarded 5 bootstrap positions with the largest distance (as possible outliers). Then we drew a convex hull of the remaining 95 bootstrap positions (using `scipy.spatial.ConvexHull`).

Tasic et al. 2016 We selected the $M = 1000$ most variables genes using our procedure with $t = 32$ and $a = 1$ (resulting in $b = 8.58$). We used t-SNE with perplexity 50 and scaled PCA initialisation (Figure 6a).

Aligning Tasic et al. 2016 and 2017 Out of the 1000 most variables genes in the Tasic et al. (2016) data set, 967 were present in the Tasic et al. (2017) data. This was the gene set we used for mapping (Figure 6b). We excluded all cells assigned to ALM clusters (23 clusters that have "ALM" in the cluster name) in Tasic et al. (2017). Each remaining cell was positioned at the median t-SNE location of its $k = 10$ nearest neighbours (in terms of correlation distance) among the Tasic et al. (2016) cells. This was used as initialisation for t-SNE of the non-ALM subset of Tasic et al. (2017) (Figure 6b) with combined perplexities 50 and 500. Note that this initialisation was not scaled (as we do for PCA initialisation, see above) because otherwise it gets distorted during optimisation (however, when applying the alignment procedure to larger data sets, we did have to scale the initialisation as otherwise t-SNE did not converge well). For the non-ALM subset we selected $M = 3000$ most variables genes using our procedure with $t = 32$ and $a = 1$ (resulting in $b = 6.51$).

Macosko et al. 2015 We selected the $M = 3000$ most variables genes using our procedure with $t = 32$ and $a = 1$ (resulting in $b = 0.19$). We standardised all features before running PCA, following the original publication. We ran t-SNE with combined perplexities 50 and 500, scaled PCA initialisation, and learning rate $\eta = 1000$ (Figure 4a). Using the default learning rate produced almost the same result, but with some visible hallmarks of incomplete convergence.

Shekhar et al. 2016 We selected the $M = 1000$ most variables genes using our procedure with $t = 32$ and $a = 1.5$ (resulting in $b = 0.21$). We only used 1000 genes because the data set contained mostly bipolar cells which are all similar to each other. We standardised all features before running PCA, following the original publication. We ran t-SNE with $\sigma = .05$ and $k = 1000$, and scaled PCA

initialisation to get the representation of the global geometry. In this case, the $N \times 50$ matrix after PCA was divided by the maximum absolute value of its elements. This makes all the values be between -1 and 1, so that the $\sigma = .05$ parameter is more interpretable. For comparison, when trying out this procedure on the Tasic et al. (2017) data set, we found that we needed to use $\sigma = 0.25$ (with the same $k = 1000$) to get a reasonable representation of the global structure.

We then used the result (without any scaling) as initialisation to run t-SNE with perplexity 50 and exaggeration $\alpha = 1.2$ (no additional early exaggeration) (Figure 4b). Without exaggeration the result was very similar but clusters were slightly less compact.

Shekhar et al. used combined data from two animals and there is some batch effect visible in Figure 4b. In the original publication batch correction was performed prior to PCA. We did not use batch correction here because this is outside the scope of the manuscript. Using larger set of genes (we tried 10 000) strongly exacerbated batch effect. The original publication used ∼13 000 genes.

Harris et al. 2018 We selected the $M = 150$ most variables genes using our procedure with $t = 32$ and $a = 1.5$ (resulting in $b = 0.52$). We used the same number of genes as in the original publication. We did *not* standardise the features prior to PCA, because this better corresponds to the procedure in the original publication (see above). We used t-SNE with perplexity 50 and scaled PCA initialisation (Figure 4c).

10x Genomics (1.3 mln cells) We used the `recipe_zheng17()` preprocessing pipeline from `scanpy` (Wolf et al., 2018) to ease the comparison with clustering and dimensionality reduction performed by Wolf et al. This pipeline follows Zheng et al. (2017) and is similar to ours: it performs library size normalisation, selects the 1000 most variable genes based on the mean-variance relationship, applies the $\log(x + 1)$ transform, standardises each feature, and uses PCA to reduce dimensionality to 50. We performed t-SNE on the data set randomly downsampled to 25 000 cells with perplexity 500 and scaled PCA initialisation. We then positioned each cell at the t-SNE position of its $k = 1$ nearest neighbour (in terms of Euclidean distance) among the 25 000-cell subset and scaled the resulting arrangement down to standard deviation 0.0001 exactly as we do with scaled PCA initialisation (see above). We used this as initialisation to run t-SNE with perplexity 30, early exaggeration $\alpha = 12$ for 500 iterations and exaggeration $\alpha = 4$ afterwards, and learning rate $\eta = 1000$ (Figure 7b).

For data sets of such size, the default learning rate $\eta = 200$ is too small and one needs to increase either the learning rate up to 1000 (as we do here) or the number of iterations from 1000 up to ∼5000. Otherwise the result remains visibly not converged ("noisy"). We increased the length of the early exaggeration phase from the default 250 to 500 iterations because otherwise cluster 24 remained split into two disconnected parts. Figure 7c corresponds to the early exaggeration $\alpha = 12$ for 500 iterations but no exaggeration after that. Figure 7d shows the result with scaled PCA initialisation instead of the downsampling-based initialisation. Figure 7e was obtained with all the default settings (random initialisation, default early exaggeration regime) apart from the learning rate $\eta = 1000$.

When initialising with scaled PCA as in Figure 7c, we obtained much better results when using aggressive early exaggeration (Linderman and Steinerberger, 2017; Arora et al., 2018): we used $\alpha = 120$ for 1000 iterations followed by another 1000 iterations with $\alpha = 4$. The result was similar to Figure 7b. Nevertheless, we do not believe that aggressive early exaggeration can always help retaining the global structure. Indeed, using perplexity 30 means that each cell is only attracted to $k = 90$ nearest neighbours. If the global structure is not visible on this scale (imagine several far separated clusters with $\gg 90$ points each), then no early exaggeration or other optimisation settings would yield an adequate representation of the global structure.

When using scaled PCA initialisation with this data set, we always rotated the result 90 degrees clockwise and flipped horizontally, to make it visually more pleasing. Note that t-SNE result can be arbitrarily rotated and flipped as this does not change the distances between points.

UMAP For our experiments with UMAP on the Tasic et al. (2017) data set, we used `n_neighbors` from $\{5, 10, 50, 100\}$ and `min_dist` from $\{.001, .01, .1, .5\}$ (defaults are 15 and .1 respectively), and tried all 4×4 combinations. When running UMAP on the 10x Genomics data set, we used the default settings.

3 Procedure

We created a Jupyter notebook in Python that guides through all the steps of this protocol. It is available at https://github.com/berenslab/rna-seq-tsne. We suggest to install Python and Jupyter from https://www.anaconda.com, install FIt-SNE following instructions at https://github.com/klugerlab/FIt-SNE, download the notebook, and follow its steps. The protocol outline below gives the key steps with additional information. The FIt-SNE library is written in C++ and provides interfaces for R, Matlab, and Python. Our code is in Python, but it is equally easy to use the other two interfaces.

Our tutorial notebook is self-contained. For simplicity, it uses the Tasic et al. (2017) data set for all demonstrations. To show how to map new cells to the reference t-SNE atlas, we split the data set into training set and test set. To show how to align two t-SNE visualisations, we split the data set into two parts. To show how to process a much larger data set, we replicate each cell 10 times and add noise. See the same GitHub repository for the notebooks that generate all figures used in this manuscript.

The notebook starts with processing the data files as they are distributed by the Allen institute (http://celltypes.brain-map.org/rnaseq). This takes 5 minutes. The raw count data in the sparse matrix format is put into `tasic2018.counts`. Cluster ids of all cells are put into `tasic2018.clusters` and colours of each cluster are in `tasic2018.clusterColors`.

The reported timings were measured on a standard personal workstation with 4 double-threaded cores, 3.4 GHz each, and 32Mb RAM. Half the amount of this RAM should be enough to run the notebook.

Step 1: Pre-processing [20 sec]

To select the 3000 most variable genes, run

```
# Get mean log non-zero expression of each gene
x = meanLogExpression(tasic2018.counts, threshold=32)
# Get near-zero frequency of each gene
y = nearZeroRate(tasic2018.counts, threshold=32)
# Adjust the threshold to select 3000 genes
selectedGenes = featureSelection(x, y, n=3000)
```

After that, we execute the pre-processing pipeline as follows:

```
counts3k = tasic2018.counts[:, selectedGenes]  # Feature selection

librarySizes = tasic2018.counts.sum(axis=1)    # Compute library sizes
CPM = counts3k / librarySizes * 1e+6           # Library size normalisation

logCPM = np.log2(CPM + 1)                      # Log-transformation

X = PCA(n_components=50).fit_transform(logCPM) # Principle component analysis (PCA)
```

Step 2: Initial data exploration using PCA [1 sec]

As we already did PCA as one of the pre-processing steps, we directly visualise the first two principal components:

```
plt.figure(figsize=(5,5))
plt.scatter(X[:,0], X[:,1], s=3, color=tasic2018.clusterColors[tasic2018.clusters])
plt.tight_layout()
```

Similar plotting code can be used to visualise all later steps as well and will therefore not be repeated as part of the protocol.

If initial PCA indicates several distinct clusters, run PCA on each of them separately. For our example data set, we perform PCA of the excitatory neurons by running:

```
subset = X[:,0] > -8   # Select the desired subset here
Xsubset = PCA(n_components=2).fit_transform(X[subset,:])
```

If clustering has already be done and cluster information is available, then it can be helpful to look at a PCA and multi-dimensional scaling (MDS) of cluster means:

```
C = np.unique(tasic2018.clusters).size
clusterMeans = np.zeros((C, X.shape[1]))
for c in range(C):
    clusterMeans[c,:] = np.mean(X[tasic2018.clusters==c,:], axis=0)

clusterMeansPCA = PCA(n_components=2).fit_transform(clusterMeans)
clusterMeansMDS = MDS(n_components=2).fit_transform(clusterMeans)
```

The given timing (1 second) is what it takes to run the code above. Actual data exploration is an interactive process that of course takes much longer.

Step 3: Creating a standard t-SNE visualisation [30 sec]

Run t-SNE with standard settings:

```
tsne30 = fast_tsne(X)
```

This uses perplexity 30.

Step 4: Exploring the data set at different scales [10 min]

Run t-SNE with different perplexities in the range from 5 to $\sim N/10$. This assumes that the data set size is small enough to allow that. For much larger data sets than the one used here, consider downsampling to some manageable size and exploring the downsampled version first.

```
tsne5   = fast_tsne(X, perplexity=5,   seed=42)
tsne50  = fast_tsne(X, perplexity=50,  seed=42)
tsne500 = fast_tsne(X, perplexity=500, seed=42)
```

Running t-SNE with the largest perplexity takes 3 minutes, which is the slowest of the three function calls. We use `seed=42` to set the same random initialisation in all runs.

For some data sets, especially if there are very small very distinct clusters, it can be helpful to run t-SNE without perplexity calibration (see the discussion of the Shekhar et al. (2016) data set above). This can be done as follows (each t-SNE run takes 2 minutes):

```
tsne01  = fast_tsne(X/np.max(np.abs(X)), sigma=0.1,  K=1000, seed=42)
tsne025 = fast_tsne(X/np.max(np.abs(X)), sigma=0.25, K=1000, seed=42)
tsne05  = fast_tsne(X/np.max(np.abs(X)), sigma=0.5,  K=1000, seed=42)
```

Step 5: Refining the t-SNE visualisation [3 min]

To initialise t-SNE with scaled PCA coordinates, run

```
PCAinit = X[:,:2] / np.std(X[:,0]) * 0.0001
tsne30pca = fast_tsne(X,  initialization=PCAinit)
```

To use a perplexity combination, along with scaled PCA initialisation, run

```
PCAinit = X[:,:2] / np.std(X[:,0]) * 0.0001
tsne50_500 = fast_tsne(X, perplexity_list=[50,500], initialization=PCAinit)
```

Choose the plots which are most useful for highlighting relevant aspects of the data structure.

Step 6: Mapping new cells to an existing visualisation (optional) [3 min]

For demonstration purposes, we use 100 randomly selected cells as "new cells" (test set) that we want to place on a t-SNE map created using the remaining cells (training set).

Perform t-SNE on the training set:

```
Xtrain = PCA(n_components=50).fit_transform(logCPM[trainingSet,:])
PCAinit_train = Xtrain[:,:2] / np.std(Xtrain[:,0]) * 0.0001
tsneTrain = fast_tsne(Xtrain, perplexity_list=[50,500], initialization=PCAinit_train)
```

And then position new points as follows (this runs in a few seconds):

```
pos = map_to_tsne(tasic2018.counts[:,selectedGenes][trainingSet,:],
                  tasic2018.counts[:,selectedGenes][testSet,:],
                  tsneTrain)
```

See the notebook for the implementation of `map_to_tsne()`.

Step 7: Creating aligned t-SNE visualisations (optional) [4 min]

We use 5000 randomly selected cells as the first data set and the rest as the second data set. To create an aligned data set of the two parts, perform t-SNE on the first data set:

```
X1 = PCA(n_components=50).fit_transform(logCPM[set1,:])
PCAinit1 = X1[:,:2] / np.std(X1[:,0]) * 0.01
tsne1 = fast_tsne(X1, perplexity=50, initialization=PCAinit1)
```

Then map the second data set on to this t-SNE result:

```
pos = map_to_tsne(tasic2018.counts[:,selectedGenes][set1,:],
                  tasic2018.counts[:,selectedGenes][set2,:],
                  tsne1)
```

Finally, use this mapping as initialisation to run t-SNE on the second data set:

```
X2 = PCA(n_components=50).fit_transform(logCPM[set2,:])
tsne2 = fast_tsne(X2, perplexity_list=[50,500], initialization=pos)
```

Step 8: Dealing with large data sets (optional) [10 min]

For demonstration purposes, we replicate each cell 10 times and add Gaussian noise to obtain a simulated data set with $N = 238\,000$. We assume that this data set is so large that we cannot use any perplexity values much larger than the default one (30). This is not quite true for this N but would be true if we did another tenfold increase.

Running t-SNE with PCA initialisation produces a very poor result with over-expanded clusters:

```
PCAinit10x = X10x[:,:2] / np.std(X10x[:,0]) * 0.0001
tsne10x = fast_tsne(X10x, initialization=PCAinit10x)
```

Running t-SNE with PCA initialisation and exaggeration coefficient $\alpha = 4$ prevents over-expansion of the clusters but fails to preserve global geometry:

```
tsne10x_ex = fast_tsne(X10x, initialization=PCAinit10x,
                       late_exag_coeff=4, start_late_exag_iter=250)
```

Instead, use downsampling-based initialisation: Perfrom t-SNE in a global-geometry-preserving way on a downsampled data set (here we use the original data set, 3 min); map all cells onto the resulting t-SNE (1.5 min); run t-SNE on the full data set with this initialisation and exaggeration turned on (3 min):

```
pos = map_to_tsne_fast(X, X10x, tsne50_500)
downsampled_init = pos / np.std(pos[:,0]) * 0.0001
tsne10x_pos = fast_tsne(X10x, initialization=downsampled_init,
                        late_exag_coeff=4, start_late_exag_iter=250)
```

See the notebook for the implementation of `map_to_tsne_fast()`.

4 Anticipated Results

A successful completion of the protocol will result in a t-SNE visualisation for a given scRNA-seq data set that preserves the global structure. This t-SNE map can be annotated with the output of a clustering procedure or the expression level of desired marker genes. The optional steps allow to achieve such a result even for very large data sets.

Acknowledgements We thank George Linderman, Leland McInnes, James Melville, Pavlin Poličar, and Alexander Wolf for discussions of t-SNE and UMAP. We thank Andreas Tolias for discussing patch-seq data. This work was supported by the Deutsche Forschungsgemeinschaft (BE5601/4-1, EXC 2064, EXC 307), the Federal Ministry of Education and Research (FKZ 01GQ1601) and the National Institute of Mental Health of and National Institute of Neurological Disorders And Stroke under Award Number U19MH114830. The content is solely the responsibility of the authors and does not necessarily represent the official views of the National Institutes of Health.

Author contributions DK and PB conceptualized the project, DK performed analysis and developed the protocol, DK and PB wrote the paper.

Competing interests The authors declare no competing interests.

References

El-ad David Amir, Kara L Davis, Michelle D Tadmor, Erin F Simonds, Jacob H Levine, Sean C Bendall, Daniel K Shenfeld, Smita Krishnaswamy, Garry P Nolan, and Dana Pe'er. viSNE enables visualization of high dimensional single-cell data and reveals phenotypic heterogeneity of leukemia. *Nature Biotechnology*, 31(6):545, 2013.

Tallulah S. Andrews and Martin Hemberg. Dropout-based feature selection for scrnaseq. *bioRxiv*, 2018.

Sanjeev Arora, Wei Hu, and Pravesh K Kothari. An analysis of the t-SNE algorithm for data visualization. In *Conference on Learning Theory*, pages 1–8, 2018.

Etienne Becht, Charles-Antoine Dutertre, Immanuel WH Kwok, Lai Guan Ng, Florent Ginhoux, and Evan W Newell. Evaluation of UMAP as an alternative to t-SNE for single-cell data. *bioRxiv*, 2018.

Gordon J Berman, Daniel M Choi, William Bialek, and Joshua W Shaevitz. Mapping the stereotyped behaviour of freely moving fruit flies. *Journal of the Royal Society Interface*, 11(99):20140672, 2014.

Aparna Bhaduri, Tomasz J. Nowakowski, Alex A. Pollen, and Arnold R. Kriegstein. Identification of cell types in a mouse brain single-cell atlas using low sampling coverage. *BMC Biology*, 16(1):113, Oct 2018.

Cathryn R Cadwell, Athanasia Palasantza, Xiaolong Jiang, Philipp Berens, Qiaolin Deng, Marlene Yilmaz, Jacob Reimer, Shan Shen, Matthias Bethge, Kimberley F Tolias, et al. Electrophysiological, transcriptomic and morphologic profiling of single neurons using patch-seq. *Nature Biotechnology*, 34 (2):199, 2016.

David M Chan, Roshan Rao, Forrest Huang, and John F Canny. t-SNE-CUDA: GPU-accelerated t-SNE and its applications to modern data. *arXiv*, 2018.

Alex Diaz-Papkovich, Luke Anderson-Trocme, and Simon Gravel. Revealing multi-scale population structure in large cohorts. *bioRxiv*, 2018.

Jiarui Ding, Anne Condon, and Sohrab P Shah. Interpretable dimensionality reduction of single cell transcriptome data with deep generative models. *Nature Communications*, 9(1):2002, 2018.

Chris Englund, Andy Fink, Charmaine Lau, Diane Pham, Ray AM Daza, Alessandro Bulfone, Tom Kowalczyk, and Robert F Hevner. Pax6, Tbr2, and Tbr1 are expressed sequentially by radial glia, intermediate progenitor cells, and postmitotic neurons in developing neocortex. *Journal of Neuroscience*, 25(1):247–251, 2005.

Xiaoping Han, Renying Wang, Yincong Zhou, Lijiang Fei, Huiyu Sun, Shujing Lai, Assieh Saadatpour, Zimin Zhou, Haide Chen, Fang Ye, et al. Mapping the mouse cell atlas by microwell-seq. *Cell*, 172(5): 1091–1107, 2018.

Kenneth D Harris, Hannah Hochgerner, Nathan G Skene, Lorenza Magno, Linda Katona, Carolina Bengtsson Gonzales, Peter Somogyi, Nicoletta Kessaris, Sten Linnarsson, and Jens Hjerling-Leffler. Classes and continua of hippocampal CA1 inhibitory neurons revealed by single-cell transcriptomics. *PLoS Biology*, 16(6):e2006387, 2018.

Geoffrey E Hinton and Sam T Roweis. Stochastic neighbor embedding. In *Advances in Neural Information Processing Systems*, pages 857–864, 2003.

Giovanni Iacono, Elisabetta Mereu, Amy Guillaumet-Adkins, Roser Corominas, Ivon Cuscó, Gustavo Rodríguez-Esteban, Marta Gut, Luis Alberto Pérez-Jurado, Ivo Gut, and Holger Heyn. bigSCale: an analytical framework for big-scale single-cell data. *Genome Research*, 2018.

John A Lee, Diego H Peluffo-Ordóñez, and Michel Verleysen. Multi-scale similarities in stochastic neighbour embedding: Reducing dimensionality while preserving both local and global structure. *Neurocomputing*, 169:246–261, 2015.

Wentian Li, Jane E Cerise, Yaning Yang, and Henry Han. Application of t-SNE to human genetic data. *Journal of Bioinformatics and Computational Biology*, 15(04):1750017, 2017.

George C Linderman and Stefan Steinerberger. Clustering with t-SNE, provably. *arXiv*, 2017.

George C Linderman, Manas Rachh, Jeremy G Hoskins, Stefan Steinerberger, and Yuval Kluger. Efficient algorithms for t-distributed stochastic neighborhood embedding. *arXiv*, 2017.

Evan Z Macosko, Anindita Basu, Rahul Satija, James Nemesh, Karthik Shekhar, Melissa Goldman, Itay Tirosh, Allison R Bialas, Nolan Kamitaki, Emily M Martersteck, et al. Highly parallel genome-wide expression profiling of individual cells using nanoliter droplets. *Cell*, 161(5):1202–1214, 2015.

Leland McInnes and John Healy. Umap: Uniform manifold approximation and projection for dimension reduction. *arXiv*, 2018.

Nicola Pezzotti, Thomas Höllt, B Lelieveldt, Elmar Eisemann, and Anna Vilanova. Hierarchical stochastic neighbor embedding. *Computer Graphics Forum*, 35(3):21–30, 2016.

Nicola Pezzotti, Boudewijn PF Lelieveldt, Laurens van der Maaten, Thomas Höllt, Elmar Eisemann, and Anna Vilanova. Approximated and user steerable tSNE for progressive visual analytics. *IEEE Transactions on Visualization and Computer Graphics*, 23(7):1739–1752, 2017.

Nicola Pezzotti, Alexander Mordvintsev, Thomas Hollt, Boudewijn PF Lelieveldt, Elmar Eisemann, and Anna Vilanova. Linear tSNE optimization for the Web. *arXiv*, 2018.

Jean-Francois Poulin, Bosiljka Tasic, Jens Hjerling-Leffler, Jeffrey M Trimarchi, and Rajeshwar Awatramani. Disentangling neural cell diversity using single-cell transcriptomics. *Nature Neuroscience*, 19 (9):1131, 2016.

Rickard Sandberg. Entering the era of single-cell transcriptomics in biology and medicine. *Nature Methods*, 11(1):22, 2014.

Arpiar Saunders, Evan Macosko, Alec Wysoker, Melissa Goldman, Fenna Krienen, Elizabeth Bien, Matthew Baum, Shuyu Wang, Aleks Goeva, James Nemesh, et al. A single-cell atlas of cell types, states, and other transcriptional patterns from nine regions of the adult mouse brain. *bioRxiv*, 2018.

Karthik Shekhar, Sylvain W Lapan, Irene E Whitney, Nicholas M Tran, Evan Z Macosko, Monika Kowalczyk, Xian Adiconis, Joshua Z Levin, James Nemesh, Melissa Goldman, et al. Comprehensive classification of retinal bipolar neurons by single-cell transcriptomics. *Cell*, 166(5):1308–1323, 2016.

Jian Tang, Jingzhou Liu, Ming Zhang, and Qiaozhu Mei. Visualizing large-scale and high-dimensional data. In *Proceedings of the 25th International Conference on World Wide Web*, pages 287–297. International World Wide Web Conferences Steering Committee, 2016.

Bosiljka Tasic, Vilas Menon, Thuc Nghi Nguyen, Tae Kyung Kim, Tim Jarsky, Zizhen Yao, Boaz Levi, Lucas T Gray, Staci A Sorensen, Tim Dolbeare, et al. Adult mouse cortical cell taxonomy revealed by single cell transcriptomics. *Nature Neuroscience*, 19(2):335, 2016.

Bosiljka Tasic, Zizhen Yao, Kimberly A Smith, Lucas Graybuck, Thuc Nghi Nguyen, Darren Bertagnolli, Jeff Goldy, Emma Garren, Michael N Economo, Sarada Viswanathan, et al. Shared and distinct transcriptomic cell types across neocortical areas. *bioRxiv*, 2017.

The Tabula Muris Consortium. Single-cell transcriptomics of 20 mouse organs creates a *Tabula Muris*. *Nature*, 2018.

Vincent Unen, Thomas Höllt, Nicola Pezzotti, Na Li, Marcel JT Reinders, Elmar Eisemann, Frits Koning, Anna Vilanova, and Boudewijn PF Lelieveldt. Visual analysis of mass cytometry data by hierarchical stochastic neighbour embedding reveals rare cell types. *Nature Communications*, 8(1):1740, 2017.

Laurens van der Maaten. Learning a parametric embedding by preserving local structure. In *Artificial Intelligence and Statistics*, pages 384–391, 2009.

Laurens van der Maaten. Accelerating t-SNE using tree-based algorithms. *Journal of Machine Learning Research*, 15(1):3221–3245, 2014.

Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-SNE. *Journal of Machine Learning Research*, 9(Nov):2579–2605, 2008.

Martin Wattenberg, Fernanda Viégas, and Ian Johnson. How to use t-SNE effectively. *Distill*, 2016.

F Alexander Wolf, Philipp Angerer, and Fabian J Theis. SCANPY: large-scale single-cell gene expression data analysis. *Genome Biology*, 19(1):15, 2018.

Scott A Yuzwa, Michael J Borrett, Brendan T Innes, Anastassia Voronova, Troy Ketela, David R Kaplan, Gary D Bader, and Freda D Miller. Developmental emergence of adult neural stem cells as revealed by single-cell transcriptional profiling. *Cell Reports*, 21(13):3970–3986, 2017.

Amit Zeisel, Hannah Hochgerner, Peter Lonnerberg, Anna Johnsson, Fatima Memic, Job van der Zwan, Martin Haring, Emelie Braun, Lars Borm, Gioele La Manno, et al. Molecular architecture of the mouse nervous system. *Cell*, 174(4):999–1014, 2018.

Grace XY Zheng, Jessica M Terry, Phillip Belgrader, Paul Ryvkin, Zachary W Bent, Ryan Wilson, Solongo B Ziraldo, Tobias D Wheeler, Geoff P McDermott, Junjie Zhu, et al. Massively parallel digital transcriptional profiling of single cells. *Nature Communications*, 8:14049, 2017.