Computational analysis

1) For each cell(column), we quantified the number of genes for which at least one read was mapped, and then excluded all cells with fewer than 800 detected genes.

2) Expression values Ei,j for gene i in cell j were calculated by dividing UMI counts for gene i by the sum of the UMI counts in cell j, to normalize for differences in coverage, and then multiplying by 10,000 to create TPM-like values, and finally computing log2(TPM + 1).

3) Batch correction was performed using ComBat46 as implemented in the R package sva47, using the default parametric adjustment mode. The output was a corrected expression matrix, which was used as an input to further analysis.

4) Selection of variable genes was performed by fitting a generalized linear model to the relationship between the squared coefficient of variation and the mean expression level in logarithmic space, and selecting genes that deviated significantly ($P < 0.05$) from the fitted curve48.

5) Dimensionality reduction using PCA and t-SNE.
We restricted the expression matrix to the subsets of variable genes and high-quality cells noted above, and then centred and scaled values before inputting them into principal component analysis (PCA) For the droplet-based dataset we used a randomized approximation to PCA, implemented using the rpca function from the rsvd R package, with the parameter k set to 100.

6) After PCA, significant principal components were identified using the permutation test51, implemented using the permutationPA function from the jackstraw R package. This test identified 13 and 15 significant principal components in the 10X and SMART-Seq2 datasets of Fig. 1b and Extended Data Fig. 2a, respectively. Scores from only these significant principal components were used as the input to further analysis.

7)For visualization, the dimensionality of the datasets was further reduced using the 'Barnes-hut' approximate version of t-SNE52,53. This was implemented using the Rtsne function from the Rtsne R package using 20,000 iterations and a perplexity setting that varied from 10 to 30 depending on the size of the dataset.

8) Cluster analysis
The nearest-neighbour graph was computed using the function nng from the R package cccd with k=0 and Euclidean distance. The k-nearest-neighbour graph was then used as the input to Infomap9, implemented using the infomap.community function from the igraph R package.

Ref:https://www.nature.com/articles/nature24489#methods