UC Berkeley
UC Berkeley Previously Published Works

Title
Identification of Cell Types from Single-Cell Transcriptomic Data

Permalink
https://escholarship.org/uc/item/07q571pm

ISBN
978-1-4939-9056-6

Authors
Shekhar, Karthik
Menon, Vilas

Publication Date
2019

DOI
10.1007/978-1-4939-9057-3_4

Peer reviewed

eScholarship.org Powered by the California Digital Library
University of California

Chapter 4

Identification of Cell Types from Single-Cell Transcriptomic Data

Karthik Shekhar and Vilas Menon

Abstract

Unprecedented technological advances in single-cell RNA-sequencing (scRNA-seq) technology have now made it possible to profile genome-wide expression in single cells at low cost and high throughput. There is substantial ongoing effort to use scRNA-seq measurements to identify the "cell types" that form components of a complex tissue, akin to taxonomizing species in ecology. Cell type classification from scRNA-seq data involves the application of computational tools rooted in dimensionality reduction and clustering, and statistical analysis to identify molecular signatures that are unique to each type. As datasets continue to grow in size and complexity, computational challenges abound, requiring analytical methods to be scalable, flexible, and robust. Moreover, careful consideration needs to be paid to experimental biases and statistical challenges that are unique to these measurements to avoid artifacts. This chapter introduces these topics in the context of cell-type identification, and outlines an instructive step-by-step example bioinformatic pipeline for researchers entering this field.

Key words Single-cell RNA-sequencing, Transcriptomic classification, Cell-type identification, Cell taxonomy, Clustering, Unsupervised machine learning, Cross-species comparison of cell-types

1 Introduction

The human body contains approximately 40 trillion cells, which exhibit a breathtaking diversity of form and function [1]. Classifying these cells into "types" is increasingly viewed as a foundational requirement to gain a detailed understanding of how tissues function and interact, and to uncover specific mechanisms that underlie pathological states [2]. Provisionally, cells of a particular type share a common identity defined by multiple, measurable properties pertaining to tissue location, function, signaling properties, morphology, electrophysiological response, molecular composition, and physicochemical interaction with other cell types (see below). Knowledge of what cell types exist and what features distinguish them will (a) facilitate genetic access to specific types so they can be labeled and manipulated in model organisms and culture systems,

Guo-Cheng Yuan (ed.), *Computational Methods for Single-Cell Data Analysis*, Methods in Molecular Biology, vol. 1935, https://doi.org/10.1007/978-1-4939-9057-3_4, © Springer Science+Business Media, LLC, part of Springer Nature 2019

(b) provide a framework to investigate the staggering cellular heterogeneity that abounds in organisms, (c) provide mechanistic insight into the generation of this heterogeneity during early development, (d) provide a framework for rationally improving in vitro-derived cell types, (e) facilitate cross-species comparisons [3], and (f) implicate roles for specific cell types and their interactions [4] in complex diseases [5, 6].

Although the genomes of complex mammals contain ~30,000 genes (and their multiple isoforms), the expression patterns of these genes are not all independent of each other. Gene regulatory processes induce correlations between the expression levels of genes, which in turn results in a "modular" structure of the transcriptome [7]. One consequence of this modularity is that the molecular states of cells occupy a low dimensional subspace (often referred to as a "manifold") within the full space of gene expression. Advances in single-cell RNA-sequencing (scRNA-seq) technology have enabled cell types to be defined using the transcriptomic state of thousands of individual cells [8–10]. In addition, the development of single-nucleus profiling techniques has allowed for thorough investigations of frozen and banked tissue, including challenging tissues such as adult human brain sections [11, 12]. A flurry of recent work has shown that unbiased classification of single-cell transcriptomes using computational methods rooted in clustering and dimensionality reduction not only recovers classically defined subsets of cells, but also enables discovery of novel types with unknown functional roles [13–15]. Our goal is to introduce the reader to the conceptual [16] and computational [17] challenges of scRNA-seq data analysis, followed by a description of a basic practical workflow of scRNA-seq analysis using the R statistical language.

1.1 What Is a Cell Type?

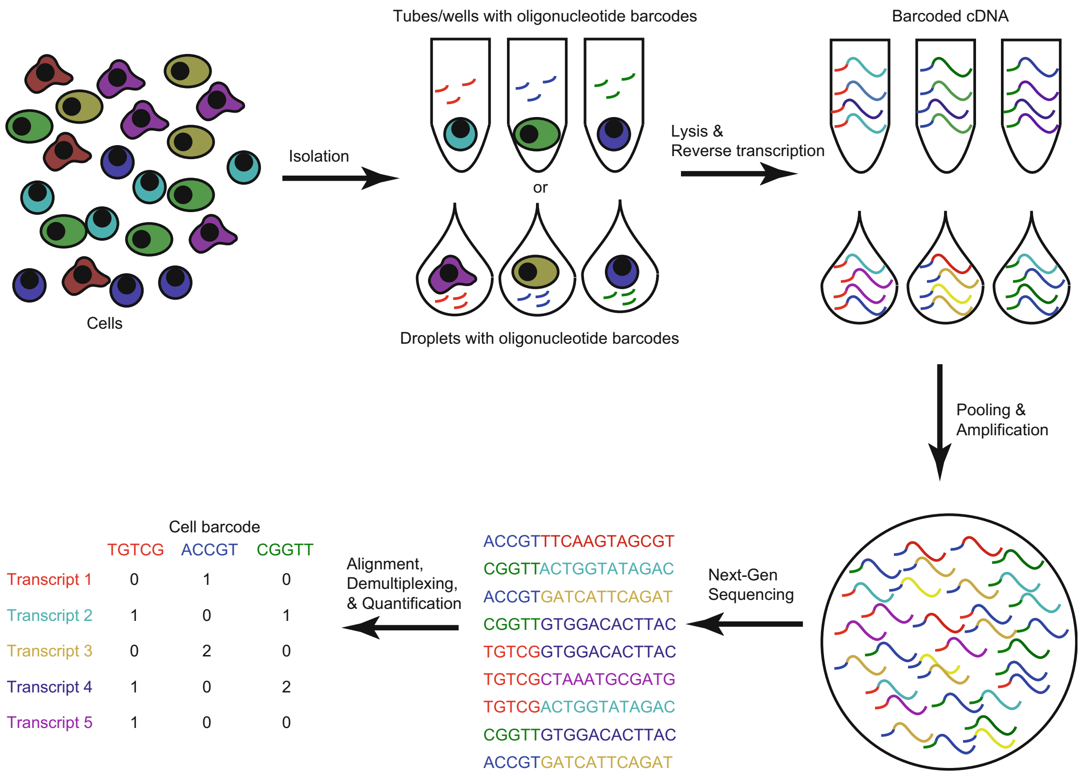
While every cell is unique, experience of biologists over many years has suggested that cells can be organized into groups based on shared features that are quantifiable. This categorization makes possible systematic and reproducible analyses of complex tissues, similar to the concept of "species," which greatly simplifies the diversity of organisms into an interpretable taxonomy, while not denying the individuality of any single member [18]. Features used to define cell types include lineage, location, morphology, activity, interactions with other cell types, epigenetic state, responsiveness to certain signals, and molecular composition (including mRNA and protein levels) [16].

scRNA-seq-based cell classification involves partitioning the data into "clusters" of single cells, wherein each cluster is defined by a unique gene expression "signature" relative to other clusters, and therefore, represents a putative cell type. It must be noted, however, that a computationally defined cluster may not necessarily correspond 1:1 to a cell type, as the molecular state of the cell assayed by scRNA-seq may not necessarily reflect all of the features noted above. Moreover, certain molecular attributes are more transient than others during the lifetime of a cell, necessitating a distinction between a cell's type (its principal identity) and its current "state" (e.g., temporary changes in firing rate of neuron during "up" and "down" states, or different levels of secretory activity of endocrine cells). scRNA-seq may resolve different "states" of the same cell type if their transcriptional signatures are sufficiently distinct, and collapse two distinct, but closely related types if the molecules that specified their identities during early development are no longer expressed during the stage of the experiment.

Thus, even when restricted to the molecular state, the difference between cell "types" and "states" is not resolvable through RNA-seq alone, and may require examination in other modalities, such as those that capture information about the cell's epigenetic state or its dynamical responses. Taken together, these caveats warrant caution in the interpretation of scRNA-seq data, especially in the context of identifying cell types exclusively from transcriptomic information. The notion of cell types is being constantly refined through ongoing work as part of large-scale projects such as the Human Cell Atlas [2] and the BRAIN initiative [19].

1.2 A Brief Overview of scRNA-Seq

scRNA-seq is not a single method, but a suite of protocols, each with its strengths and limitations [20]. Currently, every scRNA-seq protocol consists of three steps (Fig. 1): (1) single-cell capture and barcoding, (2) library preparation, and (3) sequencing. Current protocols isolate single cells by tissue dissociation, followed by either fluorescence- activated cell sorting (FACS) into separate wells on a plate or capturing individual cells in microfluidic chambers, microwells, or individual droplets. Prior to single-cell capture, the dissociated cells can be optionally taken through a sorting step using FACS or magnetic activated cell-sorting (MACS) to enrich or deplete cells expressing a specific combination of markers. Library preparation involves reverse transcribing mRNA into cDNA and amplifying, either using polymerase chain reaction (PCR) or in vitro transcription (IVT). Recently developed protocols tag transcripts during the capture stage (step 1 above) with unique molecular identifiers (UMIs), which are random nucleotide sequences [21]. Every captured transcript is, in principle, tagged with a distinct UMI, which enables downstream correction of amplification biases. The amplified cDNA is then fragmented, followed by the addition of molecular adapters at the end of amplicon fragments that allow for high-throughput sequencing. Libraries can either retain the full length of every transcript or tag either the $3'$ or the $5'$ end of each mRNA—the choice is informed by further considerations. Sequencing is generally highly multiplexed, and can either be single-end or paired-end depending on upstream choices. An important consideration can be the depth of sequencing per cell, which is often related to the number of cells profiled [22].



ACCGTTTCAAGTAGCGT
CGGTTACTGGTATAGAC
ACCGTGATCATTCAGAT
CGGTTGTGGACACTTAC
TGTCGGTGGACACTTAC
TGTCGCTAAATGCGATG
TGTCGACTGGTATAGAC
CGGTTGTGGACACTTAC
ACCGTGATCATTCAGAT

	Cell barcode		
	TGTCG	ACCGT	CGGTT
Transcript 1	0	1	0
Transcript 2	1	0	1
Transcript 3	0	2	0
Transcript 4	1	0	2
Transcript 5	1	0	0

Fig. 1 General experimental workflow for single-cell RNA-sequencing, as described in detail in the text, starting from cell isolation and extending through to the generation of counts tables showing the detection of each gene in each cell

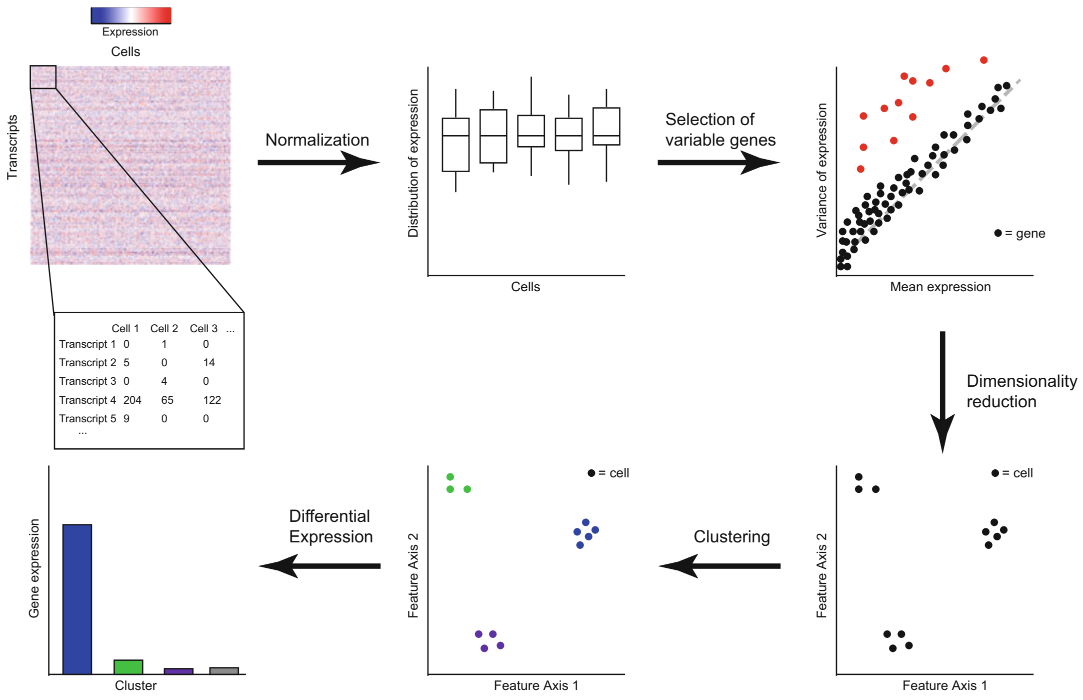
1.3 Batch Effects in scRNA-Seq Analysis

Data-driven identification of cell types can be confounded by batch effects, which result from minor, but systematic differences between experimental replicates prepared either at different times, using different reagent batches, different experimenters, or a combination of the three [23]. Batch effects can result in variation in the transcriptomic state of identical cell types across different replicates due to technical factors; when such effects are strong, cells can cluster by batch identity rather than biological identity. Batch effects can also arise if in addition to transcriptional differences, the frequencies of specific cell types are different across batches [24, 25]. If different biological conditions of interest (e.g., control vs. perturbation) or different sample sources (e.g., biopsies from cancer patients) are processed in different batches, it is statistically impossible to deconvolve biological versus technical effects. While batch effects can be mitigated through careful experimental design involving an even distribution of different biological conditions across experimental batches ("block design"), although this may not always be logistically feasible if delays in sample processing can compromise quality. In such circumstances, cell-types and molecular signals identified in a single experimental batch must be treated with suspicion and results should only be believed if they are supported across multiple independent replicates, or in other data modalities. Detecting and correcting batch effects is an ongoing area of computational innovation, and a number of approaches have been recently proposed [24–26].

Future promising avenues of research involve the integration of scRNA-seq data directly with other data modalities. In particular, recent developments linking RNA-seq to spatial location (such as FISSEQ [27] and "Spatial Transcriptomics" [28]), combined with the advent of high-resolution and expansion microscopy, are on the verge of collecting transcriptome-wide information at the single-cell level in situ, without the need for cell dissociation. Besides removing any dissociation-related biases in cell type or transcript, integration of transcriptomics and spatial location would create tissue-based atlases of cell types, providing an unbiased version of highly multiplexed in situ hybridization methods [29, 30]. Similarly, other cross-modality technologies are also at various stages of maturation: these include linking single-cell RNA-seq with electrophysiological measurements (Patch-Seq [31]), gene perturbations (CRISPR-Seq and Perturb-Seq [32]), protein expression (CITE-Seq [33]), and lineage tracing (MEMOIR [34], scGESTALT [35]). The large-scale use of all of these technologies, as well as others, is on the horizon, and will result in new multi-modal classification and characterization of cell types in complex tissues. Ultimately, the power of single-cell transcriptomics, and its associated computational methods, will continue to progress as a key component in generating new hypotheses about the organization, regulation, and function of complex tissues. Despite all of these developments, the underlying approach to scRNA-seq data analysis for cell type identification still rests on a basic framework, described below.

2 Methods

The following workflow (overview in Fig. 2) describes basic computational steps for identifying molecularly distinct cell types from single-nucleus (sn) RNA-seq data. It does not, however, cover any of the steps relating to the preprocessing, alignment, and quantification of raw sequencing data, which have been described elsewhere [36, 37]. We use the R programming language (https://www.r-project.org), which is a versatile platform for many kinds of genomic analyses, and benefits from the availability of a wide array of statistical and bioinformatic libraries. Over the years, a number of software packages have been developed for single-cell transcritpomic analysis (https://github.com/seandavi/awesome-single-cell), and many of them are available through Bioconductor (https://



	Cell 1	Cell 2	Cell 3	...
Transcript 1	0	1	0	
Transcript 2	5	0	14	
Transcript 3	0	4	0	
Transcript 4	204	65	122	
Transcript 5	9	0	0	
...				

Fig. 2 Standard computational workflow for identifying transcriptomic types from single-cell RNA-sequencing data, as described in detail in the text, starting from counts tables through to cluster assignment and differential gene expression identification. Although not every computational approach incorporates all of these steps in this order, most involve variations on this set of procedures

www.bioconductor.org/), an open-source archive of bioinformatic R libraries with an active user community. This workflow predominantly uses the Seurat package [38], an actively maintained set of tools for scRNA-seq analysis (https://satijalab.org/seurat/).

Here, we analyze single-nucleus (sn)RNA-seq data covering human frontal cortex (FC), visual cortex (VC), and cerebellum (CB) [39]. While the main text mostly refers to single "cells," the methods below and the general concepts are equally applicable to snRNA-seq data, and also to other single-cell level measurements, such as epigenomic and protein (e.g., mass cytometry) data (although statistical considerations differ).

Our workflow begins with the gene expression matrix **X,** whose rows correspond to genes, and whose columns represent single cells. Entries of the matrix represent digital counts of reads or transcripts, depending on the scRNA-seq protocol that generated the data. Although our presentation employs a specific example dataset, the steps below can be carried out with any gene expression matrix (Fig. 2). The following steps are implemented in RStudio, a free and open source integrated development environment (IDE) for R.

2.1 Preprocessing: Read the Count Matrix and Setup the Seurat Object

1. First, we load necessary packages. `utilities.R` is a script that contains some custom functions written by the authors for this workflow (https://github.com/karthikshekhar/CellTypeMIMB).

```
library(Seurat)
require(ggdendro)
require(Rmisc)
library(Matrix)
library(MASS)
library(xtable)
library(Matrix.utils)
library(DOSE)
library(reshape2)
library(topGO)
library(randomForest)
source("utilities.R")
```

2. We then read in the individual data matrices corresponding to the FC, VC, and CB downloaded from the Gene Expression Omnibus submission of [39] (NCBI Gene Expression Omnibus, GSE97942) [39]. These are stored in a locally accessible folder named `Data`. Since the majority of the entries of these expression matrices are "0," we immediately convert them to the sparse matrix format using the `Matrix` package to reduce the memory footprint.

```
FrontCor_counts = read.table("Data/GSE97930_FrontalCortex_snDrop-seq_UMI_Count
_Matrix_08-01-2017.txt.gz",
    header = TRUE)
FrontCor_counts = Matrix(as.matrix(FrontCor_counts), sparse = TRUE)
VisCor_counts = read.table("Data/GSE97930_VisualCortex_snDrop-seq_UMI_Count_Ma
trix_08-01-2017.txt.gz",
    header = TRUE)
VisCor_counts = Matrix(as.matrix(VisCor_counts), sparse = TRUE)
Cereb_counts = read.table("Data/GSE97930_CerebellarHem_snDrop-seq_UMI_Count_Ma
trix_08-01-2017.txt.gz",
    header = TRUE)
Cereb_counts = Matrix(as.matrix(Cereb_counts), sparse = TRUE)
```

3. Next, we add a "tissue of origin" tag to the three tissue matrices and bind them into a single matrix. The rows of the final matrix correspond to the union of the genes in each of the three tissue matrices. Genes that are missing in any matrix are assumed to not be expressed. We use the `rBind.fill` function in the `Matrix.utils` package to fill in the missing genes,

```
colnames(FrontCor_counts) = paste0("FrontalCortex_", colnames(FrontCor_counts)
)
colnames(VisCor_counts) = paste0("VisualCortex_", colnames(VisCor_counts))
colnames(Cereb_counts) = paste0("Cerebellum_", colnames(Cereb_counts))
Count.mat_sndrop = Matrix.utils::rBind.fill(t(VisCor_counts), t(FrontCor_count
s),
    fill = 0)
Count.mat_sndrop = Matrix.utils::rBind.fill(Count.mat_sndrop, t(Cereb_counts),
fill = 0)
Count.mat_sndrop = t(Count.mat_sndrop)
```

4. Next, we initialize an S4 R object of the class Seurat. The various downstream computations will be performed on this object.

```
# Initialize the Seurat object with the raw (non-normalized data).  Keep all
# genes expressed in >= 10 cells. Keep all cells with at least 500 detected ge
nes
snd <- CreateSeuratObject(raw.data = Count.mat_sndrop, min.cells = 20, min.gen
es = 300,
    project = "snDropBrain")
```

`snd@raw.data` is a slot in the Seurat object that stores the original gene expression matrix. We can visualize the first 10 rows (genes) and the first 10 columns (cells),

```
snd@raw.data[1:10, 1:10]

## 10 x 10 sparse Matrix of class "dgCMatrix"

##    [[ suppressing 10 column names 'VisualCortex_Ex1_occ17_AAGTGAGTGACC', 'V
isualCortex_Ex1_occ23_CCAGTACGCATC', 'VisualCortex_Ex1_occ24_CCAGGCCTTTCG' ...
]]

##
## A1BG-AS1 . . . . . . . . . .
## A1CF     . . . . . . . . . .
## A2M      . . . . . . . . . .
## A2M-AS1  . . . . . . . . . .
## A2ML1    . 1 . . . . . . . .
## A2ML1-AS1 . . . . . . . 2 . 1
## A2MP1    . . . . . . . . . .
## A3GALT2  . . . . . . . . . .
## A4GALT   . . . . . . . . . .
## AAAS     . . . . . . . . . .
```

5. We then check the dimensions of the normalized expression matrix and the number of cells from each sample. Here `snd@ident` stores the sample ID's of the cells, corresponding here to their brain region of origin.

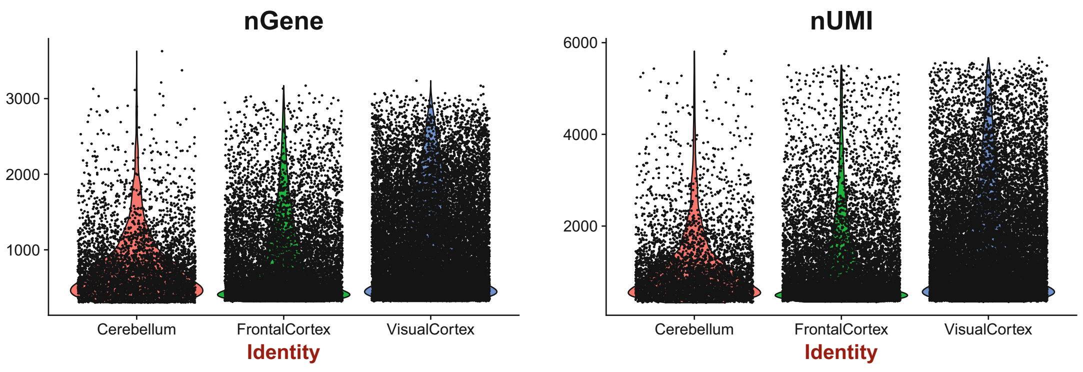


Fig. 3 Sample-wise (*x*-axis) distribution of the number of genes per cell (Left, *y*-axis) and number of UMIs (i.e., transcripts) per cell (Right, *y*-axis) depicted as violin plots. Dots represent individual cells

```
dim(snd@raw.data)

## [1] 23413 34324

table(snd@ident)
##
##    Cerebellum FrontalCortex  VisualCortex
##          4637         10319         19368
```

6. Thus, we have 23,413 genes and 34,234 cells, with 19,368 cells from the VC, 10,319 cells from the FC and 4637 cells from the CB, respectively. We can visualize common metrics such as number of genes per cell (`nGene`) and number of transcripts/UMIs per cell (`nUMI`) as "violin plots" (a fancier version of the good old "box and whisker" plots) using the Seurat plotting command `VlnPlot` (*see* Fig. 3).

```
VlnPlot(object = snd, features.plot = c("nGene", "nUMI"), nCol = 2, point.size
.use = 0.1)
```

2.2 Normalize the Data

1. Because of technical differences in cell-lysis and mRNA capture efficiency, the count vectors of two equivalent cells can differ in the total number of transcripts/UMIs across all genes. This makes it necessary to normalize the data first to attenuate these differences, which is carried out in two steps.

   (a) We rescale the counts in every cell to sum to a constant value. Here, we choose the median of the total transcripts per cell as the scaling factor. This is often referred to as "library-size normalization."

   (b) We apply a logarithmic transformation to the scaled expression values such that $E \leftarrow \log(E+1)$ (the addition of 1 is to ensure that zeros map to zero values). This transformation has two desirable properties,

- It shrinks values such that the data are more uniformly spread across its range of values, which is especially beneficial if there are outliers.
- Since $\log(A) - \log(B) = \log\left(\frac{A}{B}\right)$, it converts distances along a gene-axis to log-fold change values. This has the consequence that expression differences across cells/samples are treated equally, irrespective of the absolute expression value of the gene. This might be especially desirable for lowly expressed genes such as transcription factors.

```
med_trans = median(Matrix::colSums(snd@raw.data[, snd@cell.names]))
print(med_trans)
snd <- NormalizeData(object = snd, normalization.method = "LogNormalize", scal
e.factor = med_trans)
```

2.3 Feature Selection: Identify Highly Variable Genes

1. It is common in analysis of high-dimensional data to choose features that are likely to be informative over features that represent statistical noise, a step known as "Feature Selection." In scRNA-seq data, this is accomplished by choosing genes that are "highly variable" under the assumption that variability in most genes does not represent meaningful biology. An additional challenge is that the level of variability in a gene is related to its mean expression (a phenomenon known as heteroscedacity), which has to be explicitly accounted for. We perform variable gene selection using a recently-published Poisson-Gamma mixture model [40], which was demonstrated to accurately capture the statistical properties of UMI-based scRNA-seq data (Fig. 4).

```
snd = NB.var.genes(snd, do.idents = FALSE, num.sd = 1.2)

## [1] "Identifying variable genes based on UMI Counts. Warning - use this onl
y for UMI based data"
## [1] "Using diffCV = 0.14 as the cutoff"
## [1] "Considering only genes with mean counts less than 3 and more than 0.00
5"
## [1] "Found 1307 variable genes"
print(length(snd@var.genes))

## [1] 1307
```

Thus, we find 1307 variable genes in the data. We refer the reader to other variable gene selection methods, e.g., M3Drop [41], mean-CV regression [42] or Seurat's in-build function `FindVariableGenes`.

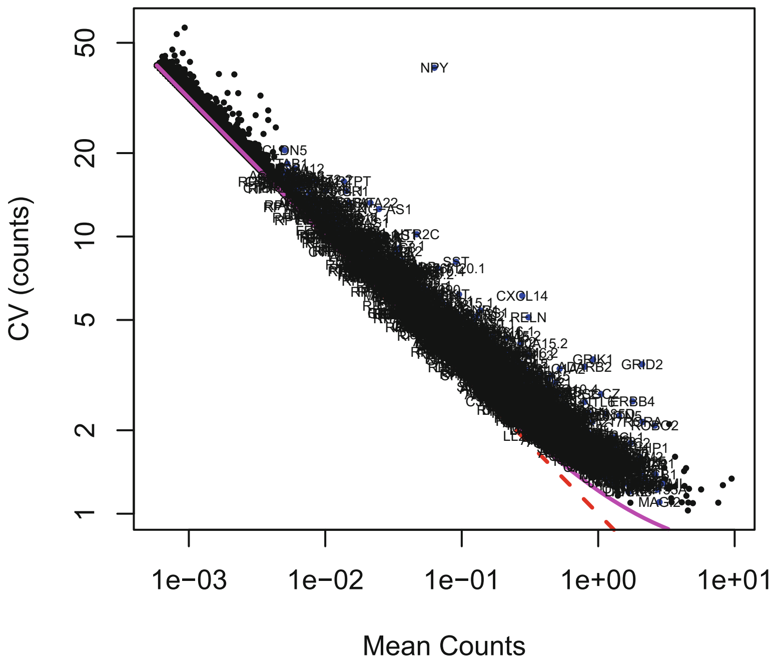


Fig. 4 Mean (*x*-axis) vs. Coefficient of variation (CV, *y*-axis) of genes (dots). Two null-models of mean-CV relationship—Poisson (dashed-red line) or the Poisson-Gamma mixture model—are also plotted

2.4 Z-Score the Data and Remove Unwanted Sources of Variation Using Linear Regression

1. Variation in scRNA-seq data that is relevant to cell identity can be masked by many unwanted sources of variation. A common challenge is batch effects, which can be reflected in both transcriptomic differences and cell-type compositional differences between equivalent experimental batches. As mentioned earlier, variations in lysis efficiency, mRNA capture, and amplification can result in substantial differences between the transcriptomes of equivalent cells. There can be additional sources of variation resulting from biological processes such as cell cycle, response to dissociation, stress, and apoptosis that might dominate the measured transcriptomic state of the cell.

Correcting for such effects continues to be an active area of research, and many sophisticated approaches have been recently introduced [24, 25], but a comprehensive overview is beyond our scope. Here, for demonstrative purposes, we remove variation in gene expression that is highly correlated with library size `nUMI`. Seurat performs a linear fit to the expression level of every gene using `nUMI` as a predictor, and returns the residuals as the "corrected" expression values. Next, the expression values are z-scored or standardized along every gene,

$$E_{ij} \leftarrow \frac{E_{ij} - \bar{E}_i}{\sigma_i}$$

Here E_{ij} is the corrected gene expression value of gene i in cell j, $\bar{E}_i$ and σ_i are the mean and the standard deviation of gene i's expression across all cells. The transformed expression values now have a zero mean and standard deviation equal to 1 across all genes.

2. Removing the effects of `nUMI` and z-scoring are performed together using Seurat's function `ScaleData`, which then stores the transformed gene expression values in the slot `snd@scale.data`.

```
snd <- ScaleData(object = snd, vars.to.regress = c("nUMI"), genes.use = snd@va
r.genes, display.progress = FALSE)
```

2.5 The Curse of Dimensionality and Dimensionality Reduction Using PCA

1. Analysis of high-dimensional scRNA-seq data presents numerous challenges, which are often collectively termed the "curse-of-dimensionality" (COD) [43]. For data that is high-dimensional and noisy, samples from the same and different cell subpopulations (i.e., cell types) can appear equidistant from each other, making it difficult to distinguish variability within types and variability across types. Usually COD is dealt with in two ways (Fig. 2). **First,** the number of features/genes can be filtered to only include highly variable genes, as described in the previous section. **Second**, the data can be projected to a lower dimensional subspace using an algorithm that preserves some important properties of the original data, including gene-gene correlations, a choice that is usually informed by the underlying biological question of interest.

There are multiple approaches to dimensionality reduction, such as principal component analysis (PCA) [44], independent component analysis (ICA) [45], non-negative matrix factorization (NMF) [46], autoencoders, and diffusion maps (DM) [47]. Dimensionality reduction results in the compression of raw gene expression data into fewer "composite" variables, each of which is a complex combination of the original gene features, which may be linear or nonlinear depending on the algorithm. These composite features encode the modular structure of the transcriptome alluded to earlier, and may be interpreted as gene modules or "metagenes," with each metagene being defined by a weighted combination of genes. Each cell's observed expression profile can then be interpreted as an aggregate of each metagene weighted by its activity in that particular cell. A situation where multiple metagenes are active in some cells but not others can result in a separation of cells in gene expression space. In this picture, every cell type is a well-separated cloud of points in the reduced dimensional space, whose location is defined by the activity patterns of gene expression modules.

2. Here, we perform Principal Component Analysis (PCA), a classical and extremely versatile dimensionality reduction method that identifies a linear subspace that most accurately captures the variance in the data [44]. Each of the individual axes of this subspace, known as principal vectors (PVs), are linear combinations of the original genes, and the projections of the original data onto these axes are known as principal components (or PCs.)

```
snd <- RunPCA(object = snd, do.print = TRUE, pcs.print = 1:2, genes.print = 5,
pcs.compute = 50)

## [1] "PC1"
## [1] "KALRN"   "PHACTR1" "PLCB1"   "CHN1"    "KCNQ5"
## [1] ""
## [1] "QKI"    "PLP1"    "CTNNA3" "ST18"    "RNF220"
## [1] ""
## [1] ""
## [1] "PC2"
## [1] "PLP1"    "RNF220"  "MBP"     "MOBP"    "SLC44A1"
## [1] ""
## [1] "ZNF385D" "FSTL5"   "GRID2"   "TIAM1"   "UNC13C"
## [1] ""
## [1] ""
```

Each PV is defined by a set of weights corresponding to the genes (known as the "loadings"). A PV is said to be "driven" by genes with high weights (positive or negative), and two PVs represent independent, orthogonal directions. The printed output of `RunPCA` lists the genes with the highest magnitude loadings (positive and negative) along the top PVs.

2.6 Visualize PCA Output

1. `Seurat` allows multiple ways to visualize the PCA output, and these are useful to gain biological intuition. `VizPCA` shows the genes with the highest absolute loadings along any number of user specified PVs (Fig. 5).

```
VizPCA(object = snd, pcs.use = 1:2)
```

2. `PCAPlot` allows plotting the cells in a reduced dimensional space of PCs, and can often highlight subpopulation structure (Fig. 6).

```
PCAPlot(object = snd, dim.1 = 1, dim.2 = 2, pt.size = 0.4)
```

3. Figures 5 and 6 show that the cells with high values of PC1 are oligodendrocytes, characterized by the high loadings of characteristic genes such as Proteolipid Protein 1 (*PLP1*) and Myelin Basic Protein (*MBP*) (Fig. 5). Next, `PCHeatmap` allows for

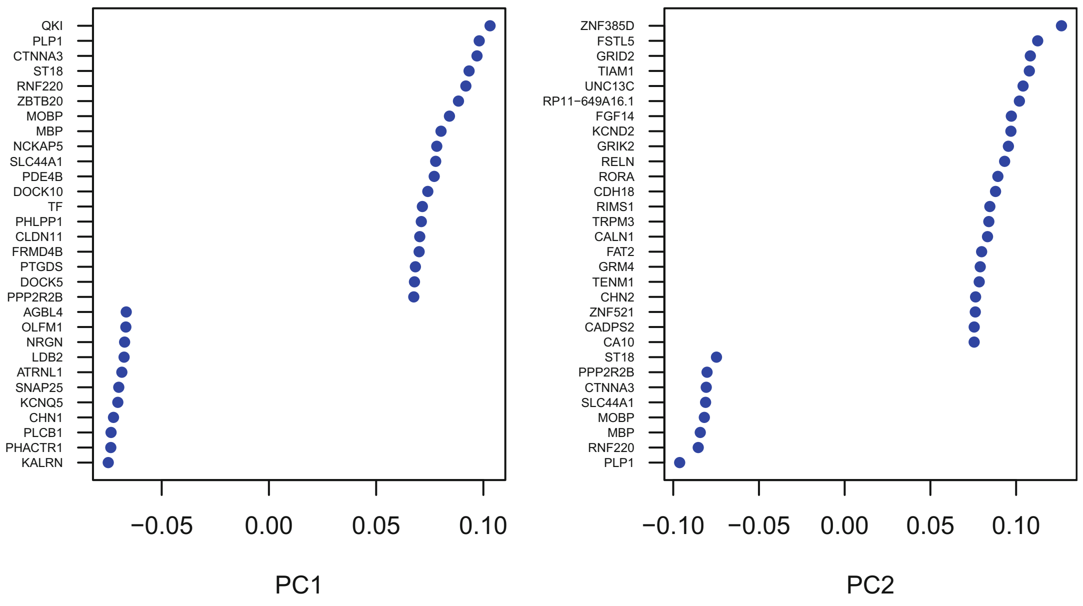


Fig. 5 Genes (*y*-axis) with the highest negative and positive loadings (*x*-axis) for the top two principal components, PC1 and PC2

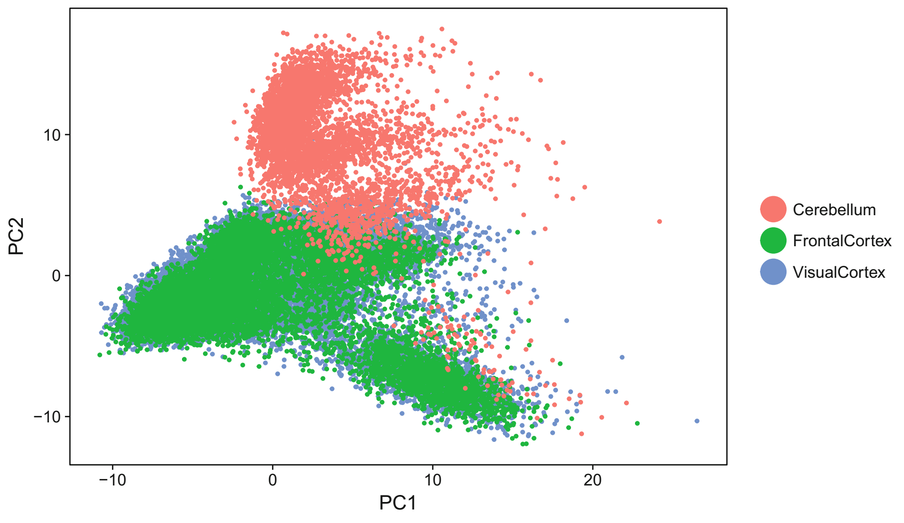


Fig. 6 Scatter plot showing the scores of individual cells (points) along the top two principal components, PC1 and PC2

easy visualization of the gene expression variation along each PC in the data, and can be particularly useful when trying to decide which PCs to include for further downstream analyses (Fig. 7). Both cells and genes are ordered according to their PCA scores and loadings respectively along each PC. Setting `cells.use` to a number plots the "extreme" cells on both ends of the spectrum. For example, here we see that genes with

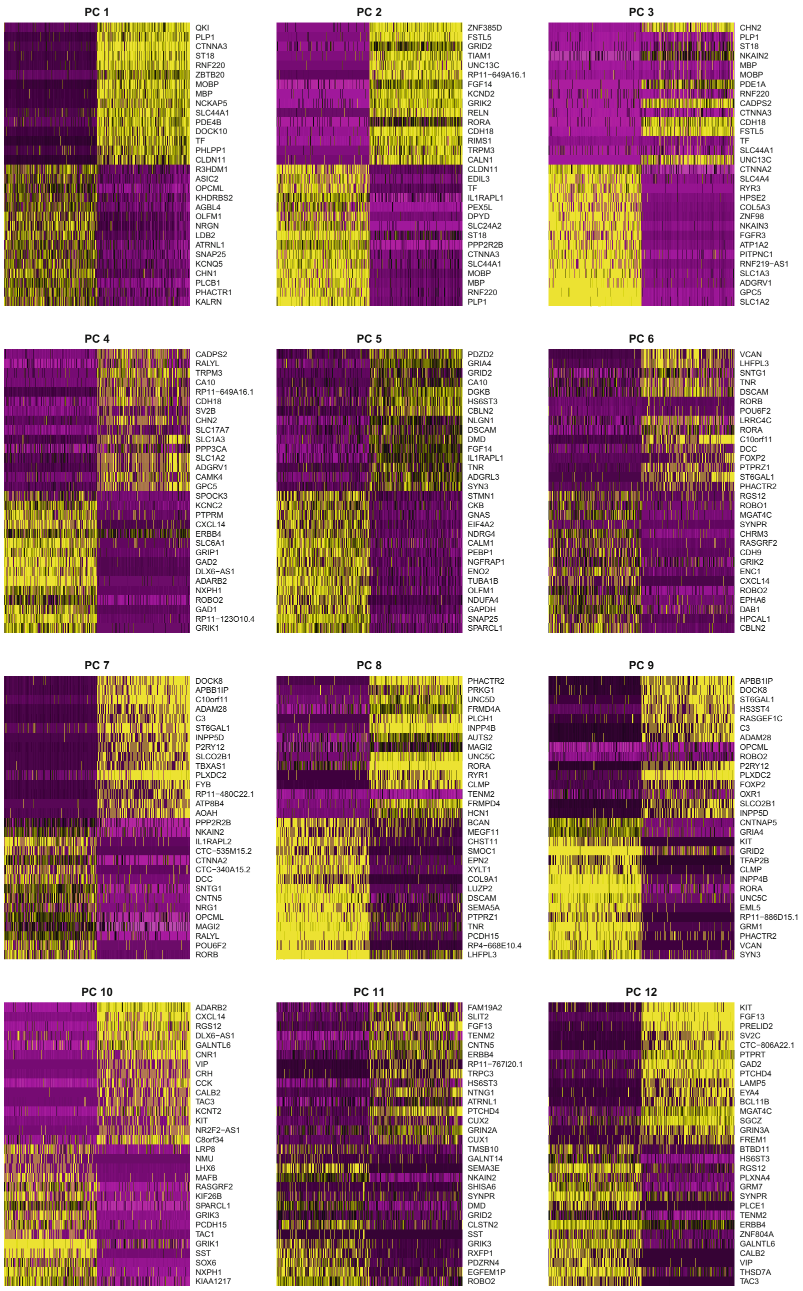


Fig. 7 Heatmaps showing expression of top 15 positive and negative loading genes in individual cells along PC1–PC12

low values of PC3 are astrocytes, characterized by the expression of the transporters *SLC1A2* and *SLC1A3*.

```
PCHeatmap(object = snd, pc.use = 1:12, cells.use = 500, do.balanced = TRUE, la
bel.columns = FALSE, use.full =
```

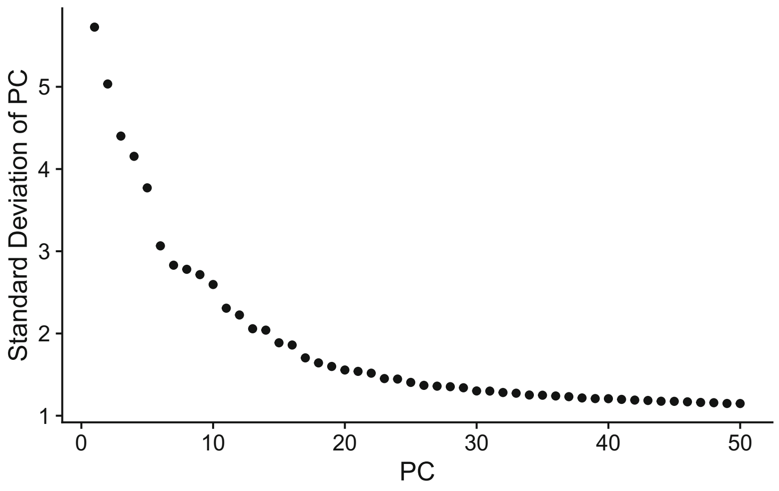


Fig. 8 Standard-deviation (*y*-axis) accounted for by the top 50 PCs (*x*-axis) to approximately identify the number of significant PCs based on the presence of an "elbow." Approximately 25 PCs are chosen for downstream analysis

While there are many formal methods to determine the number of statistically significant PCs (e.g., *see* Shekhar et al., Cell, 2016 [13]), a particularly easy and popular method is to examine the successive reduction in variance captured by increasing PCs, and identify an "elbow" where inclusion of PCs is of marginal utility (this is often called the "noise floor"). We do this using the Seurat function `PCElbowPlot` (Fig. 8).

```
PCElbowPlot(object = snd, num.pc = 50)
```

2.7 Identify Clusters

1. We choose 25 PCs based on Fig. 8. Every cell in the data is thus reduced from ~23,000 genes to 25 PCs (a ~1000 fold reduction in dimensionality!). Next, we determine subpopulations in this data using Graph-based Clustering [48] using the Seurat `FindClusters` function. Graph clustering has been widely used in recently scRNA-seq papers and has many desirable properties compared to other methods such as k-means clustering, hierarchical clustering, and density-based clustering. Here, we first build a k-nearest neighbor graph on the data, connecting each cell to its k-nearest neighbor cells based on transcriptional similarity. The nearest neighbors are determined based on proximity in PC space using a Euclidean distance metric. Next, similar to the strategy employed in Levine et al. [49] and Shekhar et al. [13], the graph edge weights are refined based on the Jaccard-similarity metric, which removes spurious edges between clusters. `FindClusters` implements an algorithm that determines clusters that maximize a mathematical

function known as the "modularity" on the Jaccard-weighted k-nearest neighbor graph. The function contains a `resolution` parameter that tunes the granularity of the clustering, with increased values leading to a greater number of clusters. We use a value of 1, but variations in this parameter need to be tested to check for robustness.

```
# save.SNN = T saves the SNN so that the clustering algorithm can be rerun usi
ng
# the same graph but with a different resolution value (see docs for full
# details)
snd <- FindClusters(object = snd, reduction.type = "pca", dims.use = 1:25, res
olution = 1,
    print.output = 0, save.SNN = TRUE, force.recalc = TRUE)
table(snd@ident)
##
##    0    1    2    3    4    5    6    7    8    9   10   11   12   13   14
## 4058 3642 2877 2214 2061 1942 1932 1806 1805 1462 1432 1248 1232  977  931
##   15   16   17   18   19   20   21   22   23   24   25
##  771  673  654  624  599  390  325  285  215  139   30
```

2. Thus, we obtain 26 clusters in the data. We can visualize the clusters using t-distributed stochastic neighbor embedding (t-SNE) [50], a 2-d embedding of the cells that preserves local distances (Fig. 9). The cells are colored according to the cluster labels,

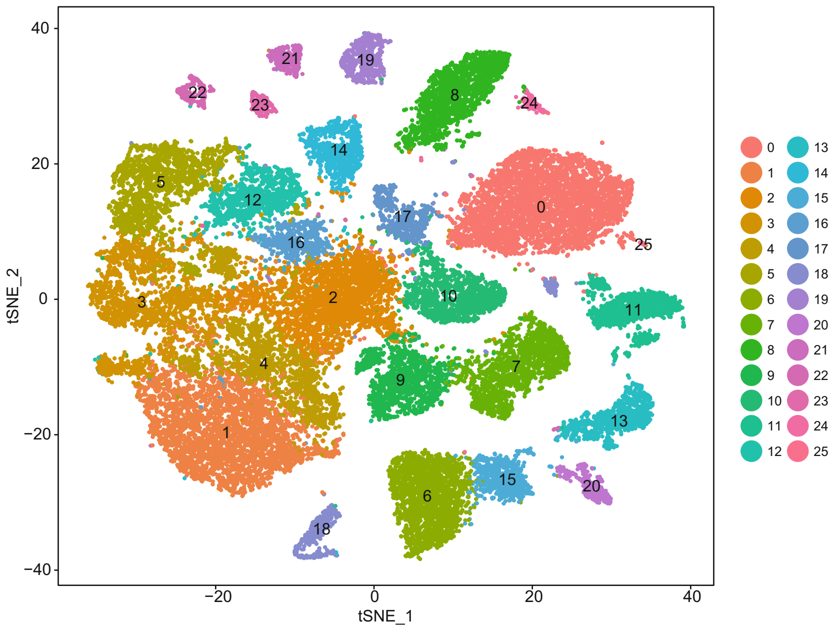


Fig. 9 Visualization of Lake et al. data using t-distributed neighbor embedding (t-SNE). Cells are colored according to their cluster membership

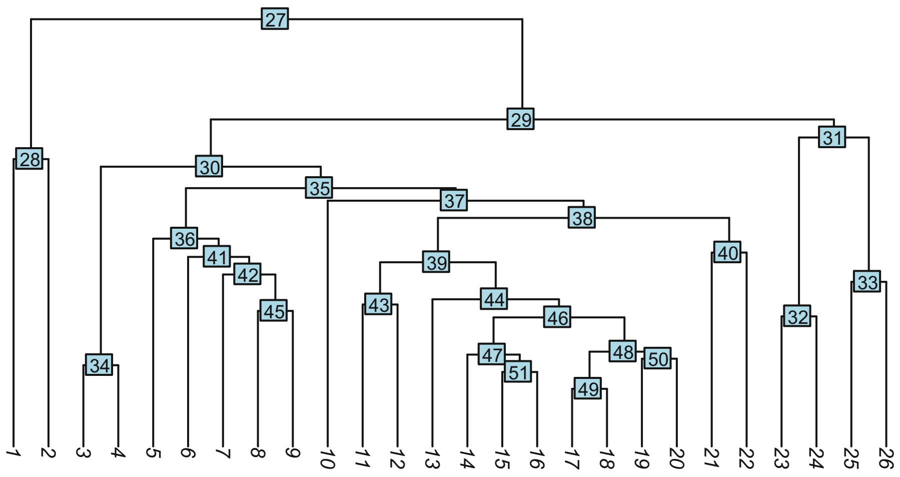


Fig. 10 Dendrogram showing transcriptional relationships between clusters (nodes)

```
snd <- RunTSNE(object = snd, dims.use = 1:25, do.fast = TRUE)
TSNEPlot(object = snd, do.label = TRUE, pt.size = 0.4)
```

3. Next, we arrange the clusters on a dendrogram based on the similarity of their average transcriptomes using Seurat's `BuildClusterTree` function (Fig. 10). This helps in visualizing relationships between clusters, and also reveals subgroups of related clusters.

```
snd <- BuildClusterTree(snd, do.reorder = T, reorder.numeric = T, genes.use =
snd@var.genes, show.progress = FALSE)
```

4. At this point, it is important to note that whether or not we have found the "optimal" number of clusters is open to interpretation. Importantly, the criterion of what constitutes a cell type cluster must be independent of the algorithm's objective—it could be data driven, such as a minimum number of differentially expressed genes enriched in that cluster compared to the rest, or the ability of the algorithm to recover certain well-known types (i.e., ground truth). Often, however, the validation of scRNA-seq clusters requires aligning molecular identity to other cell modalities such as morphology, location, and function through experimental techniques.

Here we adopt a data-driven criterion to assess cluster stability. Briefly, Seurat's `AssessNode` function trains a classifier on each binary node of the dendrogram, and calculates the classification error for left/right clusters. We can use this information to collapse any node that exhibits >15% classification error.

```
node.scores <- AssessNodes(snd)

## Growing trees.. Progress: 74%. Estimated remaining time: 10 seconds.
## Growing trees.. Progress: 49%. Estimated remaining time: 32 seconds.
## Growing trees.. Progress: 73%. Estimated remaining time: 11 seconds.
## Growing trees.. Progress: 98%. Estimated remaining time: 0 seconds.

node.scores <- node.scores[order(node.scores$oobe, decreasing = T), ]
print(head(node.scores))

##    node       oobe
## 20   49 0.11187095
## 19   48 0.09380741
## 21   50 0.07690141
## 16   46 0.07390206
## 17   47 0.05475130
## 5    34 0.03921569
```

2.8 Compare Clusters with Original Cell Type Labels from Lake et al. [39]

1. Here, we see that the maximum "out of bag classification error" (OOBE), is less than our threshold. Thus, we retain all 26 clusters. Next, we compare our clustering result to the cluster labels published in Lake et al. [39], which nominated 33 clusters in their analysis. While we have obviously fewer clusters, it would be interesting to examine how they compare to Lake et al.'s results. We first read in their cluster labels,

```
lake_clusters = unlist(lapply(strsplit(colnames(snd@data), "_"), function(x) x
[2]))
names(lake_clusters) = colnames(snd@data)
length(table(lake_clusters))

## [1] 33

head(table(lake_clusters))

## lake_clusters
##  Ast  End  Ex1  Ex2 Ex3a Ex3b
## 2409  218 5669  310  588 1089
```

Here, `Ast` refers to astrocytes, `End` refers to endothelial cells, `Ex1` refers to Excitatory neuron group 1, and so on. To compare our cluster labels against Lake et al.'s, we plot a "confusion matrix," where each row corresponds to one of Lake et al's 33 clusters, while each column corresponds to our cluster (Fig. 11). The matrix is row-normalized to depict how each cluster of Lake et al. distributes across our clusters.

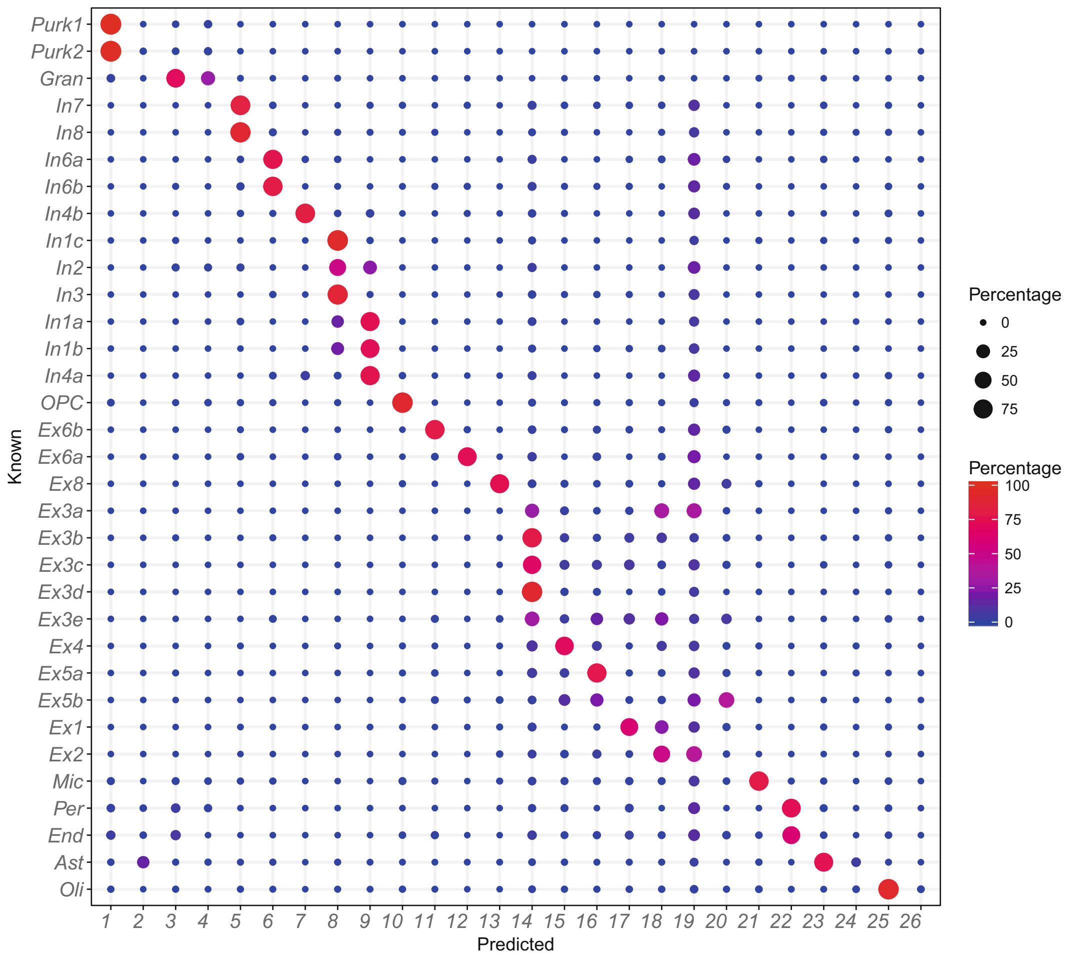


Fig. 11 Transcriptional correspondence between clusters determined from the Lake et al. dataset in this study and in the original study. Circles depict the percentage of cells of a given Lake et al. cluster (row) assigned to a cluster determined above (column)

```
A = table(lake_clusters, snd@ident)
# we post-hoc specify the row order to make the matrix diagonal
row.order = c("Purk1", "Purk2", "Gran", "In7", "In8", "In6a", "In6b", "In4b",
"In1c",
    "In2", "In3", "In1a", "In1b", "In4a", "OPC", "Ex6b", "Ex6a", "Ex8", "Ex3a"
, "Ex3b",
    "Ex3c", "Ex3d", "Ex3e", "Ex4", "Ex5a", "Ex5b", "Ex1", "Ex2", "Mic", "Per",
"End",
    "Ast", "Oli")
a = plotConfusionMatrix(A[row.order, ])
```

2. Encouragingly, we see that although our analysis workflow was agnostic to the results reported in the original paper, many of our clusters exhibit a 1:1 correspondence with the clusters of Lake et al. For example, Cluster 21 ($n = 624$) corresponds to

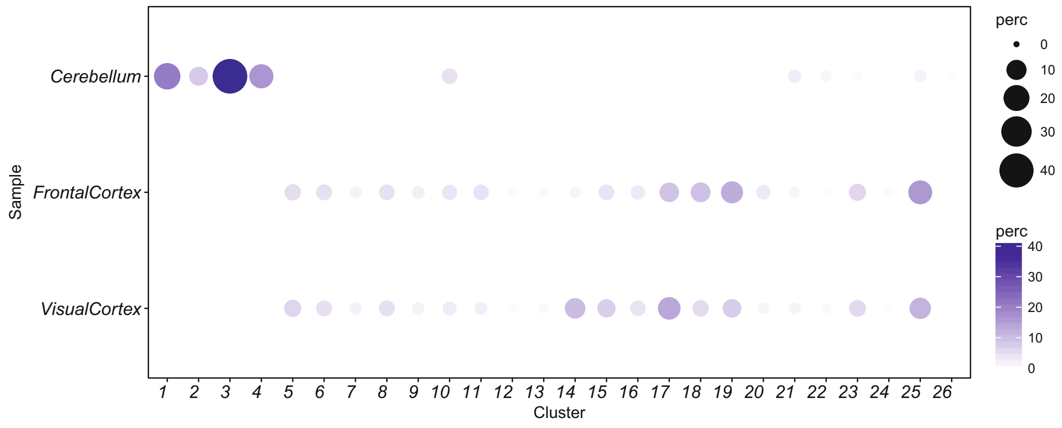


Fig. 12 Cluster composition of each brain region. Circles indicate the proportion of each cluster (columns) within each region (row). Each row sums to 1

Microglia (`Mic`), while Cluster 25 (n = 4058 cells) corresponds to Oligodendrocytes (`Oli`). In cases where multiple Lake et al. clusters map to our clusters, these are related. For example, Purkinje cell clusters `Purk1` and `Purk2` map to Cluster 1 ($n = 977$), while inhibitory neurons In6a and In6b map to Cluster 6 (n = 1462). It is likely that a second round of iterative clustering might be necessary to resolve differences between closely related types such as `In6a` and `In6b`. While all of this is encouraging, we also note some discrepancies—Clusters 2 ($n = 390$), 24 ($n = 139$) and 26 ($n = 30$), do not really correspond to any of the Lake et al., clusters, while clusters 18 ($n = 2061$) and 19 ($n = 2877$) appear to nonspecifically map to many Lake et al. clusters.

3. We can visualize the cluster composition of each of the three brain regions (Fig. 12),

```
# Each row is normalized to a 100%
plot_sample_dist(snd, row.scale = TRUE)
```

As can be seen Clusters 1–4 and 26, which include Purkinje neurons and Cerebellar granule cells, are exclusive to the CB sample, while majority of the remaining clusters are derived from the FC and VC samples.

2.9 Identify Cluster-Specific Differentially Expressed Genes

1. Next, we find cluster-specific markers by performing a differential expression (DE) analysis between each cluster and the rest using Seurat's `FindMarkers` function. `FindMarkers` supports the use of multiple statistical approaches for DE (specified in the `test.use` parameter, *see* Seurat documentation). Here, we use the Student's t-test, as it is computationally efficient. However, we note that there are many limitations to using the t-test for single-cell RNA-seq data, particularly its

inability to account for zero inflation. Readers must explore other methods such as `MAST` and `tweeDEseq` supported by Seurat (for a comprehensive review on DE methods, *see* Soneson and Robinson [51]).

```
# find markers for every cluster compared to all remaining cells, report only
the
# positive ones
snd.markers <- FindAllMarkers(object = snd, only.pos = TRUE, min.pct = 0.25, t
hresh.use = 0.25,
    test.use = "t", max.cells.per.ident = 1000)
head(snd.markers)

##                p_val avg_logFC pct.1 pct.2     p_val_adj cluster   gene
## GRID2   0.000000e+00  2.283338 0.999 0.422  0.000000e+00       1  GRID2
## RORA    0.000000e+00  1.760396 0.991 0.588  0.000000e+00       1   RORA
## INPP4B 4.623397e-308  1.120327 0.878 0.228 1.082476e-303       1 INPP4B
## GRM1   5.438182e-270  0.964801 0.831 0.164 1.273242e-265       1   GRM1
## UNC5C  5.279446e-252  1.026785 0.889 0.339 1.236077e-247       1  UNC5C
## SYN3   1.012546e-197  0.825126 0.820 0.308 2.370674e-193       1   SYN3
```

2. The output is a `data.frame` object summarizing the cluster-specific markers. Here, each row is a gene that is enriched in a cluster indicated in the column `cluster`. `pct.1` is the proportion of cells in the cluster that express this marker, while `pct.2` is the proportion of cells in the background that express this marker. We can examine markers for a given cluster as follows,

```
clust = 25  # Oligodendrocytes
head(subset(snd.markers, cluster == clust & pct.2 < 0.1))

##                 p_val avg_logFC pct.1 pct.2     p_val_adj cluster    gene
## PLP11    0.000000e+00 1.6572808 0.873 0.084  0.000000e+00      25    PLP1
## MOBP1   2.972597e-177 1.0967866 0.660 0.057 6.959740e-173      25    MOBP
## TF      7.315453e-133 0.8444486 0.513 0.039 1.712767e-128      25      TF
## FRMD4B1 2.760596e-125 0.7469469 0.519 0.084 6.463384e-121      25  FRMD4B
## CLDN11  1.497425e-117 0.7452380 0.483 0.051 3.505920e-113      25  CLDN11
## TMEM144 4.327291e-105 0.6684642 0.434 0.048 1.013149e-100      25 TMEM144
```

3. As expected, the top two genes are *PLP1* (Proteolipid Protein 1) and *MOBP* (Myelin-Associated Oligodendrocyte Basic Protein), classical markers of Oligodendrocytes. Next, we examine cluster 12 (an excitatory neuronal cluster), which corresponds to `Ex6a`, and is marked by multiple genes including *HTR2C* and *NPSR1-AS1* (Fig. 13).

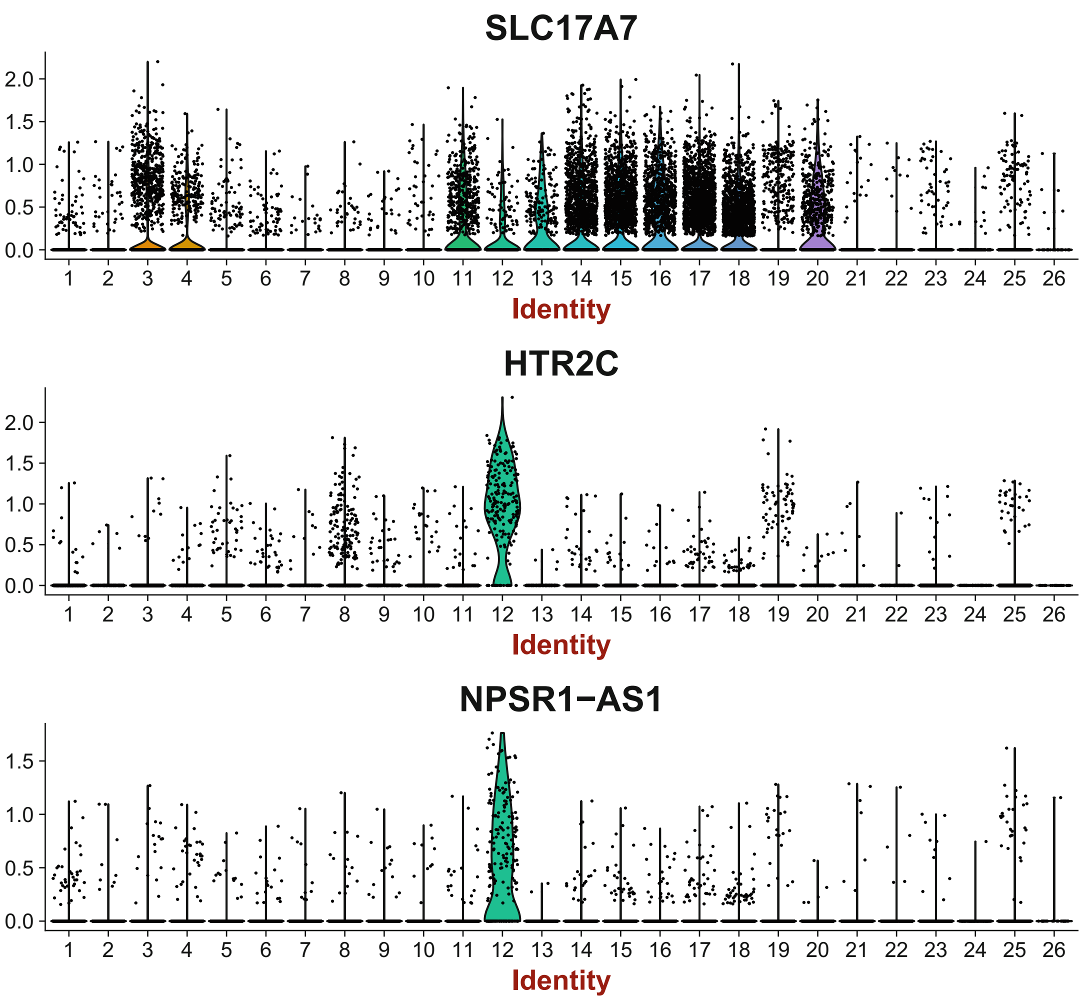


Fig. 13 Cluster-expression of a pan-excitatory neuronal marker *SLC17A7* (top) and markers specific to cluster 12 *HTR2C* (middle) and *NPSR1-AS1* (bottom)

```
head(subset(snd.markers, cluster == 12 & pct.2 < 0.1))

##                      p_val avg_logFC pct.1 pct.2    p_val_adj cluster
## HTR2C         2.715453e-67 1.0524759 0.842 0.018 6.357689e-63      12
## RP11-420N3.3 4.584219e-39 0.6270714 0.702 0.095 1.073303e-34      12
## NPSR1-AS1    4.002079e-37 0.6707752 0.651 0.013 9.370068e-33      12
## IFNG-AS1     2.622851e-35 0.6828199 0.623 0.011 6.140881e-31      12
## PCP4         1.398067e-22 0.4261675 0.535 0.080 3.273294e-18      12
## CRYM         1.012721e-21 0.3684937 0.507 0.068 2.371085e-17      12
##                      gene
## HTR2C               HTR2C
## RP11-420N3.3 RP11-420N3.3
## NPSR1-AS1       NPSR1-AS1
## IFNG-AS1         IFNG-AS1
## PCP4                 PCP4
## CRYM                 CRYM

VlnPlot(snd, c("SLC17A7", "HTR2C", "NPSR1-AS1"), nCol = 1, point.size.use = 0.
01)
```

Examining the identity of these clusters in detail is beyond the scope of this workflow. Readers are encouraged dig deeper, and attempt to test variations in the methods outlined above. We end by demonstrating two common approaches to interpret results: (a) Examining gene-set enrichments, and (b) aligning clusters to alternative datasets.

2.10 Examine Clusters for Enrichment of Biological Processes

1. After identifying markers, we can evaluate whether cluster-specific genes are enriched for any Gene Ontology (GO), Disease Ontology (DO), or Disease Gene Network (DGN) gene lists or categories. Each of these calls has multiple parameters, reflecting stringency of statistical overlap, but they are useful tools to evaluate clusters for functional or disease relevance.

```
#### evaluate for each cluster###
require(org.Hs.eg.db)

## Loading required package: org.Hs.eg.db

##

x = as.list(org.Hs.egALIAS2EG)
geneList = rep(0, nrow(Count.mat_sndrop))
names(geneList) = rownames(Count.mat_sndrop)
geneList = geneList[intersect(names(geneList), names(x))]
newallgenes = names(geneList)
for (ii in 1:length(geneList)) {
    names(geneList)[ii] = x[[names(geneList)[ii]]][1]
}

gene_enrichment_results = list()

for (cl in as.character(unique(snd.markers$cluster))) {
    print(paste0("Running cluster ", cl))
    testgenes = subset(snd.markers, cluster == cl)$gene
    gene_enrichment_results[[cl]] = list()
    #### Run against topGO####
    testgeneList = geneList
    testgeneList[which(newallgenes %in% testgenes)] = 1
    genegene_enrichment_results = list()
    tab1 = c()
    for (ont in c("BP", "MF", "CC")) {
        sampleGOdata <- suppressMessages(new("topGOdata", description = "Simpl
e session",
            ontology = ont, allGenes = as.factor(testgeneList), nodeSize = 10,
annot = annFUN.org,
            mapping = "org.Hs.eg.db", ID = "entrez"))
        resultTopGO.elim <- suppressMessages(runTest(sampleGOdata, algorithm =
"elim",
            statistic = "Fisher"))
        resultTopGO.classic <- suppressMessages(runTest(sampleGOdata, algorith
m = "classic",
            statistic = "Fisher"))
        ## look at results
        tab1 <- rbind(tab1, GenTable(sampleGOdata, Fisher.elim = resultTopGO.e
lim,
            Fisher.classic = resultTopGO.classic, orderBy = "Fisher.elim", top
Nodes = 200))
    }
    gene_enrichment_results[[cl]][["topGO"]] = tab1

    #### Run against DOSE###
    x <- suppressMessages(enrichDO(gene = names(testgeneList)[testgeneList ==
1],
        ont = "DO", pvalueCutoff = 1, pAdjustMethod = "BH", universe = names(t
estgeneList),
        minGSSize = 5, maxGSSize = 500, qvalueCutoff = 1, readable = T))
    gene_enrichment_results[[cl]][["DO"]] = x
    dgn <- suppressMessages(enrichDGN(names(testgeneList)[testgeneList == 1]))
    gene_enrichment_results[[cl]][["DGN"]] = dgn
}

## [1] "Running cluster 1"
## [1] "Running cluster 2"
## [1] "Running cluster 3"
## [1] "Running cluster 4"
## [1] "Running cluster 5"
## [1] "Running cluster 6"
## [1] "Running cluster 7"
## [1] "Running cluster 8"
## [1] "Running cluster 9"
## [1] "Running cluster 10"
## [1] "Running cluster 11"
## [1] "Running cluster 12"
## [1] "Running cluster 13"
## [1] "Running cluster 14"
## [1] "Running cluster 15"
## [1] "Running cluster 16"
## [1] "Running cluster 17"
## [1] "Running cluster 18"
## [1] "Running cluster 19"
## [1] "Running cluster 20"
## [1] "Running cluster 21"
## [1] "Running cluster 22"
## [1] "Running cluster 23"
## [1] "Running cluster 24"
## [1] "Running cluster 25"
## [1] "Running cluster 26"

save(gene_enrichment_results, file = "gene_enrichment_analysis.rda")
```

2. As an example, view the GO, DO, and DGN categories enriched for genes distinguishing cluster 1 (Purkinje Neurons). Note that the categories are arranged by adjusted p-value, and many are not significantly enriched.

```
gene_enrichment_results[["1"]][["topGO"]][1:5, ]

##        GO.ID                                        Term Annotated
## 1 GO:0035235 ionotropic glutamate receptor signaling ...        25
## 2 GO:0071625                      vocalization behavior        17
## 3 GO:0007612                                   learning       131
## 4 GO:1904861                 excitatory synapse assembly        10
## 5 GO:0051965     positive regulation of synapse assembly        63
##   Significant Expected Rank in Fisher.classic Fisher.elim Fisher.classic
## 1           6     0.14                      6     4.1e-09        5.1e-09
## 2           4     0.09                     16     2.0e-06        2.3e-06
## 3           7     0.73                     24     8.4e-06        1.1e-05
## 4           3     0.06                     31     2.0e-05        2.2e-05
## 5           5     0.35                     32     2.6e-05        3.1e-05

gene_enrichment_results[["1"]][["DO"]][1:5, ]

##                         ID                             Description
## DOID:0060037 DOID:0060037 developmental disorder of mental health
## DOID:1827       DOID:1827        idiopathic generalized epilepsy
## DOID:0060041 DOID:0060041                  autism spectrum disorder
## DOID:12849     DOID:12849                        autistic disorder
## DOID:0060040 DOID:0060040        pervasive developmental disorder
##             GeneRatio  BgRatio       pvalue    p.adjust      qvalue
## DOID:0060037     12/57 344/7041 1.474101e-05 0.002046357 0.001921672
## DOID:1827         4/57  20/7041 1.698221e-05 0.002046357 0.001921672
## DOID:0060041      7/57 171/7041 4.187781e-04 0.025231379 0.023694023
## DOID:12849        7/57 171/7041 4.187781e-04 0.025231379 0.023694023
## DOID:0060040      7/57 180/7041 5.704404e-04 0.027495225 0.025819932
##
geneID
## DOID:0060037 AUTS2/CACNA1A/CNTNAP5/GRIA3/MACROD2/NBEA/NLGN1/NOS1AP/NRXN1/SH
ANK2/SYN3/XKR4
## DOID:1827                                                          CACNA1A/GA
D2/HCN1/KCNMA1
## DOID:0060041                              AUTS2/CNTNAP5/MACROD2/NBEA/NLGN
1/NOS1AP/NRXN1
## DOID:12849                                AUTS2/CNTNAP5/MACROD2/NBEA/NLGN
1/NOS1AP/NRXN1
## DOID:0060040                              AUTS2/CNTNAP5/MACROD2/NBEA/NLGN
1/NOS1AP/NRXN1
##             Count
## DOID:0060037    12
## DOID:1827        4
## DOID:0060041     7
## DOID:12849       7
## DOID:0060040     7

gene_enrichment_results[["1"]][["DGN"]][1:5, ]

##                           ID                Description GeneRatio
## umls:C1272641 umls:C1272641  Systemic arterial pressure     19/89
## umls:C1271104 umls:C1271104     Blood pressure finding     18/89
## umls:C0236969 umls:C0236969 Substance-Related Disorders     10/89
## umls:C1510586 umls:C1510586   Autism Spectrum Disorders     10/89
## umls:C0007758 umls:C0007758           Cerebellar Ataxia      6/89
##                  BgRatio       pvalue     p.adjust       qvalue
## umls:C1272641 442/17381 7.549992e-13 3.439175e-10 3.250238e-10
## umls:C1271104 386/17381 8.367822e-13 3.439175e-10 3.250238e-10
## umls:C0236969 167/17381 1.354158e-08 3.710394e-06 3.506557e-06
## umls:C1510586 246/17381 5.129828e-07 1.054180e-04 9.962667e-05
## umls:C0007758 107/17381 1.815118e-05 2.984054e-03 2.820120e-03
##
geneID
## umls:C1272641 105/26053/129684/64478/728215/9758/442117/23072/3778/140733/2
3026/30010/5592/6446/57419/23345/84216/440279/114786
## umls:C1271104      105/26053/129684/64478/728215/9758/442117/23072/3778/140
733/23026/30010/5592/57419/23345/84216/440279/114786
## umls:C0236969                                                          105/
3899/64478/55691/2893/10207/140733/9369/23345/114786
## umls:C1510586                                                           260
53/776/1804/140733/26960/22871/9378/9369/22941/26137
## umls:C0007758
773/2259/2572/2895/2911/23345
##                Count
## umls:C1272641    19
## umls:C1271104    18
## umls:C0236969    10
## umls:C1510586    10
## umls:C0007758     6
```

2.11 Compare with Mouse Cortical Cell Types

1. One of the many challenges in cell type classification studies is that of aligning clusters across different datasets, which might include different batches, different conditions (e.g., normal vs. disease), or even different species. Here we attempt to map clusters from a dataset of visual cortex (VC) neurons isolated and profiled from adult mouse using the Smart-seq method [15] to our Human CB, VC, and FC clusters using a supervised learning algorithm. We use a multiclass classification approach described previously [13].

First, we read in the mouse VC data comprised of 1679 cells and create a *Seurat* S4 object. To match the gene ID's to Human data, we capitalize all gene names—note that a more exact, albeit lengthier approach, would be to match genes based on an appropriate orthology database. We also read in the cluster assignments of each cell. Tasic et al. identified 49 transcriptomic types, comprising 23 inhibitory, 19 excitatory, and 7 non-neuronal types [15]. We next select features to train our classifier. We identify variable genes using Seurat's `FindVariableGenes` function (Fig. 14), which is more appropriate for Smart-seq data [40]. After expanding the set of variable genes in the snRNA-seq data using `NB.var.genes`, we compute the common variable genes to train a multi-class classifier.

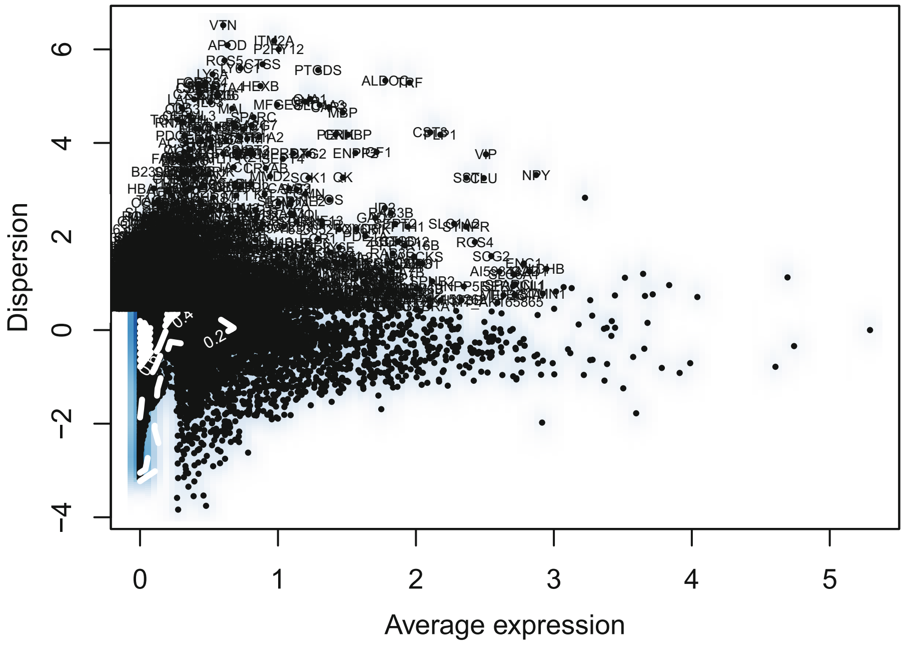


Fig. 14 Identification of highly variable genes using Seurat's `FindVariableGenes` function (see documentation for details). This is an appropriate strategy for feature selection on scRNA-seq that does not contain UMIs

```
tasic_data = Matrix(as.matrix(read.csv("tasic_2016/genes_counts_2.csv", header
= TRUE,
    row.names = 1)), sparse = TRUE)
# Change gene name format to match to Human
rownames(tasic_data) = toupper(rownames(tasic_data))
mouse_vc <- CreateSeuratObject(raw.data = tasic_data, min.cells = 10, min.gene
s = 500,
    project = "tasic")
mouse.clusters = read.csv("tasic_2016/cluster_assignment_simple.csv", row.name
s = 2)
mouse.labels = mouse.clusters$primary
names(mouse.labels) = rownames(mouse.clusters)
mouse_vc@meta.data$type = mouse.labels[rownames(mouse_vc@meta.data)]
mouse_vc = SetAllIdent(mouse_vc, id = "type")
mouse_vc <- NormalizeData(object = mouse_vc, normalization.method = "LogNormal
ize",
    scale.factor = 10000)
mouse_vc <- FindVariableGenes(object = mouse_vc, mean.function = ExpMean, disp
ersion.function = LogVMR, x.low.cutoff = 0.0125, x.high.cutoff = 3, y.cutoff =
0.5, set.var.genes = TRUE)
```

```
var.genes_snd = NB.var.genes(snd, do.idents = FALSE, num.sd = 0.8, do.plot = F
ALSE, set.var.genes = FALSE)

## [1] "Identifying variable genes based on UMI Counts. Warning - use this onl
y for UMI based data"
## [1] "Using diffCV = 0.11 as the cutoff"
## [1] "Considering only genes with mean counts less than 3 and more than 0.00
5"
## [1] "Found 1977 variable genes"

var.genes = intersect(var.genes_snd, mouse_vc@var.genes)
```

2. Next, we train a Random Forest (RF) model [52] on the snRNA-seq data and use that to assign cluster labels to mouse VC data. Given a cell, the classifier maps it to one of 26 clusters. To account for scale differences between the snRNA-seq (3-′-biased, UMI-based) and Smart-seq (full-length, non-UMI-based), we standardize the two datasets (z-score values along each gene). After training it on the snRNA-seq data, we apply this classifier to each cell from the mouse VC data, and assign it to one of 26 snRNA-seq clusters.

```
rf_model = RF_train(snd, var.genes, do.scale = TRUE)

## [1] "Using mininum of 50 percent cells or 700 cells per cluster for trainin
g"
## [1] 13067

pred.labels <- predict(rf_model, t(t(scale(t(mouse_vc@data[var.genes, ])))))
pred.labels <- factor(as.numeric(as.character(pred.labels)) + 1)
names(pred.labels) = colnames(mouse_vc@data)
```

3. How do the cluster assignments compare with the cluster labels obtained from Tasic et al. [15]? Note that the latter labels were not used in any way to either construct the classifier, or to influence the cluster assignment of cells. It would therefore be interesting to see if there is any correspondence between mouse cortical cell types, and their assigned “Human” type based on an unbiased classifier. We examine the confusion matrix, as before (Fig. 15),

```
a = plotConfusionMatrix(table(mouse.labels, pred.labels), order = "Row")
```

The rows correspond to the Tasic et al. clusters, while each column corresponds to an snRNA-seq cluster. The matrix is row-normalized, such that each row adds up to a 100%. First, we see that Clusters 1–4 and 26, which are of Cerebellar origin, receive very few matches from mouse VC data, which largely map to Human clusters originating from VC and FC samples. Among non-neuronal cells, we see that mouse astrocytes and oligodendrocytes map to clusters 23 and 25, which are Human astrocytes and oligodendrocytes, respectively. Inhibitory neuronal groups expressing parvalbumin (*Pvalb*), Somatostatin (*Sst*), and Vasoactive intestinal peptide (*Vip*) map to clusters 6, 5, and 8 respectively. Examining the expression of these markers in the snRNA-seq data validates the RF cluster assignments (Fig. 16). Thus, despite the fact that these two data sets differ in species (human vs. mouse), cell fraction profiled (cytoplasmic vs. nucleus-only), profiling method (Smart-Seq vs. droplet-based sequencing), and clustering method (gene clustering vs. PCA-based methods vs. PCA-Louvain clustering), the overall results are comparable and interpretable, suggesting that the transcriptomic space these cells occupy is being appropriately parsed into subtypes.

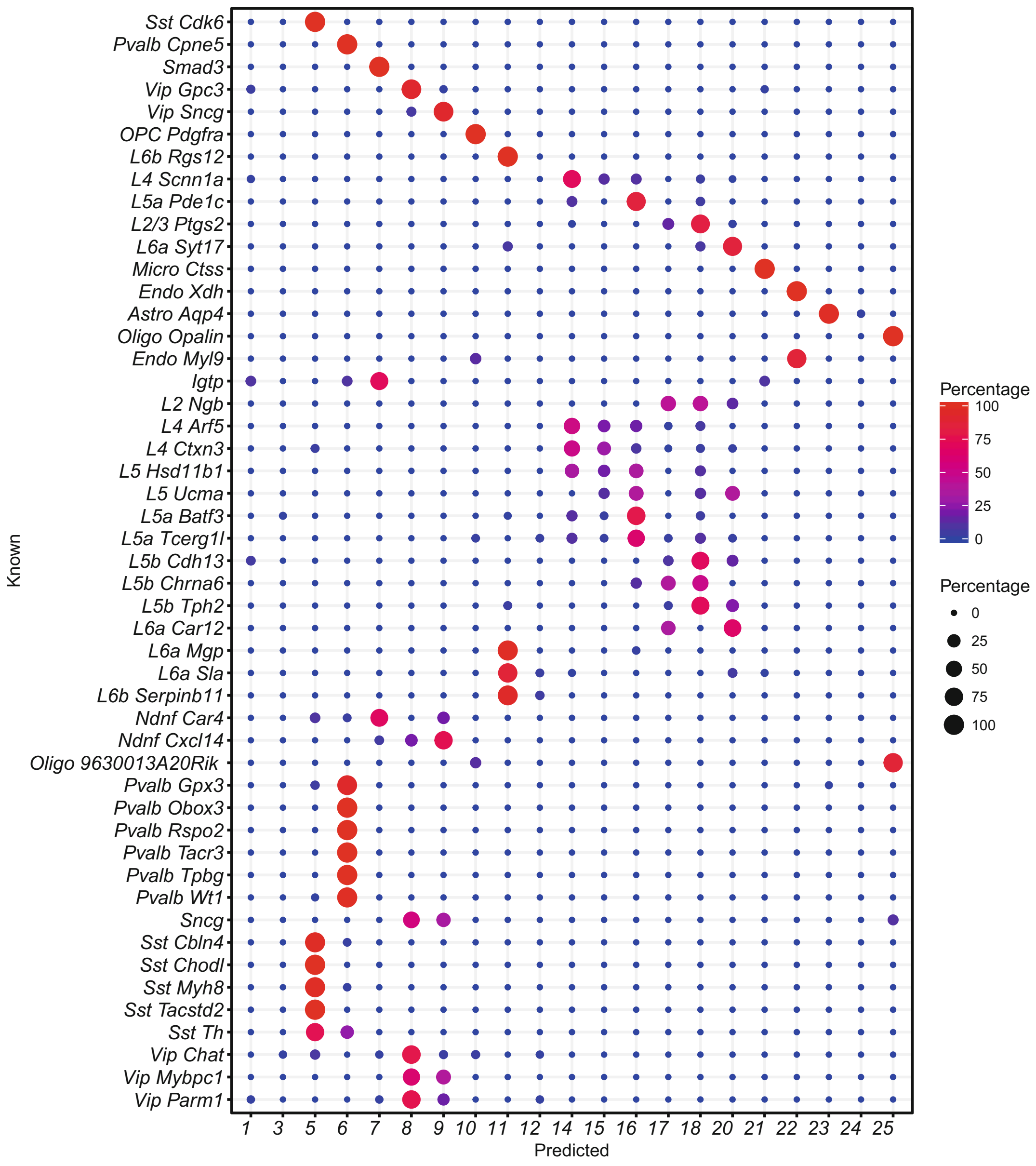


Fig. 15 Transcriptional correspondence between mouse cortical clusters reported in Tasic et al. [15] (rows) and those in this study (columns). Representation as in Fig. 11

```
VlnPlot(snd, c("PVALB", "SST", "VIP"), nCol = 1, point.size.use = 0.01)
```

This concludes the basic workflow. We can save files from the analysis as follows.

```
save(list=c("snd","snd.markers","rf_model","mouse_vc"), file="LakeSeuratAnalys
isObject.Rdata")
```

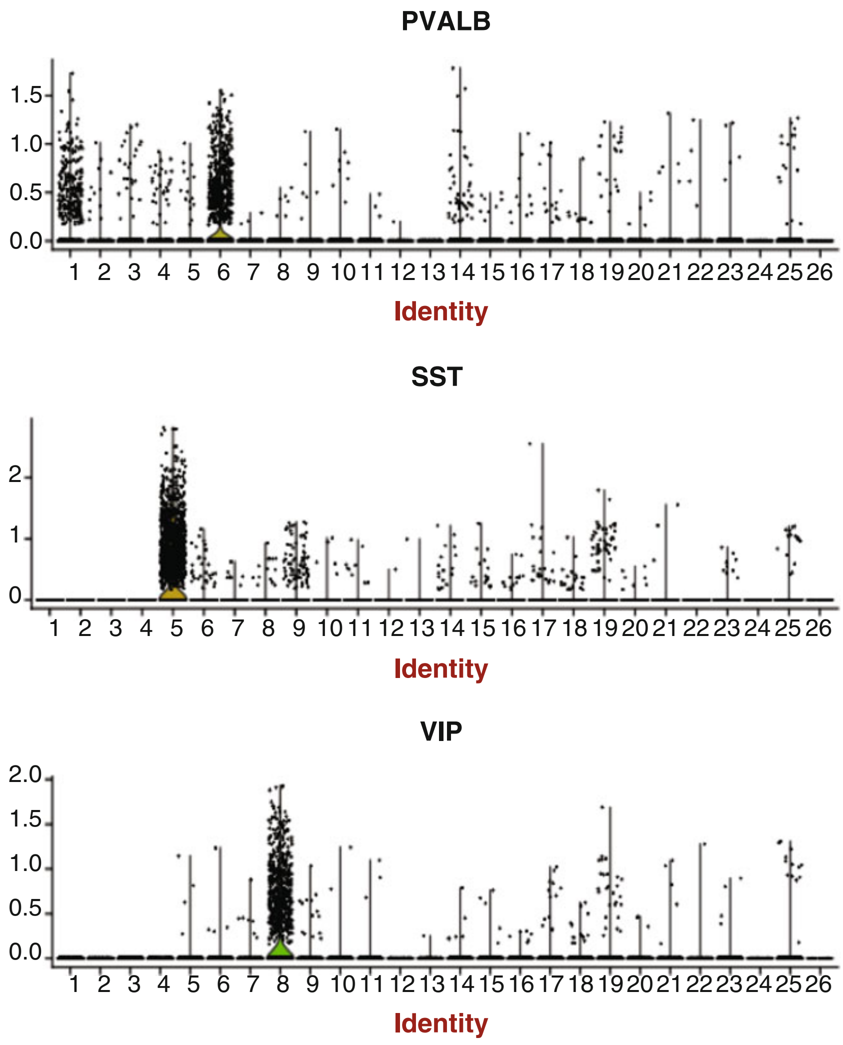


Fig. 16 Expression of three classic markers known to distinguish inhibitory neuronal categories *PVALB* (top), *SST* (middle), and *VIP* (bottom)

Acknowledgments

K. S. would like to acknowledge support from NIH 1K99EY028625-01, the Klarman Cell Observatory, and the laboratory of Dr. Aviv Regev at the Broad Institute. We would like to gratefully acknowledge critical feedback from Drs. Inbal Benhar and Jose Ordovas-Montanes.

References

1. Vickaryous MK, Hall BK (2006) Human cell type diversity, evolution, development, and classification with special reference to cells derived from the neural crest. Biol Rev Camb Philos Soc 81(3):425–455
2. Regev A et al (2017) The human cell atlas. Elife:6
3. Tosches MA et al (2018) Evolution of pallium, hippocampus, and cortical cell types revealed by single-cell transcriptomics in reptiles. Science 360(6391):881–888
4. Boisset JC et al (2018) Mapping the physical network of cellular interactions. Nat Methods
5. Tanay A, Regev A (2017) Scaling single-cell genomics from phenomenology to mechanism. Nature 541(7637):331–338
6. Trapnell C (2015) Defining cell types and states with single-cell genomics. Genome Res 25(10):1491–1498
7. Cleary B et al (2017) Efficient generation of transcriptomic profiles by random composite measurements. Cell 171(6):1424–1436.e18
8. Klein AM et al (2015) Droplet barcoding for single-cell transcriptomics applied to embryonic stem cells. Cell 161(5):1187–1201
9. Macosko EZ et al (2015) Highly parallel genome-wide expression profiling of individual cells using nanoliter droplets. Cell 161 (5):1202–1214
10. Zheng GX et al (2017) Massively parallel digital transcriptional profiling of single cells. Nat Commun 8:14049
11. Habib N et al (2016) Div-Seq: single-nucleus RNA-Seq reveals dynamics of rare adult newborn neurons. Science 353(6302):925–928
12. Lake BB et al (2016) Neuronal subtypes and diversity revealed by single-nucleus RNA sequencing of the human brain. Science 352 (6293):1586–1590
13. Shekhar K et al (2016) Comprehensive classification of retinal bipolar neurons by single-cell transcriptomics. Cell 166(5):1308–1323.e30
14. Villani A-C et al (2017) Single-cell RNA-seq reveals new types of human blood dendritic cells, monocytes, and progenitors. Science 356(6335):eaah4573
15. Tasic B et al (2016) Adult mouse cortical cell taxonomy revealed by single cell transcriptomics. Nat Neurosci 19(2):335–346
16. Zeng H, Sanes JR (2017) Neuronal cell-type classification: challenges, opportunities and the path forward. Nat Rev Neurosci 18(9):530
17. Stegle O, Teichmann SA, Marioni JC (2015) Computational and analytical challenges in single-cell transcriptomics. Nat Rev Genet 16 (3):133
18. Arendt D (2008) The evolution of cell types in animals: emerging principles from molecular studies. Nat Rev Genet 9(11):868–882
19. Ecker JR et al (2017) The BRAIN initiative cell census consortium: lessons learned toward generating a comprehensive BRAIN cell atlas. Neuron 96(3):542–557
20. Kolodziejczyk AA et al (2015) The technology and biology of single-cell RNA sequencing. Mol Cell 58(4):610–620
21. Islam S et al (2014) Quantitative single-cell RNA-seq with unique molecular identifiers. Nat Methods 11(2):163
22. Menon V (2017) Clustering single cells: a review of approaches on high- and low-depth single-cell RNA-seq data. Brief Funct Genomics
23. Hicks SC, Teng M, Irizarry RA (2015, 025528) On the widespread and critical impact of systematic bias and batch effects in single-cell RNA-Seq data. bioRxiv
24. Butler A et al (2018) Integrating single-cell transcriptomic data across different conditions, technologies, and species. Nat Biotechnol 36 (5):411
25. Haghverdi L et al (2018) Batch effects in single-cell RNA-sequencing data are corrected by matching mutual nearest neighbors. Nat Biotechnol 36:421–427
26. Lopez R et al (2018) Bayesian inference for a generative model of transcriptome profiles from single-cell RNA sequencing. bioRxiv:292037
27. Lee JH et al (2014) Highly multiplexed subcellular RNA sequencing in situ. Science 343 (6177):1360–1363
28. Stahl PL et al (2016) Visualization and analysis of gene expression in tissue sections by spatial transcriptomics. Science 353(6294):78–82
29. Chen KH et al (2015) Spatially resolved, highly multiplexed RNA profiling in single cells. Science 348(6233):aaa6090
30. Lubeck E et al (2014) Single-cell in situ RNA profiling by sequential hybridization. Nat Methods 11(4):360
31. Fuzik J et al (2016) Integration of electrophysiological recordings with single-cell RNA-seq data identifies neuronal subtypes. Nat Biotechnol 34(2):175
32. Dixit A et al (2016) Perturb-Seq: dissecting molecular circuits with scalable single-cell

RNA profiling of pooled genetic screens. Cell 167(7):1853–1866.e17

33. Stoeckius M et al (2017) Simultaneous epitope and transcriptome measurement in single cells. Nat Methods 14(9):865
34. Frieda KL et al (2017) Synthetic recording and in situ readout of lineage information in single cells. Nature 541(7635):107–111
35. Raj B et al (2018) Simultaneous single-cell profiling of lineages and cell types in the vertebrate brain. Nat Biotechnol 36(5):442–450
36. Pertea M et al (2016) Transcript-level expression analysis of RNA-seq experiments with HISAT, StringTie and Ballgown. Nat Protoc 11(9):1650
37. Villani AC, Shekhar K (2017) Single-cell RNA sequencing of human T cells. Methods Mol Biol 1514:203–239
38. Satija R et al (2015) Spatial reconstruction of single-cell gene expression data. Nat Biotechnol 33(5):495–502
39. Lake BB et al (2018) Integrative single-cell analysis of transcriptional and epigenetic states in the human adult brain. Nat Biotechnol 36 (1):70–80
40. Pandey S et al (2018) Comprehensive identification and spatial mapping of Habenular neuronal types using single-cell RNA-Seq. Curr Biol 28(7):1052–1065.e7
41. Andrews TS, Hemberg M (2017) Identifying cell populations with scRNASeq. Mol Asp Med
42. Brennecke P et al (2013) Accounting for technical noise in single-cell RNA-seq experiments. Nat Methods 10(11):1093
43. Keogh E, Mueen A (2017) Curse of dimensionality. In: Encyclopedia of machine learning and data mining. Springer, pp 314–315
44. Hotelling H (1933) Analysis of a complex of statistical variables into principal components. J Educ Psychol 24(6):417
45. Hyvärinen A, Karhunen J, Oja E (2004) Independent component analysis, vol 46. Wiley, New York
46. Lee DD, Seung HS (2001) Algorithms for non-negative matrix factorization. In: Leen TK, Dietterich TG, Tresp V (eds) Advances in neural information processing systems, vol 13. MIT, Cambridge, UK
47. Haghverdi L et al (2016) Diffusion pseudotime robustly reconstructs lineage branching. Nat Methods 13(10):845
48. Lancichinetti A, Fortunato S (2009) Community detection algorithms: a comparative analysis. Phys Rev E Stat Nonlinear Soft Matter Phys 80(5 Pt 2):056117
49. Levine JH et al (2015) Data-driven phenotypic dissection of AML reveals progenitor-like cells that correlate with prognosis. Cell 162 (1):184–197
50. LVD M, Hinton G (2008) Visualizing data using t-SNE. J Mach Learn Res 9 (Nov):2579–2605
51. Soneson C, Robinson MD (2018) Bias, robustness and scalability in single-cell differential expression analysis. Nat Methods 15 (4):255
52. Breiman L (2001) Random forests. Mach Learn 45(1):5–32