nature protocols

PROTOCOL

https://doi.org/10.1038/s41596-021-00561-x

Check for updates

# Prioritization of cell types responsive to biological perturbations in single-cell data with Augur

Jordan W. Squair[1,2,3,6]✉, Michael A. Skinnider[1,2,4,6]✉, Matthieu Gautier[1], Leonard J. Foster[4,5] and Grégoire Courtine[1,2]✉

**Advances in single-cell genomics now enable large-scale comparisons of cell states across two or more experimental conditions. Numerous statistical tools are available to identify individual genes, proteins or chromatin regions that differ between conditions, but many experiments require inferences at the level of cell types, as opposed to individual analytes. We developed Augur to prioritize the cell types within a complex tissue that are most responsive to an experimental perturbation. In this protocol, we outline the application of Augur to single-cell RNA-seq data, proceeding from a genes-by-cells count matrix to a list of cell types ranked on the basis of their separability following a perturbation. We provide detailed instructions to enable investigators with limited experience in computational biology to perform cell-type prioritization within their own datasets and visualize the results. Moreover, we demonstrate the application of Augur in several more specialized workflows, including the use of RNA velocity for acute perturbations, experimental designs with multiple conditions, differential prioritization between two comparisons, and single-cell transcriptome imaging data. For a dataset containing on the order of 20,000 genes and 20 cell types, this protocol typically takes 1–4 h to complete.**

## Introduction

Since the first report of whole-transcriptome RNA-sequencing in a single cell more than a decade ago[1], the field of single-cell genomics has witnessed explosive growth[2]. Widely available workflows for single-cell and single-nucleus RNA-sequencing (scRNA-seq and snRNA-seq, respectively) now enable the measurement of gene expression in hundreds of thousands, or even millions, of individual cells[3]. This exponential increase in scale initially opened up the possibility of generating reference cellular atlases of healthy tissues and even entire organisms[4–8]. Consortia such as the Human Cell Atlas seek to realize this aim by mapping all of the cells within the human body[9]. More recently, continued increases in measurement scale have culminated in the generation of atlases spanning multiple experimental conditions, which catalog the impact of biological perturbations at single-cell resolution through side-by-side comparison with an unperturbed control. This approach has enabled the dissection of cell-type-specific transcriptional programs activated in diseases such as asthma[10], Alzheimer's disease[11,12] and ulcerative colitis[13], or in response to stimulation with inflammatory cytokines[14], gene mutation[15], and aging[16], among many other perturbations[17].

Due to the high dimensionality of scRNA-seq data, investigators rely on specialized bioinformatic methods to facilitate the biological interpretation of these complex datasets. In this respect, the transition to atlases spanning multiple experimental conditions has brought new challenges. The changes in the transcriptional program of a given cell type in response to a disease or experimental manipulations are generally subtle compared with the marked differences between cell types within a healthy tissue, which can be identified using unsupervised clustering. Computational approaches to experiments with multiple conditions have focused primarily on identifying individual genes, proteins or regions of open chromatin whose abundance displays a statistically significant difference relative to

[1]Center for Neuroprosthetics and Brain Mind Institute, Faculty of Life Sciences, École Polytechnique Fédérale de Lausanne (EPFL), Lausanne, Switzerland. [2]NeuroRestore, Department of Clinical Neuroscience, Lausanne University Hospital (CHUV) and University of Lausanne (UNIL), Lausanne, Switzerland. [3]International Collaboration on Repair Discoveries (ICORD), University of British Columbia, Vancouver, British Columbia, Canada. [4]Michael Smith Laboratories, University of British Columbia, Vancouver, British Columbia, Canada. [5]Department of Biochemistry and Molecular Biology, University of British Columbia, Vancouver, British Columbia, Canada. [6]These authors contributed equally: Jordan W. Squair, Michael A. Skinnider. ✉e-mail: jordan.squair@epfl.ch; michael.skinnider@msl.ubc.ca; gregoire.courtine@epfl.ch

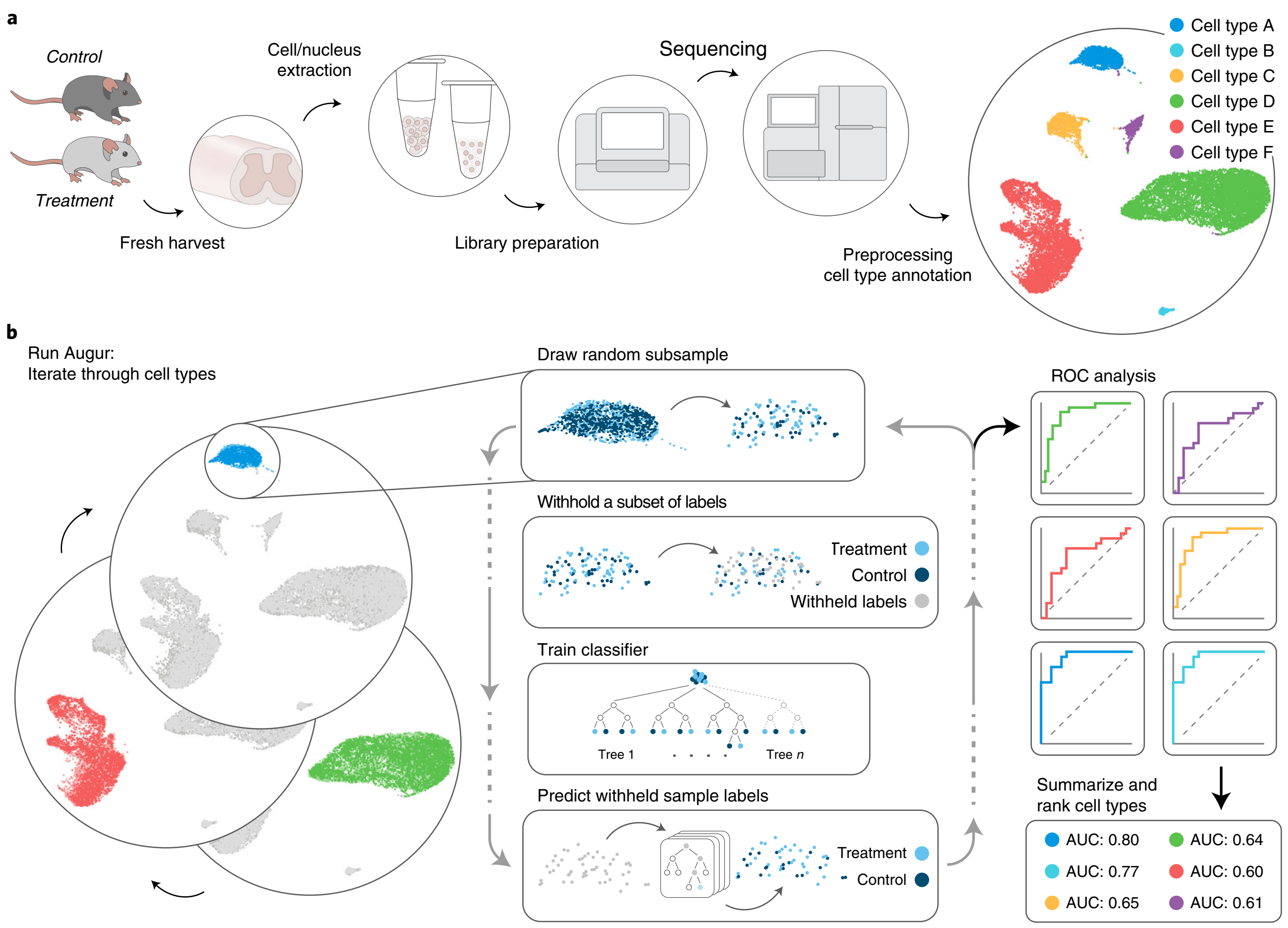


**Fig. 1 | Overview of the workflow for cell-type prioritization in single-cell data. a**, Schematic overview of the experimental basis to identify the cell types responsive to biological perturbations with Augur. **b**, Schematic overview of the machine-learning framework for cell-type prioritization implemented by Augur.

an unperturbed control[18–20]. While these statistical tools allow reliable inferences at the level of individual analytes, many biological questions involving multiple experimental conditions require inferences at the level of cell types. This level of analysis is essential to identify the cellular states or subpopulations that mediate the biological response to the perturbation of interest. For example, single-cell transcriptomics has been applied to identify the specific populations of neurons that are activated in response to behavioral stimuli[21–25]. Inferences at the level of cell types are also essential to determine the cellular loci of dysfunction in disease[11,26] or to demonstrate the involvement of specific cell types in the response to therapeutic interventions[27,28].

## Development of the protocol

The absence of bespoke computational methods to perform inferences at the level of cell types compelled us to develop Augur[29]. Augur employs a machine-learning framework to rank the cell types found within a single-cell dataset according to the relative magnitude of their response to a biological perturbation (Fig. 1). We refer to this paradigm as cell-type prioritization. In developing Augur, we reasoned that the cell types that respond more strongly to a perturbation should become more separable, within the multidimensional space of molecular measurements, than the populations of cells that are less responsive to this perturbation. To quantify this separability, we formulated a classification task. Concretely, a machine-learning classifier is trained to predict the experimental condition associated with each cell, such as treatment or control. Separate classifiers are trained for each cell type, before evaluating the accuracy of each classifier in cross-validation. The accuracy with which experimental conditions can be predicted from molecular measurements alone quantifies the relative magnitude of the response from each cell type. This quantification thus allows Augur to prioritize the cell types most responsive to the biological perturbation of interest.

In this protocol, we provide a detailed, annotated workflow for cell-type prioritization in single-cell data using Augur. The protocol includes code blocks in the R programming language that illustrate the steps necessary to proceed from a read count matrix to a list of cell types, ranked on the basis of the relative magnitude of their response to a biological perturbation. The protocol incorporates detailed discussions of data preprocessing, parameter settings and visualization of the results. We also discuss potential pitfalls and common modifications of the protocol, including the use of continuous or multiclass sample labels, the use of RNA velocity[30] in cell-type prioritization, a test for differential prioritization, and cell-type prioritization in datasets confounded by batch effects. We present case studies describing the application of Augur to five different single-cell datasets. The protocol is written at the level of a user with some basic familiarity with the console, such as the ability to enter commands into the R command prompt, but no specific programming expertise or background in computational biology.

## Applications of the method

We originally developed Augur to identify the neural circuits involved in the recovery of walking after paralysis when delivering epidural electrical stimulation (EES) to the lumbar spinal cord[29,31]. To uncover these neural circuits, we performed snRNA-seq of the lumbar spinal cord in mice with spinal cord injury after walking for 30 min with or without EES. Unsupervised clustering identified a total of 38 neuronal subtypes. We then applied Augur to identify the neuronal subtypes that responded more prominently to the production of walking with EES. Augur prioritized V2a and V1/V2b interneurons. These cells receive synapses from large-diameter proprioceptive afferents, which are thought to be directly depolarized by the electrical field generated by EES[32]. We validated this prediction using RNA in situ hybridization, identifying an increase in the expression of the immediate early gene (IEG) Fos among prioritized neuronal subtypes. We also conducted virus-mediated anatomical tracing to demonstrate that these interneurons establish dense synaptic projections onto motor neurons.

We subsequently applied Augur to data curated from >20 previously published studies, thereby demonstrating the versatility of Augur to prioritize cell types in many single-cell datasets that encompass at least two experimental conditions. For instance, we validated Augur by showing that the resulting cell-type prioritizations recapitulate the known dose–response curve in mononuclear phagocytes stimulated with lipopolysaccharide for 2, 4 or 6 h[33]. We also applied Augur to two independent datasets that studied the response of neurons in the visual cortex to light stimulation[22,34]. Although these two datasets were produced using entirely orthogonal technologies, Augur ranked excitatory neurons from specific cortical layers in identical order for both datasets. Indeed, since Augur does not consider the nature or distribution of the molecules quantified within each cell, it can be applied to a variety of single-cell technologies. For instance, in our original description of Augur, we demonstrated its application to single-cell transcriptome imaging techniques such as STARmap[34] or MERFISH[35] and single-cell epigenomics technologies such as single-cell ATAC-seq[36]. Augur can also be applied to gene-specific estimates of RNA velocity[30] to dissect the response to acute perturbations on the timescale of transcription (e.g., minutes to hours; Box 1). Thus, Augur is a versatile method that can be applied to understand the impact of any biological perturbation using multidimensional measurements in single cells.

Beyond the binary classification framework that characterizes conventional treatment versus control experimental designs, we have equipped Augur with the capacity to perform cell-type prioritization by multiclass classification (for experiments with three or more experimental conditions) or regression (for experiments with continuous sample labels). In the former case, the macro-averaged area under the receiver operating characteristic curve (AUC) over all pairwise comparisons is reported to the user, whereas in the latter case, the concordance correlation coefficient (CCC)[37] is reported. Additional guidance on the differences between the use of binary, multiclass and continuous labels in Augur is provided in Box 2.

To adapt Augur to more complex experimental designs, we devised a test for 'differential prioritization' to identify statistically significant differences in the cellular response to perturbation between two backgrounds (e.g., genotypes), or even between two entirely distinct perturbations. Differential prioritization is achieved through a permutation test of the difference in AUC ($\Delta$AUC) between two sets of cell-type prioritizations, compared with the expected $\Delta$AUC between the same two prioritizations after random permutation of sample labels. Further discussion of the application of differential prioritization to identify differentially responsible cell types, and details of the new, optimized workflow for differential prioritization presented here, is provided in Box 3.

**Box 1 | RNA velocity**

Applying Augur to prioritize cell types in single-cell transcriptomics relies on an implicit assumption that the transcriptional state of a cell is informative about its response to a perturbation. However, because there is a lag in the transcription of new mRNA molecules in response to a stimulus, in the case of a very acute perturbation, in which transcription is measured on the timescale of minutes to hours following the perturbation, the total transcriptional output of the cell may not yet fully reflect the impact of the perturbation. In such cases, rather than applying Augur to the total gene expression observed in each cell, it may be more informative to consider estimates of the RNA velocity[30]. This method uses the ratio of spliced to unspliced mRNAs observed to estimate the time derivative of gene expression. This high-dimensional vector is, in turn, predictive of future cell state, on a timescale of a few hours.

Since Augur makes no assumptions about the nature or distribution of input features, the algorithm can readily be applied to a matrix of RNA velocity, instead of total gene expression. We found that in acute perturbations (ranging from 45 min to 4 h in duration), such as 1 h of light exposure in the mouse visual cortex, the RNA velocity captured significantly more information about the perturbation response than total gene expression. We validated this finding with experimental measurements of transcriptional activity obtained by metabolic labeling[78]. Conversely, such an information gain was not observed in chronic perturbations. Of note, typical RNA velocity analyses perform an implicit feature selection step by filtering lowly expressed genes and genes for which velocity could not be reliably estimated. Consequently, we provide an argument to Augur to specify that an RNA velocity matrix is provided as input (`augur_mode = "velocity"`), which in turn disables the feature selection procedure used by default within Augur.

**Box 2 | Binary, multiclass and continuous labels**

The intuition underlying Augur is that cell types that undergo a more profound response to a stimulus should become more separable in the multidimensional space of molecular measurements than unaffected cell types. For instance, in single-cell transcriptomics, cells from stimulated versus control samples should become easier to distinguish within cell types that are transcriptionally activated in response to a stimulus. Augur formalizes this notion by asking how accurately the sample label can be predicted from molecular measurements. At the most basic level, this is achieved by training machine-learning classifiers to predict whether a cell was obtained from the treatment condition or the control condition. Because there are only two conditions in this framework, this procedure is referred to as binary classification.

The accuracy of cell-type prioritization is quantified using a metric known as the area under the receiver operating characteristic curve, or AUC. The AUC can be thought of as the probability that any given cell from the stimulated condition is ranked higher by the classifier than any given cell from the unstimulated condition. An AUC of 0.5 corresponds to random chance (that is, there is only a 50% chance that stimulated cells are ranked higher than unstimulated cells), whereas an AUC of 1.0 corresponds to perfect accuracy (that is, every stimulated cell is ranked higher than every unstimulated cell).

When single-cell datasets include more than two conditions, a binary classification framework can be employed to compare individual pairs of conditions that are of interest. For instance, the user could first compare group A with group B, and then separately compare group B with group C. However, in some cases, it may be more relevant to ask how well all of the experimental groups can be distinguished from one another (e.g., how well can the classifier predict that a cell belongs to group A versus groups B or C). This procedure is referred to as multiclass classification. In multiclass classification, Augur first calculates the AUC for distinguishing group A from all other groups, then group B from all other groups, and so on. These 'one-vs.-rest' AUCs are then averaged to obtain the macro-averaged AUC, which is reported to the user.

In other cases, experimental labels may not be categorical in nature at all. For example, investigators might measure a continuous phenotype in mice, such as body weight, and wish to identify cell types whose transcriptional output varies continuously with the phenotype of interest. In these cases, Augur performs cell-type prioritization by regression, instead of classification, by training random forest models to predict the numerical value associated with the sample from which the cell was obtained. The CCC is reported to the user, instead of the AUC, as a basis for cell-type prioritization. This metric captures both the precision and consistency of the predictions, unlike metrics such as the correlation (which captures only consistency) or the root mean-squared error (which captures only accuracy). A CCC of 0 reflects no ability to predict the phenotype from molecular measurements, whereas a CCC of 1 reflects perfect predictions.

In some cases, single-cell datasets can be analyzed using more than one of these three frameworks (binary, multiclass or continuous). Users are advised to carefully consider the nature of their experimental conditions and, if possible, to compare results from different frameworks. To this end, case study 3 in this protocol demonstrates how cell-type prioritizations from binary and multiclass classification approaches to the same dataset can be compared and visualized.

## Comparison with other methods

The primary approach that has been applied to identify cell types responsive to a given biological perturbation is based on the number of differentially expressed (DE) genes within each cell type. In this approach, investigators conduct a differential expression analysis using their preferred statistical test, comparing perturbed and unperturbed cells of a given type, and then tally the number of genes that pass a threshold for statistical significance. This number provides a quantitative basis for cell-type prioritization: cell types with more DE genes are inferred to be more responsive, whereas cell types with fewer DE genes are inferred to be less responsive.

Although attractive for their conceptual simplicity, approaches based on the number of DE genes present a number of limitations. The most critical issue affecting these methods is their bias toward cell types that are found with a higher relative abundance in the dataset. This can be rationalized on the basis that larger numbers of cells provide greater statistical power for identifying significant univariate differences. In both simulations and ground-truth datasets, this bias can lead DE-based methods to produce biologically incoherent prioritizations. A further limitation is that in sparsely sequenced, droplet-based scRNA-seq datasets, the vast majority of DE genes are detected from among

## Box 3 | Differential prioritization

In single-cell datasets with more complex experimental designs, the user may not only be interested in which cell types are responsive to a given perturbation, but also whether certain cell types respond more or less strongly to one perturbation than the other. For instance, an investigator might wish to compare the effects of drug A and drug B on a complex tissue with a common untreated control group and identify cell types that respond more strongly to drug A.

To compare datasets with two different prioritizations, we designed a permutation-based test for differential prioritization. In this procedure, the user first performs cell-type prioritization on drug A and drug B separately, then calculates the ΔAUC between drug A and drug B. To compute the statistical significance of the ΔAUC, an empirical null distribution of ΔAUCs is then calculated for each cell type by permuting the sample labels, then repeating cell-type prioritization in the permuted data. Permutation *P*-values are then calculated as previously described[79]. This procedure thus enables the identification of statistically significant differences in cell-type prioritization between conditions, as well as the condition in which the cell type is more transcriptionally separable.

Our original approach to differential prioritization[29] involved permuting sample labels across the entire dataset 1,000 times, and then applying Augur to each permuted dataset in turn. Although we showed that this framework allowed us to identify neuronal subpopulations preferentially activated by a stimulus in different backgrounds (e.g., in mice homozygous versus heterozygous at a particular locus), this workflow was very slow, requiring thousands of core-hours; consequently, differential prioritization was generally accessible only to users with access to high-performance computing resources. We thus set out to develop an optimized workflow that would enable users to perform differential prioritization with more limited computational resources—for instance, one that would run overnight on a laptop computer.

Our new, more efficient workflow for differential prioritization incorporates two optimizations. First, we reasoned that because the feature selection procedure applied to each cell type is deterministic, performing feature selection just once (instead of separately for each permutation) would reduce computational demands. Second, we asked whether we could achieve results that approximated the complete workflow using either fewer permutations, or fewer subsamples within each permuted dataset. Results with 100 or more permutations closely approximated those from the full dataset, but differential prioritization began to degrade with <100 permutations, which still required >130 core-hours to complete (Extended Data Fig. 1a–e). The correlation to the full dataset degraded much more rapidly with <50 subsamples per permutation (Extended Data Fig. 1f–h), indicating a full complement of subsamples is necessary to approximate the null distribution (Extended Data Fig. 1i). However, we reasoned that individual subsamples from the same theoretical null distribution could be reused, so long as they were sampled with replacement from a sufficiently large background: in other words, drawing means of 50 subsamples randomly from a background of 5,000 subsamples should approximate the true means, computed from a much larger total of 500,000 independent subsamples, reasonably well. Indeed, we found that summarizing a pool of 500 or more randomly permuted subsamples into a total of 1,000 means of 50 subsamples each matched the null distribution from the complete set of 1,000 subsamples (Extended Data Fig. 1j) and yielded results that were almost perfectly correlated to the full dataset (Extended Data Fig. 1k–m), at ~100-fold lower computational cost (Extended Data Fig. 1n).

We implemented this optimized workflow for differential prioritization in Augur, which can be called using the argument `augur_mode = "permutation"`. By default, this mode will perform a total of 500 rounds of threefold cross-validation in small subsamples of cells, then draw 50 subsamples at a time to obtain a null distribution of 1,000 mean AUCs. The latest release of Augur also includes new functions to calculate permutation *P*-values and visualize the results of a differential prioritization analysis (Steps 28–30 in this protocol).

the top 20% of most highly expressed genes, with essentially no usable signal coming from the remaining 80% of the transcriptome. Finally, many bespoke statistical methods for identifying DE within single-cell transcriptomics data require significant computational resources (that is, CPU time, memory or both) to analyze the large datasets.

In contrast to DE-based methods, Augur employs a repeated subsampling procedure that eliminates bias toward more abundant cell types. Moreover, Augur incorporates information from the entire transcriptome, and can even perform cell-type prioritization using only the 20% most lowly expressed genes, albeit with some loss of accuracy. Finally, because all operations in Augur are either optimized for sparse matrices or take place on dense matrices for only a small subsample of cells, Augur is a computationally efficient method that can scale up to millions of cells. When using four cores, Augur typically requires <1 h and 4 GB of random-access memory (RAM) per core to perform cell-type prioritization, although runtime and memory requirements will vary with the size of the dataset and the CPU used to perform the computations.

Some of the limitations described above have been addressed by tallying the number of DE genes within random samples of cells of a fixed size, through a subsampling procedure analogous to that employed by Augur[26,38]. This procedure addresses the relationship between the number of DE genes and the number of cells per type, although not the tendency of DE-based methods to make use only of information from the most highly expressed genes. Moreover, in the case of a subtle biological perturbation, no DE genes may be detected within any small sample of cells. This is especially true for sparsely sequenced datasets, which provide less information to call DE genes. Finally, on a conceptual level, these approaches aggregate many independent univariate comparisons rather than calculating a single measure in an inherently multidimensional space, which may reduce their accuracy. In simulated data, we found that Augur produces significantly more accurate cell-type prioritizations than those achieved by counting the number of DE genes when using an identical subsampling procedure, particularly in sparsely sequenced datasets or in subtler perturbations (Extended Data Fig. 2).

In some settings, domain-specific knowledge has been applied to formulate quantitative criteria for cell-type prioritization. A prominent example is the proportion of neurons expressing the gene Fos, an IEG that reveals neuronal activation, or other IEGs[23,24,39,40]. Others have combined summary statistics from univariate differential expression analyses with a secondary source of information, such as gene-level associations to a phenotype of interest from genome-wide association studies[21]. In general, criteria of this nature rely on strong assumptions and prior knowledge about the biological system of interest. Prioritizations based on IEG expression may also be confounded by the generally low expression of these genes, leading to noisy quantifications that in turn present a barrier to accurate prioritization.

Beyond its specific advantages with respect to DE-based methods, Augur has a number of additional strengths. First, the use of a random forest classifier in Augur eliminates any assumptions about the distributions of the input features. Consequently, Augur is robust both to the nature of the molecular measurements and the specific preprocessing and normalization steps used to arrive at the input genes-by-cells matrix. Second, unlike distance-based methods[41], the measure of separability calculated in Augur accounts for the specific biological variability and technical noise across all genes within each cellular subpopulation. Third, as a decision tree method, the random forest automatically identifies features relevant to the perturbation, making Augur robust to the inclusion of noisy input features. (Note, however, that a feature selection step is included within Augur to decrease runtime.) Finally, the AUC provides a single, intuitive measure of cell-type prioritization, with an AUC of 0.5 implying cells cannot be assigned to a specific experimental condition on the basis of their molecular profiles with better-than-random accuracy, and an AUC of 1 reflecting perfect classification.

## Limitations

### Augur relies on subjective definitions of cell types

Unlike genes, cell types are not universally understood entities. They are typically assigned in single-cell genomics data based on a laborious process that may involve multiple iterative rounds of unsupervised clustering, followed by subclustering or cluster merging. Clusters of cells with similar transcriptional profiles are ultimately assigned human-readable labels through manual inspection. As a result, annotation of the same dataset by two different investigators may produce different cell-type annotations—sometimes markedly so. By performing an inference at the level of cell types, Augur is inherently sensitive to the subjectivity that accompanies cell-type annotation, and more generally, to the definition of ‘cell types’ themselves[42,43]. For example, although the operator-dependent nature of cell-type annotation can occasionally manifest in outright errors (such as mislabeling of cellular subpopulations or use of inappropriate clustering parameters), it is also generally the case that no single set of clusters provides a definitive description of a biological system. In the central nervous system, for example, the cell types ‘neuron’ and ‘inhibitory interneuron’ might represent two equally correct annotations for a particular cell, differing only in their biological resolution. In other cases, where a population of cells falls along a smooth continuum or trajectory of gene expression, a set of discrete and nonoverlapping clusters may provide a poor description of the biological system. Issues of this nature can be partially addressed by constructing a clustering tree[44] that describes the relationships between clusters at many different biological resolutions and applying Augur to all possible clustering solutions, in order to clarify which resolution may be most relevant to the perturbation of interest. We discuss existing approaches to cell-type annotation in detail in this protocol, with the aim of orienting the reader to potential pitfalls.

### Augur aggregates continuous underlying gradients of response intensity

A cell type that mediates the tissue- or organism-level response to a given perturbation may itself comprise subpopulations of ‘responder’ and ‘non-responder’ cells. Alternatively, cells of a given type might fall along a continuous spectrum of perturbation response intensity, with some cells exhibiting no transcriptional response, some exhibiting a profound response and the majority of cells lying somewhere in between. A necessary limitation of inferences at the level of cell types, rather than at the level of individual cells, is that continuous gradients of response intensity will be aggregated into a metric that characterizes the average difficulty of separating perturbed and unperturbed cells.

### Augur does not provide gene-level results

A related limitation is that, because Augur performs an inference at the level of cell types, it fails to resolve which individual genes are specifically involved in the perturbation response. We have found that the feature importance from the random forest classifier does provide some indication of the genes used to discriminate perturbed and unperturbed cells, and consequently, this information is provided to the user as an output for the sake of convenience, with the option to specify one of two different feature importance methods (mean decrease in accuracy or mean decrease in the Gini coefficient). However, when the inference of interest is at the level of individual genes, we believe the application of a statistical test for differential expression is, both conceptually and pragmatically, the most appropriate approach.

### Augur does not detect changes in cellular composition

We have designed Augur to identify cell types that are most separable from unperturbed cells following a perturbation. However, some perturbations may manifest primarily in a change in the relative abundance of a particular cellular subpopulation. By design, the subsampling procedure implemented in Augur will discard information about the relative abundances of each cell type. Because Augur samples equal numbers of cells from each condition prior to cross-validation, this subsampling procedure is also not confounded by the proportion of cells obtained from each experimental condition. Moreover, a change in the abundance of a particular cell type may or may not be accompanied by changes in the intrinsic transcriptional profile of that cell type. For this reason, like others who have considered the problem of 'differential state' analysis[19], we suggest that investigators perform a simple statistical test for differential abundance of each cell type following perturbation to help contextualize the results of an Augur analysis.

In the most extreme case, a biological perturbation may lead to a scenario where a cell type is present almost exclusively in one condition (i.e., the appearance of a new cell type in perturbed tissues, or the disappearance of a cell type observed in unperturbed tissues). Because scenarios of this nature do not provide a basis for any inferences about the transcriptional programs activated within that cell type in response to perturbation, analysts will be limited to commenting on the observed change in the relative abundance of that cell type between conditions.

### Augur runtime scales with the number of cell types

As discussed above, Augur is a computationally efficient method that can analyze hundreds of thousands of cells using only the computational resources of a laptop computer, and generally runs within 1 h. A minor limitation is that the runtime of Augur scales approximately linearly with the number of cell types analyzed. Thus, in a setting with many dozens or even hundreds of cell types (for instance, the entire human body[6]), Augur may require more time to complete its analysis.

## Overview of the procedure

The basic workflow for cell-type prioritization with Augur proceeds as follows. First, preprocessing of raw read counts is performed to produce a gene expression matrix in which each cell is annotated with both its experimental condition of origin and an inferred cell type (Steps 1–7). Second, this annotated matrix is provided to Augur as input, along with a series of parameters that may be specified by the user, and cell-type prioritization is performed (Steps 8 and 9). Finally, the output is interpreted and visualized (Steps 10–12). An overview of the complete workflow is shown in Fig. 2.

### Preprocessing and cell-type annotation (Steps 1–7)

To perform cell-type prioritization using Augur, three pieces of information are required as inputs. First, Augur requires a matrix or data frame containing the single-cell molecular measurements that form the basis for cell-type prioritizations. In the context of single-cell transcriptomics, this generally corresponds to a matrix in which each row represents a gene, each column represents a cell, and entries represent the expression of each gene in each cell. In other single-cell modalities, rows may instead represent transcripts, proteins, open chromatin regions or other molecular features. Second, Augur requires the ground-truth sample label associated with each cell: for example, whether the cell was obtained from a stimulated or control animal. Third, Augur requires a cell-type annotation for each cell in the dataset. These annotations correspond to inferred cell types at the biological resolution of interest to the investigator(s). Augur expects that the label and cell type for each cell are included in a data frame that may contain other cell-level metadata for each cell in the input gene expression matrix. Alternatively, the expression matrix and associated metadata may be provided jointly as input, in the form of a Seurat[45], monocle[3] or SingleCellExperiment[46] object.

In this protocol, we walk through one possible workflow for preprocessing a raw read count matrix to annotate cell types for input to Augur. This workflow includes common analysis steps including normalization, dimensionality reduction, data integration, clustering, marker gene identification and visualization[47]. Because the assignment of cell types has a critical impact on the output of Augur, we

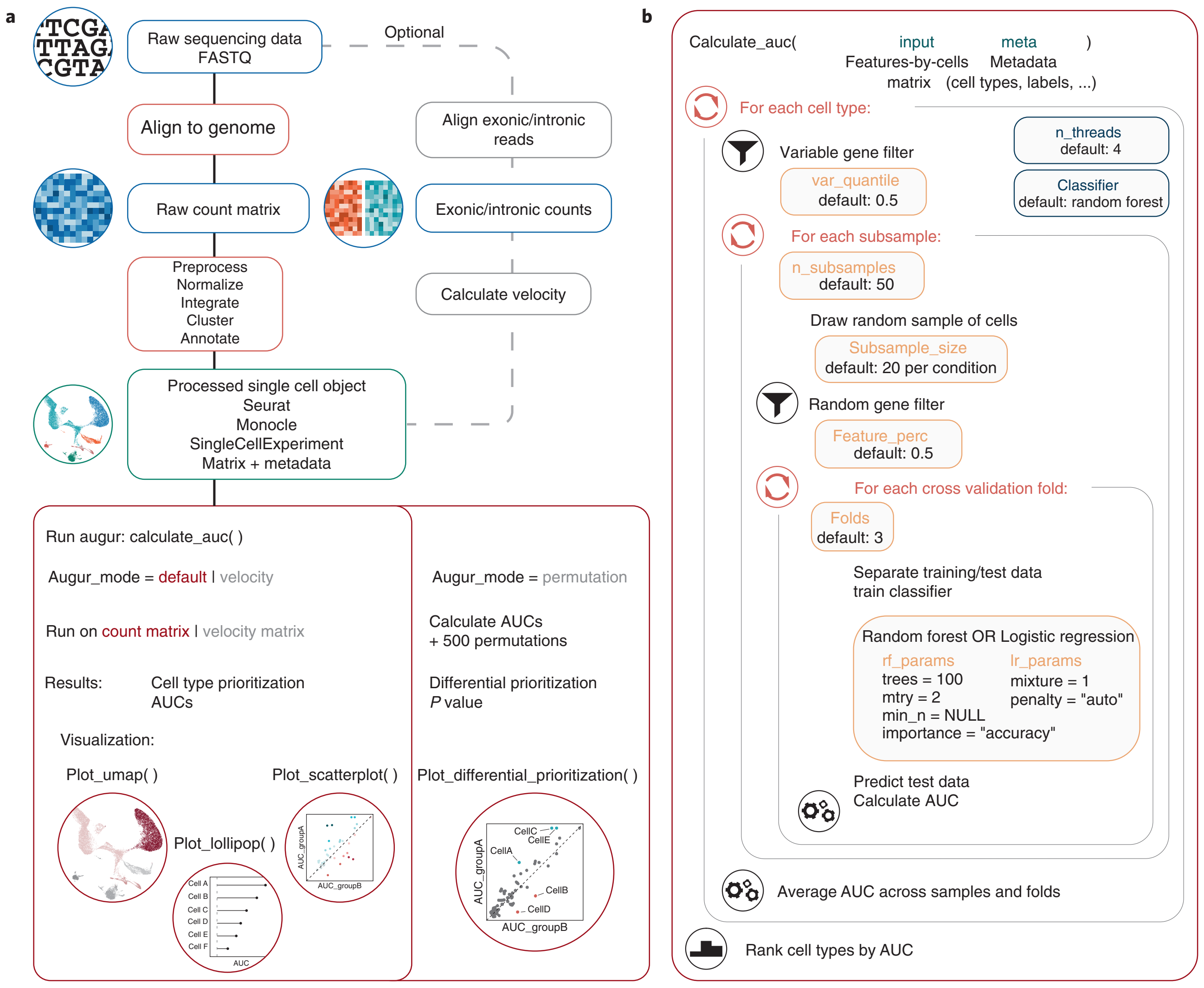

a
Raw sequencing data FASTQ
Optional
Align to genome
Align exonic/intronic reads
Raw count matrix
Exonic/intronic counts
Preprocess
Normalize
Integrate
Cluster
Annotate
Calculate velocity
Processed single cell object
Seurat
Monocle
SingleCellExperiment
Matrix + metadata
Run augur: calculate_auc( )
Augur_mode = default | velocity
Augur_mode = permutation
Run on count matrix | velocity matrix
Calculate AUCs + 500 permutations
Results: Cell type prioritization AUCs
Differential prioritization P value
Visualization:
Plot_umap( )
Plot_scatterplot( )
Plot_lollipop( )
Plot_differential_prioritization( )
b
Calculate_auc( input meta )
Features-by-cells matrix
Metadata (cell types, labels, ...)
For each cell type:
n_threads default: 4
Classifier default: random forest
Variable gene filter
var_quantile default: 0.5
For each subsample:
n_subsamples default: 50
Draw random sample of cells
Subsample_size default: 20 per condition
Random gene filter
Feature_perc default: 0.5
For each cross validation fold:
Folds default: 3
Separate training/test data
train classifier
Random forest OR Logistic regression
rf_params
trees = 100
mtry = 2
min_n = NULL
importance = "accuracy"
lr_params
mixture = 1
penalty = "auto"
Predict test data
Calculate AUC
Average AUC across samples and folds
Rank cell types by AUC

**Fig. 2 | Augur workflow. a**, Overview of the complete workflow to run Augur, from raw sequencing reads to cell-type prioritization, including visualization of results and key algorithmic parameters. **b**, Overview of operations performed internally within the Augur function `calculate_auc` and the remaining Augur parameters.

discuss each of these steps in detail. We also describe how investigators may vary the biological resolution of their cell-type annotation to perform cell-type prioritization at multiple resolutions, if desired. Our workflow employs Seurat[45] to preprocess scRNA-seq data, but other packages are available that implement much of the same functionality, notably including SCANPY[48] in the Python programming language.

Alternatively, if a suitable reference atlas is available, cell types can be automatically assigned using supervised annotation frameworks[49,50], with minor adaptations to the workflow. Several approaches to cell-type classification have recently been benchmarked[51]. It is important to consider that the accuracy of these annotations will depend on both (i) the accuracy of the original cell-type annotations in the reference atlas and (ii) the accuracy of the classification procedure used to automatically assign cell types in the user's dataset.

We assume that the user has produced a read count matrix as input prior to beginning this protocol. A number of tools are available to accomplish this, such as CellRanger[52], dropEst[53], kallisto | bustools[54], alevin[55] or STARsolo[56]; Augur is agnostic to the particular methodology used to generate the input matrix. Similarly, to analyze RNA velocity data, we assume the user has produced separate matrices of intronic and exonic read counts, which can be generated using tools such as velocyto[30] or dropEst[53]. For other single-cell modalities, analogous preprocessing steps should be carried out as appropriate to arrive at an input matrix. Approaches to preprocessing single-cell ATAC-seq data, for example, have recently been comprehensively benchmarked[57]. We additionally expect that the user has performed some basic quality control of the input matrix. Commonly, this would involve removing low-quality cells, which is often accomplished by filtering cells on the basis of thresholds on the total number of reads or unique molecular identifiers (UMIs), as well as the proportion of reads mapping to mitochondrial genes; more sophisticated approaches have also been proposed[58,59]. Users may also wish to apply a computational tool for detection of so-called 'doublets', in which more than one cell has received the same barcode[60,61]. Low-quality cells and doublets may also be discarded at the cell-type annotation stage (Step 6), if manual inspection of certain clusters reveals that they are characterized by low read counts or implausible combinations of marker genes.

**Cell-type prioritization (Steps 8 and 9)**

After preprocessing the input dataset and assigning cell types, cell-type prioritization is performed. A number of parameters can be optionally specified by the user to control the precise manner in which this procedure is carried out within Augur (Table 1). Although we have found that Augur is generally highly robust to these parameters, with a few notable exceptions, we here provide a brief overview of the operations executed within Augur to clarify how these parameters impact cell-type prioritization (Fig. 2).

Augur first checks that the input is valid and contains all of the necessary metadata, then automatically detects the type of sample labels provided (binary, multiclass or continuous; Box 2), which in turn determines how cell-type prioritization is performed (that is, through binary classification, multiclass classification, or regression). Next, cell types without a certain minimum number of cells are removed (parameter 'min_cells'). Then, if the dataset contains >1,000 features, feature selection is performed for each cell type. Specifically, a local polynomial regression is fit between the mean and the coefficient of variation, and a subset of highly variable genes (HVGs) are retained on the basis of their residuals in this model (parameter 'var_quantile'). Performing HVG selection separately for each cell type allows Augur to specifically select genes relevant to the perturbation response within cell types, and prevents differences in gene expression between cell types from confounding HVG selection. Then, from this cell-type-specific matrix, Augur repeatedly draws small subsamples of cells of fixed size (parameters 'n_subsamples' and 'subsample_size'). To further improve runtime, only a randomly selected subset of HVGs is retained in each subsample (parameter 'feature_perc'). Each subsample is in turn divided into $k$ folds (parameter 'folds'). A classifier is trained to predict the sample labels from data in the first $k - 1$ folds. By default, Augur employs a random forest classifier, but also implements logistic regression (parameter 'classifier'), which each have specific settings controlled by the parameters 'rf_params' and 'lr_params', respectively. The trained classifier is then applied to predict sample labels in the held-out fold. The predicted labels are compared with the experimental ground truth, and a number of different evaluation metrics are computed. Finally, each metric is averaged first over folds, then over subsamples, for each cell type. Because the feature selection and cross-validation steps are independent for each cell type, this entire process is parallelized over cell types (parameter 'n_threads').

**Interpret and visualize output (Steps 10–12)**

The primary output from Augur consists of a ranked list of cell types, along with a measure of their 'separability' derived from the cell-type prioritization procedure described above (the AUC, macro-averaged AUC, or CCC, depending on the label type, averaged over folds and subsamples). Augur additionally returns an expanded suite of metrics that includes values calculated in each individual fold of cross-validation, the feature importance of each feature in each fold, and the input and parameters that were supplied to obtain the results.

We discuss several ways to visualize these results, including how the results of a cell-type prioritization analysis may be overlaid onto a low-dimensional representation of the input dataset to provide a global portrait of the perturbation response. We also discuss how alternative visualization methods may be used to compare Augur results obtained using different parameter settings, and how to visualize the results of a differential prioritization analysis. Finally, we discuss the use of clustering trees[44] to refine the cell-type annotations themselves that form the basis for prioritization by Augur.

## Experimental design

Augur can be applied to perform cell-type prioritization in essentially any single-cell dataset with two or more experimental conditions. In this protocol, we illustrate the use of Augur through case studies of five datasets (Table 2). The experimental design of these datasets is as follows.

**Table 1 | Overview of Augur parameters**

| Parameter | Description | Default value |
|---|---|---|
| `input` | Input object: either a Seurat, monocle or SingleCellExperiment object; or a matrix, sparse matrix or data frame with genes (features) in rows and cells in columns | N/A (required input) |
| `meta` | If the input object is a matrix, the metadata associated with the input matrix | `NULL` |
| `label_col` | Column within the metadata that contains experimental conditions (e.g., group, disease or timepoint) | `"label"` |
| `cell_type_col` | Column within the metadata that contains cell-type labels | `"cell_type"` |
| `n_subsamples` | Number of random subsamples of fixed size to draw from the complete dataset for each cell type | `50` |
| `subsample_size` | Number of cells per type to subsample randomly from each experimental condition | `20` |
| `folds` | The number of folds of cross-validation to perform | `3` |
| `min_cells` | The minimum number of cells for a particular cell type in each condition to retain that type for analysis (note: if `NULL`, defaults to `subsample_size`) | `NULL` |
| `var_quantile` | Quantile of HVGs to retain for each cell type, using the variable gene filter; for example, set `var_quantile = 0.9` to retain only the top 10% most variable genes | `0.5` |
| `feature_perc` | Proportion of genes that are randomly selected as features for input to the classifier in each subsample, using the random gene filter | `0.5` |
| `n_threads` | Number of cores to use for parallelization | `4` |
| `show_progress` | Display of a progress bar with estimated time remaining | `TRUE` |
| `classifier` | The classifier to use in calculating AUC; choices are `"rf"` (random forest) or `"lr"` (logistic regression) | `"rf"` |
| `rf_params` | List of parameters for the random forest classifier<br>● `trees`: number of trees in the random forest<br>● `mtry`: number of features randomly sampled at each split<br>● `min_n`: minimum number of observations to split a node in the decision tree<br>● `importance`: method by which feature importance is calculated; choices are `"accuracy"` (mean decrease in accuracy) or `"gini"` (mean decrease in the Gini coefficient) | `list(trees = 100, mtry = 2, min_n = NULL, importance = "accuracy")` |
| `lr_params` | List of parameters for the logistic regression classifier<br>● `mixture`: the proportion of L1 versus L2 regularization to use in the model<br>● `penalty`: the total amount of regularization to use | `list(mixture = 1, penalty = "auto")` |

1 Kang et al.[14]: scRNA-seq data collected using the 10x Genomics platform[52] from peripheral blood mononuclear cells (PBMCs) from eight lupus patients, before and after stimulation with recombinant IFN-β. This dataset is used to demonstrate the basic Augur workflow, beginning from a read count matrix.

2 Skinnider et al.[29]: snRNA-seq data from the mouse spinal cord, collected using the 10x Genomics platform after 30 min of walking with or without EES. This dataset is used to demonstrate the application of Augur to an RNA velocity matrix to specifically prioritize immediate responses to perturbation on the timescale of transcription.

3 Bhattacherjee et al.[62]: scRNA-seq data from the prefrontal cortex of mice after exposure to a cocaine addiction paradigm, followed by maintenance or withdrawal for 48 h or 15 d, collected using the 10x Genomics platform. This dataset is used to demonstrate the use of Augur to perform cell-type prioritization with more than two conditions (that is, multiclass classification).

4 Moffitt et al.[35]: multiplexed error-robust fluorescence in situ hybridization (MERFISH) data from the hypothalamus of male and female mice, before and after a range of social behaviors. This dataset is used to demonstrate the application of differential prioritization to identify cell types preferentially activated during one perturbation or in one background, and as an example of cell-type prioritization in a single-cell modality other than RNA-seq.

5 Synthetic data: we simulated scRNA-seq data from a tissue with five cell types, sequenced in two experimental batches, to illustrate the application of Augur to a dataset with significant batch effects. This dataset is used to discuss the specific types of batch effects that can confound cell-type prioritization in a scenario where the ground truth is known, and demonstrate how batch effect correction can restore accurate prioritizations.

**Table 2 | Case study datasets**

| Case study | Publication | Accession | Description | Replicates | Cells | Cell types | Protocol |
|---|---|---|---|---|---|---|---|
| 1 | Kang et al.[14] | GSE96583 | PBMCs before and after stimulation with recombinant IFN-β | 8 | 24,673 | 8 | 10x |
| 2 | Skinnider et al.[29] | GSE142245 | Neurons from the lumbar segment of mice walking after spinal cord injury, with or without targeted electrical epidural stimulation of the lumbar spinal cord (TESS) | 6 | 6,171 | 39 | 10x |
| 3 | Bhattacherjee et al.[62] | GSE124952 | Prefrontal cortices of mice addicted to cocaine, then exposed to withdrawal for 48 h, 15 d, or maintained on cocaine | 6 | 12,936 | 8 | 10x |
| 4 | Moffitt et al.[35] | doi:10.5061/dryad.8t8s248 | Hypothalamic preoptic region in naive and parenting male and female mice | 19 | 566,446 | 83 | MERFISH |

## Materials

### Equipment

**Hardware**

- At least 16 GB of RAM
- (Optional) For RNA velocity calculation, at least 32 GB of RAM

**Software**

- R version 3.6.0 or later
- R packages:
  - sparseMatrixStats (https://github.com/const-ae/sparseMatrixStats)
  - Augur (https://github.com/neurorestore/Augur)
  - tidyverse (https://www.tidyverse.org)
  - Seurat (https://satijalab.org/seurat)
  - (Optional) velocyto.R (https://github.com/velocyto-team/velocyto)

**Data (optional)**

To demonstrate Augur, we detail case studies of five datasets. The first is provided as a raw count matrix, which is preprocessed from scratch to illustrate the process of cell-type annotation. The second is provided as separate matrices of intronic and exonic read counts, with an accompanying metadata table. The other three datasets are provided in preprocessed form as Seurat objects. Vignettes containing the R code used to construct these objects from publicly available data are provided in Supplementary Notes 1–4.

Alternatively, users can specify their own input in one of four formats:

1 As a matrix, sparse matrix or data frame with cells in columns, accompanied by a metadata data frame with cells as row names
2 As a Seurat object
3 As a monocle3 object
4 As a SingleCellExperiment object

The names of the columns of the metadata data frame that contain cell-type labels and sample labels, respectively, can be specified using the 'cell_type_col' and 'label_col' arguments to 'calculate_auc'.

### Equipment setup

**Software installation**

R can be installed by following the instructions available at https://cran.r-project.org/doc/FAQ/R-FAQ.html#How-can-R-be-installed_003f. Alternatively, R can be obtained through the RStudio integrated development environment (IDE), available at https://rstudio.com/products/rstudio/. The code in this protocol has been tested with R version 3.6.0.

To install Augur from GitHub, start R and enter the following commands:

```
install.packages("devtools")
devtools::install_github("Bioconductor/MatrixGenerics")
```

```
devtools::install_github("const-ae/sparseMatrixStats")
devtools::install_github("neurorestore/Augur")
```

This protocol additionally uses functions from the tidyverse R package. To install tidyverse, enter the following command:

```
install.packages("tidyverse")
```

To follow the aspects of the protocol that concern preprocessing of raw expression data (including normalization, dimensionality reduction, data integration, clustering, marker gene identification, and visualization), or to use the preprocessed example datasets provided with the protocol, users will also require the Seurat R package. To install Seurat, enter the following command:

```
install.packages("Seurat")
```

▲**CRITICAL** The anticipated results presented in this protocol were generated with R version 3.6.0 and Seurat version 3.1.5. Users may obtain slightly different results with other versions of R or Seurat.
To calculate gene-specific RNA velocity estimates, as discussed in case study 2, install the velocyto.R package by entering the following commands:

```
devtools::install_github("hredestig/pcaMethods")
devtools::install_github("velocyto-team/velocyto.R")
```

▲**CRITICAL** The velocyto.R package has a number of system dependencies, including the boost and openmp libraries. For additional details on these dependencies and how to install them, users should consult the velocyto.R documentation, available from https://github.com/velocyto-team/velocyto.R, or the issues tracker at https://github.com/velocyto-team/velocyto.R/issues.

Finally, to perform batch effect correction using the methods demonstrated in case study 5, users will require the BiocNeighbors, scater and batchelor R packages. To install these packages, enter the following commands:

```
devtools::install_github("LTLA/BiocNeighbors")
BiocManager::install("scater")
BiocManager::install("batchelor")
```

**Input datasets**

The processed and analysis-ready input data discussed in this protocol can be obtained from Zenodo at https://doi.org/10.5281/zenodo.4473025. To obtain these data, navigate to this URL in a web browser, and download the file 'augur_protocol_zenodo.tar.gz'. Then, extract the .tar.gz file. On Unix systems, for example, this can be done using the following command:

```
tar -xzvf augur_protocol_zenodo.tar.gz --strip 1
```

▲**CRITICAL** The file paths specified in the following section assume you have entered the directory containing the input files prior to starting an R session. If you have not done so, set your working directory within R. For instance, if you have downloaded the example data from Zenodo in the directory 'augur_protocol', run the following command:

```
setwd("augur_protocol")
```

▲**CRITICAL** If you wish to follow along with the protocol in an R session, please note that copying code directly from a PDF can introduce unexpected errors in R, due to the formatting of the PDF. To prevent errors of this nature, we provide a Markdown document containing all of the code presented in this protocol in a format that can readily be copied and pasted into an R session (Supplementary Note 5)
▲**CRITICAL** Users should be aware that the runtimes presented throughout this protocol are estimates that may vary depending on the CPU used to perform the computations.

## Procedure

### Case study 1: interferon-stimulated PBMCs (cell-type annotation and basic cell-type prioritization)

▲CRITICAL Our first case study makes use of scRNA-seq data from PBMCs stimulated with interferon, compared with unstimulated cells[14]. We use this dataset to demonstrate a workflow for cell-type annotation using the Seurat R package, basic cell-type prioritization with Augur, and several different ways to visualize the results.

**Preprocessing and cell-type annotation ● Timing ~20 min**

▲CRITICAL Steps 1–7 describe a standard workflow for preprocessing and cell-type annotation of a count matrix using the Seurat package (https://satijalab.org/seurat/vignettes.html). To skip this workflow, the user can simply load the preprocessed Seurat object using the command `sc = readRDS("rnaseq/processed/Kang2018.rds")` and proceed directly to Step 8.

1 Open R and load all of the necessary libraries by entering the following commands:

```
library(tidyverse)
library(Seurat)
library(Augur)
```

2 Load the preprocessed data downloaded from Zenodo:

```
input_dir = "rnaseq/raw"
mat = readRDS(file.path(input_dir, "Kang2018_mat.rds"))
meta = readRDS(file.path(input_dir, "Kang2018_meta.rds"))
# create the Seurat object
sc = CreateSeuratObject(mat, min.cells = 3, min.features = 0,
  meta.data = meta)
# confirm dimensions of the object
dim(sc)
# [1] 15706 24673
# print the experimental conditions
unique(sc$label)
# [1] "ctrl" "stim"
```

3 Perform integration of the data from control and stimulated conditions.

```
sc = sc %>%
  # Split the object into a list for input the Seurat integration
  SplitObject(split.by = 'label') %>%
  # Normalize the data using regularized negative binomial models
  map(~ SCTransform(.)) %>%
  # Use Seurat to find anchors across the conditions (baseline/stim)
  PrepSCTIntegration(
  anchor.features = SelectIntegrationFeatures(.)) %>%
  FindIntegrationAnchors(
  anchor.features = SelectIntegrationFeatures(.),
  normalization.method = 'SCT') %>%
  # Integrate data
  IntegrateData(normalization.method = 'SCT')
```

▲CRITICAL STEP Augur requires that cell types be present across experimental conditions in order to perform cell-type prioritization. This integration step ensures that the data are aligned across conditions, meaning that cells cluster by cell type rather than by condition, and thereby promote the accurate identification of cell-type clusters. For more details on individual steps, users may wish to consult the Seurat vignette at https://satijalab.org/seurat/v3.2/immune_alignment.html.

4 Perform dimensionality reduction of the integrated dataset, as input to the clustering step.

```
sc = sc %>%
  # Run principal component analysis
  RunPCA(npcs = 30, verbose = F) %>%
  # Embed in two dimensions
  RunUMAP(dims = 1:20, do.fast = T)
```

**▲CRITICAL STEP** Users may need to adjust the number of dimensions used in Steps 4 and 5 according to the intrinsic dimensionality of their dataset. It may be informative to inspect the proportion of variance explained by each principal component; within Seurat, this can be achieved using the `ElbowPlot` function. Seurat also includes a permutation test to identify statistically significant principal components, using the `JackStraw` function; see the vignette at https://satijalab.org/seurat/articles/pbmc3k_tutorial.html for further details. We encourage users to experiment with these parameters as necessary.

5 Cluster the dataset to identify cell-type clusters by constructing a *k*-nearest-neighbor network, then performing graph-based clustering on the network.

```
sc = sc %>%
  FindNeighbors(dims = 1:10) %>%
  FindClusters(resolution = 0.4)
```

**▲CRITICAL STEP** Users may also need to adjust the resolution with which the dataset is clustered (parameter `resolution` in function `FindClusters`). To guide the selection of an appropriate resolution, users may wish to inspect marker genes from each cluster and plot reduced-dimensionality representations of the dataset, such as uniform manifold approximation and projection (UMAP) visualizations[63,64]. We also discuss how Augur may be used in combination with clustering trees to refine the selection of the resolution parameter in Box 4.

6 Manually annotate the cell-type identities of each cluster, using marker genes to guide annotation. This may involve performing a test for differential expression to find marker genes of each cluster de novo, or inspecting the average expression of known marker genes for the system of interest in each cluster.

In addition to experimenting with different clustering resolutions, users may also need to manually merge clusters, or subcluster individual clusters by repeating Step 5 for a subset of the data. Below, we show an example of this by manually assigning several clusters to the cell type 'CD4 T cells'. We emphasize that appropriate cell-type annotation requires domain-specific knowledge, and best practices will depend both on the biological system under investigation and the biological question of interest.

```
# Find marker genes for cluster 7
markers = FindConservedMarkers(sc, ident.1 = 7, grouping.var = "label")
# Print the top 10 markers for cluster 7
rownames(markers)[1:10]
# [1] "FCGR3A" "FAM26F" "VMO1" "GBP5" "TNFSF10" "C3AR1" "ATP1B3" "MS4A7"
"CFD" "SERPINA1"
# Repeat these lines for each cluster in turn
# Alternatively, inspect some known marker genes
FeaturePlot(sc, features = c("FCGR3A", "LYZ", "MS4A1", "NKG7"))
## See Fig. 3c


# Annotate the clusters
sc = RenameIdents(sc,
  `0` = "CD4 T cells",
  `1` = "CD14+ Monocytes",
  `2` = "CD4 T cells",
  `3` = "B cells",
  `4` = "NK cells",
  `5` = "CD8 T cells",
  `6` = "CD4 T cells",
```

## Box 4 | Cell-type prioritization on clustering trees

When investigators perform a differential expression analysis between experimental conditions, the output from the analysis is at the level of genes, entities whose genomic coordinates and primary sequence are widely accepted. In contrast, Augur returns output at the level of cell types, which are more subjectively defined entities, generally assigned by manual annotation of each cell's high-dimensional molecular profile. In many systems, there may be multiple biologically relevant resolutions at which cell types can be defined. Moreover, even within classically defined cell types, individual cells may exhibit unexpected heterogeneity. The subjectivity inherent in current workflows for cell-type annotation has a critical impact on cell-type prioritization, in that it defines the scope of prioritizations that Augur is capable of returning.

Considering multiple possible clustering solutions can help users avoid common pitfalls in cell-type annotation, particularly as they impact Augur. One such issue is selecting the appropriate biological resolution for the question of interest. As one example, is the user interested in the perturbation response at the level of neurons, interneurons, excitatory interneurons or excitatory interneurons from specific cortical layers? Another benefit of considering multiple clustering solutions is the possibility of diagnosing errors in either the clustering procedure itself or the manual annotation of these clusters, particularly if one particular clustering solution produces anomalous Augur results.

Clustering trees[44] provide a means to visually compare multiple clustering solutions. To produce a clustering tree, a dataset is clustered at various resolutions. The clustering results are then sorted by resolution, and the overlap between clusters at adjacent resolutions is calculated. This calculation provides a basis to construct a graph in which each node is a cluster, and edges are drawn between clusters whose overlap exceeds a user-specified threshold. Visualizing this graph provides a global overview of how the clustering solution changes with the resolution parameter. Clustering trees provide a framework for assessing and comparing clustering solutions that can be used without any further information. However, we have found that overlaying Augur results onto a clustering tree can further help refine and select an optimal resolution. In cases when more than one different biological resolution is reasonable a priori, the combination of Augur with a clustering tree provides a means to directly compare cell-type prioritizations across resolutions. This combination can also reveal potential pitfalls; for instance, whereas a slow increase in the AUC moving down one branch of the clustering tree suggests a convergence on a particular subpopulation mediating the perturbation response, an abrupt transition between resolutions could reflect a problem with the clustering. Similarly, if the hierarchy of cell-type annotations does not align with prior biological knowledge, the user may wish to revisit their annotations of each cluster.

The following commands provide an example of how to plot a clustering tree, using the dataset from case study 2 (ref. [29]), and overlay the AUCs computed by Augur at multiple different resolutions onto the tree. In this case study, we have clustered the dataset using the Seurat function `FindClusters`, with the `resolution` parameter set to either 0.01, 0.1, 0.5, 1, 2 or 4. However, the 'clustree' package is agnostic to the specific clustering algorithm applied to the data. Users may find these values of the resolution parameter provide a useful starting point to construct a clustering tree using Seurat. However, these values may need to be adjusted for any given dataset, to identify a range of resolutions over which the total number of clusters identified changes substantially between adjacent resolutions. The code below uses the clustree package to visualize changes in the AUC over varying resolutions; however, other approaches to visualization are possible[80].

```
# load in Seurat object containing multiple levels of clustering information
sc = readRDS("rnaseq/processed/Skinnider2020_tree.rds")
# load in AUCs computed for each resolution
augur_tree = readRDS("rnaseq/processed/Skinnider2020_tree_aucs.rds")
# use clustree to generate the raw clustering tree object
# note that you will need to set the variable name used in the multi-level
# clustering: here, we use "name_snn_res"
library(clustree)
layout = clustree(sc, prefix = "name_snn_res.", return = "layout",
                  prop_filter = 0.15)
# add a 'node' column to index this to the tree
augur_tree %<>%
  mutate(node = paste0(res, "C", cell_type)) %>%
  arrange(node)
# add the AUC in a separate column
stat = layout %>%
  left_join(augur_tree %>% select(node, auc)) %>%
  mutate(auc = ifelse(is.na(auc), 0.5, auc))
layout$auc = stat$auc[match(layout$node, stat$node)]
# plot the graph with AUCs overlaid (modified from clustree.R)
gg = ggraph(layout) +
  geom_edge_link(arrow = arrow(length = unit(0.2 * 5, "points"),
                               ends = "last"),
                 end_cap = circle(5.5 * 1.5, "points"),
                 start_cap = circle(5.5 * 1.5, "points"),
                 aes_(colour = 'black',
                      alpha = ~ in_prop,
                      edge_width = ~ is_core)) +
  scale_edge_width_manual(values = c(.2, .2), guide = F) +
  scale_edge_colour_manual(values = 'black', guide = F) +
  scale_edge_alpha(name = 'In-proportion (%)', labels = function(x) x * 100,
                   limits = c(0, 1)) +
  scale_color_paletteer_c("grDevices::RdYlBu", direction = -1,
                          name = "AUC", breaks = seq(0.5, 1, 0.1),
                          limits = c(0.425, 0.85)) +
  scale_size_continuous('Cells', range = c(3, 6), breaks = c(500, 1e3, 5e3)) +
  clustree:::add_node_points("auc", "size", 1, colnames(layout)) +
  geom_node_text(aes_(label = ~cluster), size = 1.5, colour = "black") +
  guides(color = guide_colorbar(nbin = 10, ticks = F, raster = F)) +
  ggraph::theme_graph(base_family = "",
                      plot_margin = ggplot2::margin(rep(0, 4))) +
```

**Box 4 | Cell-type prioritization on clustering trees (Continued)**

```
  theme(legend.position = 'top',
        legend.key.height = unit(0.4, "lines"),
        legend.key.width = unit(0.7, "lines"),
        legend.text = element_text(size = 5),
        legend.title = element_text(size = 5))
gg
## see Fig. 4e
```

```
  `7` = "FCGR3A+ Monocytes",
  `8` = "Dendritic cells",
  `9` = "B cells",
  `10` = "Megakaryocytes",
  `11` = "CD4 T cells",
  `12` = "Dendritic cells")
# Add cell type annotations into the metadata of the Seurat object
sc$cell_type = Idents(sc)
```

▲**CRITICAL STEP** If manual annotation reveals clusters of cells that do not express any marker genes, or express unusual combinations of markers, the user may wish to consider discarding these clusters as potential low-quality cells or doublets.
▲**CRITICAL STEP** The use of the `RenameIdents` function to annotate clusters above is specific to the clusters returned by `FindClusters` using R version 3.6.0 and Seurat version 3.1.5, and may need to be adjusted for different versions of Seurat if these return different clustering solutions.
▲**CRITICAL STEP** During this step, the user may find it helpful to plot a low-dimensionality visualization of the dataset that includes the clusters identified. For example, a UMAP visualization can be generated by running the following command:

```
DimPlot(sc, label = TRUE)
## See Fig. 3a
```

7 After data integration, Seurat will use the integrated expression matrix as the default assay in all future operations performed on the object. However, this matrix contains only the subset of genes used to align the raw expression data across samples, comprising a relatively small subset of the original gene expression matrix. To provide the full gene expression matrix as input to Augur, reset the default assay in the Seurat object:

```
DefaultAssay(sc) = "RNA"
```

**Cell-type prioritization ● Timing ~15 min**

8 Now that the Seurat object contains experimental conditions (in the `label` column) and cell-type annotations (in the `cell_type` column) for each cell, cell-type prioritization can be performed. To run Augur using default settings, on four cores, use the following command:

```
augur = calculate_auc(sc)
```

Alternatively, if the input data are not in a Seurat, monocle or SingleCellExperiment object, the user can pass in the expression matrix and the accompanying metadata data frame separately, as described above in ‘Overview of the procedure’. For instance, these can be extracted from the Seurat object and provided as input into Augur directly, using the following commands:

```
expr = GetAssayData(sc)
meta = sc@meta.data
augur = calculate_auc(input = expr, meta = meta)
```

This will produce identical results.
**? TROUBLESHOOTING**

9 (Optional) Augur has a number of parameters that can be specified by the user, which are enumerated in Table 1. These parameters control either how the input is processed by Augur or how cell-type prioritization occurs. For instance, to run Augur on a Seurat object in which the experimental conditions are in a column named `condition`, and cell types are in a column named `cluster`, run the following command:

```
augur = calculate_auc(sc, label_col = "condition", cell_type_col =
"cluster")
```

By default, Augur uses a random forest classifier to perform cell-type prioritization. As discussed above, this has a number of advantages. Notably, the use of a decision-tree-based classifier eliminates any assumptions about the distributions of the input features, and confers robustness to the inclusion of noisy input features by automatically identifying genes whose expression is informative about the perturbation. In certain cases, however, it can be useful to employ a second classifier as a check of the robustness of the results. To use penalized logistic regression instead of a random forest classifier within Augur, run the following command:

```
augur = calculate_auc(sc, classifier = "lr")
```

As a final example, to increase the number of subsamples performed within Augur from 50 (default) to 100, run the following command:

```
augur = calculate_auc(sc, n_subsamples = 100)
```

Note that increasing the number of subsamples will increase the runtime of the function.
In general, Augur is highly robust to the values of these parameters, and consequently, we recommend varying them primarily to evaluate the robustness of the obtained prioritizations. An important exception is the `trees` parameter, which determines the number of trees in the random forest. We found that although varying the number of trees minimally affects the relative rankings of the various cell types, this parameter effectively controls the dynamic range of AUCs obtained in any given dataset. Because cell-type prioritization is most effective when the distribution of AUCs spans a wide range, we recommend increasing the number of trees in scenarios where all AUCs are close to 0.5 (for instance, cells undergoing an exceptionally subtle perturbation, or very sparsely sequenced datasets), and decreasing the number of trees when all AUCs are close to 1. Some other common pitfalls in parameter setting are highlighted in the Troubleshooting section.
**? TROUBLESHOOTING**

**Interpret and visualize results ● Timing ~10 min**

10 The primary output from Augur is a ranked list of all cell types in the input dataset, sorted in descending order by their AUCs. To view a list of the top-ranked cell types, enter the following command:

```
print(augur$AUC)
```

Guidance on interpreting the AUCs is provided in Box 5.
**? TROUBLESHOOTING**

11 Augur also includes several methods to visualize cell-type prioritizations. For example, the ranked list of AUCs for each cell type can be plotted as a so-called lollipop plot:

```
# Plot a lollipop plot
plot_lollipop(augur)
## See Fig. 3f
```

Alternatively, the AUC for each cell type can be overlaid onto the UMAP, or another low-dimensional representation. In addition to the Augur output, this function requires a processed Seurat, monocle3 or SingleCellExperiment object as input, which includes a low-dimensional representation of the data. UMAP is used as the default dimensionality reduction, but an alternative dimensionality reduction technique can be specified using the `reduction` argument.

**Box 5 | Interpreting the AUC**

The AUC provides a quantitative measure of the classifier's ability to predict which experimental condition a cell came from, based on measurements of gene expression in that cell. For cell types that are completely unaffected by a perturbation, the classifier should do no better than random guesses. On the other hand, for cell types that undergo a profound perturbation response, the classifier should readily learn to predict whether a cell came from a treated or control sample. This intuition is captured by the AUC on a scale that ranges from 0 to 1, reflecting the probability that any given cell from the perturbed condition is ranked higher by the classifier than any given cell from the unperturbed condition. Thus, an AUC of 0.5 corresponds to random chance, with half of perturbed cells ranked higher than unperturbed cells. Conversely, an AUC of 1.0 corresponds to perfect accuracy, with every perturbed cell ranked higher than every unperturbed cell.

Depending on the underlying biological question, both the absolute value of the AUC and the relative rankings of each cell type will likely be of interest. The absolute magnitude of the AUC reflects the intensity of the perturbation response. An AUC close to 1, for instance, reflects a near-perfect ability to distinguish perturbed from unperturbed cells, which in turn suggests that these cells undergo a profound response to the perturbation. With subtler biological perturbations, however, it may simply not be possible to achieve an AUC >0.6. In this case, it will likely be more instructive to consider the relative rankings of the cell types in the biological system. For instance, investigators may wish to plan follow-up experiments in one or a handful of the top-ranked cell types. Additionally, when interpreting Augur results or planning follow-up experiments, it can be very helpful to consider the magnitude of separation between cell types that are adjacent in the rankings. For instance, if Augur achieves an AUC of 0.7 for the top-ranked cell type, but no other cell type exceeds an AUC of 0.6, this would provide a basis to hypothesize that the top-ranked cell type is undergoing a much more profound perturbation response than any other cell type in the biological system, and direct follow-up experiments accordingly. On the other hand, if Augur achieves nearly identical AUCs for several of the top-ranked cell types, this would suggest there is little basis to pick out a single cell type as most profoundly affected.

```
# Overlay AUCs onto a UMAP
plot_umap(augur, sc)
## See Fig. 3g
```

Or, if the user is more interested in the rank of each cell type than in the numeric values of the AUC, the `mode` argument to the `plot_umap` function can be adjusted:

```
# Overlay cell type prioritization ranks onto UMAP instead
plot_umap(augur, sc, mode = "rank")
## See Fig. 3h
```

12 (Optional) Other outputs from Augur include (i) a larger suite of classification evaluation metrics, calculated on each fold of each subsample, and (ii) the feature importance assigned to each gene in each fold. If desired, these can be inspected using the following commands:

```
# Inspect all metrics for each cross-validation fold
str(augur$results)
# Inspect feature importance for each cross-validation fold
str(augur$feature_importance)
```

## Case study 2: walking after paralysis with spinal cord neurostimulation (RNA velocity)

▲ **CRITICAL** Our second case study makes use of snRNA-seq data from the spinal cord of mice walking after paralysis with or without neurostimulation of the lumbar spinal cord[29]. We reanalyze this dataset to demonstrate the application of Augur to prioritize cell types on the basis of RNA velocity estimates (discussed in Box 1). For further details on the velocyto R package, the user is referred to the documentation available at http://velocyto.org.

### RNA velocity calculation ● Timing ~30 min

13 Open a new R session, and load all of the necessary libraries by entering the following commands:

```
library(tidyverse)
library(Seurat)
library(Augur)
library(Matrix)
library(velocyto.R)
```

14 Load the exonic and intronic read count matrices, and accompanying metadata, from the Zenodo download:

```
input_dir = "rnaseq/processed"
mats = readRDS(file.path(input_dir, "Skinnider2020_mats.rds"))
meta = readRDS(file.path(input_dir, "Skinnider2020_meta.rds"))
# Print the names of the matrices
names(mats)
# [1] "spliced" "unspliced"
# Confirm the number of neuronal subtypes
n_distinct(meta$cell_type)
# [1] 39
```

15 Filter lowly expressed genes from the exonic and intronic read count matrices.

```
emat = filter.genes.by.cluster.expression(
        mats$spliced,
        setNames(meta$cell_type, meta$barcode),
        min.max.cluster.average = 0.005)
nmat = filter.genes.by.cluster.expression(
        mats$unspliced,
        setNames(meta$cell_type, meta$barcode),
        min.max.cluster.average = 0.004)
```

▲ **CRITICAL STEP** The user may need to adjust the parameter `min.max.cluster.average`, which specifies the minimum average expression, on the basis of the sequencing depth of their dataset.

16 Estimate the time derivative of gene expression for each gene in each cell using velocyto. Note that this function will automatically filter genes for which reliable estimates of the RNA velocity could not be calculated.

```
vel = gene.relative.velocity.estimates(emat, nmat, deltaT = 1,
kCells = 10,
  fit.quantile = 0.01)
input = Matrix(vel$deltaE, sparse = TRUE)
# Print the number of genes and cells in the RNA velocity matrix
dim(input)
# [1] 2225 6171
# Confirm this matches the number of cells in the metadata table
dim(meta)
# [1] 6171 4
```

▲ **CRITICAL STEP** The parameters `deltaT`, `kCells` and `fit.quantile` control the calculation of RNA velocity, which in turn may impact cell-type prioritization. For further information, users are encouraged to consult the velocyto.R documentation.

**Cell-type prioritization ● Timing ~1 h**

17 Cell-type prioritization can now be performed using the RNA velocity matrix as input. Since the RNA velocity calculation has already performed feature selection, discarding both lowly expressed genes and genes for which the RNA velocity could not be reliably estimated, the feature selection procedure within Augur is disabled by specifying `augur_mode = "velocity"` when calling `calculate_auc`.

```
augur = calculate_auc(input = input, meta = meta, augur_mode
= "velocity")
```

The output can then be inspected and visualized using the same code as presented in case study 1 (Steps 11–12).

### Case study 3: prefrontal cortex in cocaine withdrawal (multiclass classification)

▲ **CRITICAL** In our third case study, we analyze scRNA-seq data from the prefrontal cortex of mice exposed to a cocaine self-addiction paradigm, followed by either maintenance of cocaine, a 48 h withdrawal period or a 15 d withdrawal period[62]. We use these data to illustrate how Augur can be applied to analyze experimental designs with more than two conditions (discussed in Box 2).

**Pairwise cell-type prioritization ● Timing ~1 h 30 min**

18 Open a new R session, and load the necessary libraries:

```
library(tidyverse)
library(magrittr)
library(Seurat)
library(Augur)
```

19 Load the preprocessed data downloaded from Zenodo:

```
input_dir = "rnaseq/processed"
sc = readRDS(file.path(input_dir, "Bhattacherjee2019.rds"))
# Print the number of cell types
n_distinct(sc$cell_type)
# [1] 8
# Print the experimental conditions
unique(sc$label)
# [1] "withdraw_15d_Cocaine" "Maintenance_Cocaine" "withdraw_48h_Cocaine"
```

20 This dataset contains three different experimental conditions: mice maintained on cocaine or subjected to withdrawal for 48 h or 15 d. Arguably the simplest way to analyze these data, then, is to use Augur to prioritize cell types in each pairwise comparison (that is, maintenance versus 48 h, maintenance versus 15 d, and 48 h versus 15 d). To run Augur on each of these three pairwise comparisons in turn, enter the following commands:

```
# Begin by specifying the comparisons of interest
comparisons = list(
  c("Maintenance_Cocaine", "withdraw_48h_Cocaine"),
  c("Maintenance_Cocaine", "withdraw_15d_Cocaine"),
  c("withdraw_48h_Cocaine", "withdraw_15d_Cocaine"))
# Create a list to store Augur results
pairwise_results = list()
# Run Augur on each comparison
for (comparison in comparisons) {
  # Subset the Seurat object to these two experimental conditions only
  Idents(sc) = sc$label
  sub = subset(sc, idents = comparison)
  # Run Augur
  augur = calculate_auc(sub)
  # Store the results
  comparison_name = paste(comparison, collapse = ":")
  pairwise_results[[comparison_name]] = augur
  }
```

To extract and combine only the final results (that is, the ranked list of AUCs for each cell type) from each pairwise comparison into a single data frame, run the following command:

```
pairwise_aucs = pairwise_results %>%
  map(extract2, "AUC") %>%
  bind_rows(.id = "comparison")
print(pairwise_aucs)
```

**Multiclass cell-type prioritization ● Timing ~30 min**

21 The code above demonstrates how the user can compare each pair of experimental comparisons separately and in turn. However, it is also possible to apply Augur to directly compare all three experimental comparisons at the same time. In this framework, called multiclass classification, Augur attempts to train a random forest classifier to predict whether a given cell was obtained from the maintenance, 48 h or 15 d condition (Box 2). To run Augur in multiclass mode, simply provide the entire Seurat object as input. Augur will automatically detect the presence of three different experimental conditions.

```
multiclass_augur = calculate_auc(sc)
```

22 A natural question then arises: which set of Augur results to use—those from all pairwise comparisons or from the multiclass classification framework? Although the answer to this question will depend on the underlying biological question of interest, the user may find it helpful to directly compare the results from each application of Augur. The comparison can be visualized as a scatterplot, using the `plot_scatterplot` function. For example, the following command will plot the multiclass AUC for each cell type on the *x*-axis, and the binary AUC in a pairwise comparison of maintenance and 48 h on the *y*-axis:

```
plot_scatterplot(
multiclass_augur, pairwise_results[["Maintenance_Cocaine:
withdraw_48h_Cocaine"]])
## See Fig. 5f
```

## Case study 4: sex-specific differences in mouse parenting (differential prioritization)

▲ CRITICAL In our fourth case study, we analyze single-cell transcriptome imaging data from the mouse hypothalamus, acquired using MERFISH[35]. We reanalyze this dataset to demonstrate the use of differential prioritization (discussed in Box 3), by comparing hypothalamic neuronal subtypes preferentially activated during parenting in male or female mice. This case study also provides an example of the application of Augur to single-cell genomics assays other than single-cell RNA-seq.

### Cell-type prioritization in male and female mice ● Timing ~1 h 20 min

23 Open a new R session, and load the necessary libraries:

```
library(tidyverse)
library(magrittr)
library(Seurat)
library(Augur)
```

24 Load the preprocessed data downloaded from Zenodo:

```
input_dir = "spatial/processed"
sc = readRDS(file.path(input_dir, "Moffitt2018.rds"))
# Print the number of cell types
n_distinct(sc$cell_type)
# [1] 83
# Print the experimental conditions
unique(sc$Behavior)
# [1] "Naive" "Parenting" "Virgin Parenting"
[4] "Aggression to pup" "Aggression to adult" "Mating"
# Check the different animal sexes available
unique(sc$Animal_sex)
[1] "Female" "Male"
```

25 This study applied MERFISH to study the brain of mice after multiple different social behaviors, including parenting, aggression and mating. In this case study, we will focus only on identifying neuronal subtypes with transcriptional perturbations induced by parenting. To subset the dataset accordingly, enter the following commands:

```
comparison = c("Naive", "Parenting")
Idents(sc) = sc$Behavior
sc = subset(sc, idents = comparison)
```

26 To provide a basis for differential prioritization, we will calculate the AUC separately in male and female mice after parenting, using a common control group including all naive mice. To run Augur on MERFISH data from male and female mice in turn, enter the following commands:

```
observed_results = list()
sexes = c("Female", "Male")
for (sex in sexes) {
  # Subset the Seurat object
  cell_ids = sc@meta.data %>%
  rownames_to_column(var = 'cell') %>%
  filter(Behavior == "Naive" | Animal_sex == sex) %>%
  pull(cell)
  sc_sex = sc %>% subset(cells = cell_ids)
# Set the labels
  sc_sex$label = sc_sex$Behavior
# Run Augur
  augur = calculate_auc(sc_sex)
  # Store the results
  observed_results[[sex]] = augur
  }
```

To extract and combine the AUCs for parenting in male and female mice, run the following command:

```
aucs = observed_results %>%
  map(extract2, "AUC") %>%
  bind_rows(.id = "sex")
print(aucs)
```

**Differential prioritization ● Timing ~8 h**

27 The AUCs for parenting in male and female mice provide an initial indication of which neuronal subpopulations might be preferentially prioritized in either sex. However, to select a small number of neuronal subtypes for experimental follow-up, it is often desirable to evaluate the statistical significance of the ΔAUC, in addition to its absolute value. This can be achieved using a permutation test, which allows the null distribution of the ΔAUC to be empirically estimated for each cell type (Box 3). Here, we demonstrate the use of a new, faster implementation of this permutation test, using the argument `augur_mode = 'permute'`, which provides roughly a 100-fold increase in speed over our original implementation[29] while producing nearly identical results (Extended Data Fig. 1). Nonetheless, due to the high computational burden of permutation, we adjust the `n_threads` argument to run the permutation test using eight threads instead of four. To calculate a distribution of permuted AUCs for each sex, performing a total of 500 iterations of cross-validation in subsamples of cells, enter the following commands:

```
permuted_results = list()
for (sex in sexes) {
  # Subset the Seurat object
  cell_ids = sc@meta.data %>%
  rownames_to_column(var = 'cell') %>%
  filter(Behavior == "Naive" | Animal_sex == sex) %>%
  pull(cell)
  sc_sex = sc %>% subset(cells = cell_ids)
  # Set the labels
  sc_sex$label = sc_sex$Behavior
  # Run Augur in permutation mode
  augur = calculate_auc(
  sc_sex,
    n_threads = 8,
    augur_mode = 'permute')
  # Store the results
  permuted_results[[sex]] = augur
  }
```

28 The observed AUCs can now be compared with these permuted results to estimate the statistical significance of the ΔAUC between male and female mice, using the following command:

```
pvals = calculate_differential_prioritization(augur1 = observed_results$Male, augur2 = observed_results$Female, permuted1 = permuted_results$Male, permuted2 = permuted_results$Female)
```

29 To print all neuronal subtypes with a nominally significant ΔAUC, enter the following command:

```
pvals %>% filter(pval < 0.05) %>% arrange(pval)
```

Alternatively, to print only those neuronal subtypes with a statistically significant ΔAUC after correcting for multiple hypothesis testing using the Benjamini–Hochberg procedure, enter the following command:

```
pvals %>% filter(padj < 0.05) %>% arrange(pval)
```

30 Finally, to plot these results, use the plot_differential_prioritization function:

```
plot_differential_prioritization(pvals)
## See Fig. 5g
```

## Case study 5: cell-type prioritization with batch effects

▲ **CRITICAL** In our fifth and final case study, we analyze simulated scRNA-seq data with a severe batch effect. We use this dataset to illustrate how the presence of a batch effect may confound cell-type prioritization. We also show how methods for cell-type prioritization based on counting the number of DE genes within each cell type are biased toward more abundant cell types. Finally, we demonstrate the application of an exemplary method for batch effect correction to the confounded dataset, and show that this correction restores the correct cell-type prioritization. We provide additional context for this case study, and a full description of the underlying simulations, in Box 6.

### Cell-type prioritization with a confounding batch effect ● Timing ~15 min

31 Open a new R session, and load the necessary libraries:

```
library(tidyverse)
library(magrittr)
library(Seurat)
library(Augur)
library(batchelor)
library(scater)
```

## Box 6 | Cell-type prioritization in simulated data with a confounding batch effect

In our original description of Augur[29], we performed a series of simulation studies to investigate the impact of batch effects on cell-type prioritization. We analyzed the performance of cell-type prioritization in the presence of a technical factor that is either independent of, partially confounded with or completely confounded with the perturbation of interest. In this scenario, this technical factor is discussed as a batch effect, but it is worth emphasizing that these results generalize to any arbitrary confounding variable.

We found that Augur was remarkably robust to most types of batch effects (see Extended Data Fig. 10 in ref. [29]). When the batch effect was orthogonal to the effects of the perturbation, the 'separability' of cells of each type between conditions was effectively unchanged, and cell-type prioritization was unaffected. Similarly, when one of the two batches experienced a stronger perturbation response, the separability again remained unchanged. When one of the two batches experienced a stronger perturbation response and certain batches were more or less likely to contain cells from the unperturbed population, then the AUC began to scale with both perturbation intensity and the magnitude of the batch effect. However, the relative rankings of each cell type remained unaffected. These simulation studies thus highlighted the robustness of cell-type prioritization using Augur to many different forms of batch effects.

In case study 5, we therefore consider a fourth scenario. In this simulated dataset, perturbed and unperturbed cells of each type are unevenly represented across batches, and the magnitude of the batch effect also varies across cell types. A scenario of this nature could manifest, for example, in the context of central nervous system tissues, where neurons are much more sensitive to sample handling and dissociation protocols than other cell types. Single-cell RNA-seq data from a tissue with five cell types were simulated using Splatter[74]. These cell types are unevenly represented within the tissue. The Gini coefficient of cell-type proportions is 0.52, equivalent to the mean across 22 published scRNA-seq datasets analyzed in our original study[29]. The dataset contains only 500 genes, to reduce the computational requirements of this case study. Moreover, the dataset contains two distinct experimental batches. Most cell types in the dataset are affected by a mild batch effect. A single cell type, however, exhibits a much more pronounced batch effect. We illustrate how these parameters influence cell-type prioritization.

32 Load the preprocessed data downloaded from Zenodo:

```
input_dir = "rnaseq/processed"
sc = readRDS(file.path(input_dir, "BatchSimulation.rds")
```

33 Print some summary statistics that illustrate the experimental design of the simulated dataset, the simulation ground truth and the nature of the confounding batch effect:

```
# Print the number of cell types
n_distinct(sc$cell_type)
# [1] 5
# Print the experimental conditions and batches
table(sc$label, sc$batch)
#          Batch1    Batch2
#  Group1    2076       359
#  Group2     360      2205
#  Print the cell type frequencies and the simulated perturbation
magnitude for each cell type
table(sc$cell_type, sc$perturbation_magnitude)
#               0.1   0.3   0.5   0.7   0.9
#  CellTypeA     0     0     0     0   173
#  CellTypeB     0     0     0   425     0
#  CellTypeC     0     0  3241     0     0
#  CellTypeD     0   780     0     0     0
#  CellTypeE   381     0     0     0     0
# Similarly, print the simulated batch effect magnitude for each cell
type
table(sc$cell_type, sc$batch_effect)
#                 0.1      1
#  CellTypeA    173      0
#  CellTypeB    425      0
#  CellTypeC   3241      0
#  CellTypeD      0    780
#  CellTypeE    381      0
# Visualize the impact of the batch effects in this processed dataset
Idents(sc) = sc$cell_type
p1 = DimPlot(sc)
```

```
p1
## See Fig. 6a
Idents(sc) = sc$label
p2 = DimPlot(sc)
p2
## See Fig. 6b
Idents(sc) = sc$batch
p3 = DimPlot(sc)
p3
## See Fig. 6d
```

34 The primary approach that has been used to identify cell types responsive to a perturbation is based on a univariate analysis of DE genes. Our simulated dataset, in which the ground-truth perturbation intensity is known, provides an opportunity to illustrate one of the major deficiencies of this approach: namely, that the number of DE genes is strongly correlated to the relative abundance of each cell type. To perform cell-type prioritization based on the number of DE genes, using a Wilcoxon rank-sum test, enter the following commands:

```
# First, normalize the data using Seurat
sc %<>% NormalizeData()
# Now, perform a Wilcoxon rank-sum test for DE between conditions within
each cell      type
cell_types = unique(sc$cell_type)
DE = data.frame()
for (cell_type in cell_types) {
  # Subset the Seurat object to the relevant cell type
  Idents(sc) = sc$cell_type
  sub = sc %>% subset(idents = cell_type)
  Idents(sub) = sub$label
  # Run DE analysis, disabling all filters in Seurat
  markers = FindMarkers(sub, ident.1 = 'Group1', ident.2 = 'Group2',
                             assay = 'RNA', slot = 'data',
                             min.pct = -Inf, min.cells.feature = 0,
                             min.cells.group = 3, logfc.threshold = -Inf,
                             test.use = 'wilcox') %>%
  rownames_to_column('gene') %>%
  mutate(cell_type =!!cell_type) %>%
  # Convert Bonferroni adjusted p-values to false discovery rates
  mutate(p_val_adj = p.adjust(p_val, method = 'fdr'))
  # bind to main results bin
  DE %<>% bind_rows(markers)
}
# Print the number of identified DE genes per cell type, at 5% FDR
DE %>%
  filter(p_val_adj < 0.05) %>%
  group_by(cell_type) %>%
  dplyr::count() %>%
  arrange(desc(n))
# # A tibble:   5 ×2
# # Groups:     cell_type [5]
# cell_type       n
# <chr>        <int>
# 1 CellTypeD   338
# 2 CellTypeC   258
# 3 CellTypeA   153
# 4 CellTypeB   136
# 5 CellTypeE   15
## See Fig. 6f,g
```

The ground-truth ranking of cell types according to their response to the simulated perturbation is, in descending order, A, B, C, D and E. Differential expression analysis fails to recover the ground-truth perturbation intensity, instead nominating highly abundant cell types as being most profoundly perturbed.

35 Next, perform cell-type prioritization with Augur, using the uncorrected data as input.

```
augur = calculate_auc(sc)
# Print the cell type prioritization results
augur$AUC
# A tibble:   5 ×2
# cell_type   auc
# <chr>       <dbl>
# 1 CellTypeA 0.985
# 2 CellTypeB 0.932
# 3 CellTypeD 0.851
# 4 CellTypeC 0.846
# 5 CellTypeE 0.596
## See Fig. 6h,i
```

Applying Augur to the uncorrected data more accurately reflects the ground-truth perturbation intensity than the results of a DE analysis. However, the pronounced batch effect affecting cell type D has confounded the cell-type prioritizations, causing this cell type to be incorrectly ranked.

**Batch effect correction and cell-type prioritization ● Timing ~20 min**

36 Next, we demonstrate how computational correction of the underlying batch effects can restore accurate cell-type prioritization. Many tools are available for batch effect correction in single-cell datasets, including Seurat, Harmony[65], Scanorama[66], LIGER[67] and BBKNN[68]. Several of these methods have recently been compared in a systematic benchmark[69]. Here, we demonstrate the use of the mutual nearest neighbors method[70], as implemented in the 'batchelor' R package, to correct batch effects, but these other tools may also be appropriate. To perform batch effect correction on the simulated dataset, enter the following commands:

```
sce = sc %>% as.SingleCellExperiment() %>% logNormCounts()
corr = mnnCorrect(
  sce[rowSums(counts(sce)) > 0, sce$batch == 'Batch1'],
  sce[rowSums(counts(sce)) > 0, sce$batch == 'Batch2'])
# Extract the results and ensure they are matched to the metadata
meta = sc@meta.data
expr = as(assays(corr)@listData$corrected[, colnames(sc)],
'dgCMatrix') %>%
  extract(, rownames(meta))
augur_corrected = calculate_auc(input = expr, meta = meta)
# Print the cell type prioritization results
augur_corrected$AUC
# # A tibble: 5 ×2
# cell_type   auc
# <chr>       <dbl>
# 1 CellTypeA 0.965
# 2 CellTypeB 0.921
# 3 CellTypeC 0.915
# 4 CellTypeD 0.868
# 5 CellTypeE 0.860
## See Fig. 6j,k
```

After batch effect correction, Augur correctly recovers the simulated perturbation intensities.

## Troubleshooting

Basic troubleshooting advice can be found in Table 3.

**Table 3 | Troubleshooting table**

| Step | Problem | Possible reason | Solution |
|---|---|---|---|
| 8 | Error: labels contain missing values *or* Error: cell types contain missing values *or* Error: matrix contains missing values | Augur requires a complete dataset, meaning there are no missing values in either the labels, the cell types or the input feature matrix | Remove cells with unassigned labels/cell types, or correct label/cell-type assignments<br>Replace NAs in the feature matrix with zeroes, or remove features with NA values |
| | Error: only one label provided | Cell-type prioritization requires at least two distinct experimental conditions | Check your input dataset to ensure the `label` column includes at least two distinct experimental conditions |
| | Error: number of cells in metadata does not match number of cells in expression | Input to Augur has been supplied in the form of a feature matrix and metadata table whose dimensions do not match | Check your input dataset to ensure that the feature matrix and metadata table describe the same cells, in the same order |
| | Augur fails with an uninformative error message, such as 'all scheduled cores encountered errors in user code' | An error has been encountered on one or more cores during cell-type prioritization, when parallelizing Augur over multiple cores | Re-run `calculate_auc` with the argument `n_threads = 1` to obtain a more informative error message |
| 9 | All AUCs are close to 0.5 *or* all AUCs are close to 1 | Insufficient information is available to separate cells of any type on the basis of their perturbation response (AUCs ≈ 0.5), or enough information is available to perfectly separate cells of all types on the basis of the perturbation response (AUCs ≈ 1) | If near-perfect classification is reached for all cell types, preventing an informative ranking of the perturbation response, lower the number of trees in the random forest classifier using the `n_trees` argument. Alternatively, if only near-random classification can be reached for all cell types, increase the number of trees in the random forest classifier |
| 10 | `augur$AUC` is `NULL` | Augur has performed cell-type prioritization on continuous experimental labels using regression rather than classification, and returned a summary table in `augur$CCC` | Check the format of your input labels. For example, if you meant to perform cell-type prioritization on ordinal labels (e.g. groups 1, 2 and 3), ensure these are encoded as a factor and not as integer values |
| 8–10 | AUCs are inconsistent with biological expectation *or* are all close to 1 | The default assay of the Seurat object is set to the integrated expression matrix (`"integrated"`) | Reset the default assay in your Seurat object using the `DefaultAssay` function, or extract the desired expression matrix manually using `GetAssay` and provide this, alongside an accompanying data frame of cell-level metadata, as input to `calculate_auc` using the `input` and `meta` arguments |

More generally, if unexpected or implausible results are observed, the user may consider whether the following potential pitfalls apply to their dataset.

1 If a features matrix and metadata table have been provided separately as input, are the cell barcodes (rows in the metadata table, columns in the features matrix) in the same order in both? If not, single-cell molecular profiles may be incorrectly matched to cell types and/or experimental conditions.

2 If a Seurat, monocle or SingleCellExperiment has been provided as input, what is the default assay? By default, Augur uses the functions `GetAssayData()`, `exprs()` and `assay()`, respectively, to extract the features matrix from these objects. Thus, if the input object contains more than one assay (for example, in the case of multimodal single-cell data such as CITE-seq[71]), confirm that the appropriate function returns the expected assay. Alternatively, to run Augur on a specific assay, consider extracting this assay and providing it separately, along with the metadata table, as input.

3 Have the cell types in the dataset been appropriately clustered and labeled? Many common pitfalls result from the propagation of errors in the cell-type clustering stage. For instance, although Augur itself is robust to many types of batch effects, a failure to integrate data from different conditions or batches during the cell-type clustering stage can lead to clusters that predominantly consist of cells

from specific batches or conditions, which will preclude an accurate assessment of separability within those cell types. Tools such as kBET[72] are available to assess the integration of cells from different batches or experimental conditions; simply inspecting the proportion of cells of each type from each experimental condition may also be informative. Alternatively, cell types may have been correctly clustered, but mislabeled by manual annotation. Reclustering cell types at multiple different resolutions, then plotting Augur results on a clustering tree (Box 4), can help diagnose issues in either clustering or manual annotation.

## Timing

The computational time required to complete the basic workflow for cell-type prioritization (case study 1) is ~45 min, assuming Augur is run on four cores. Differential prioritization (case study 4) requires a longer runtime because Augur must be run twice (once for each comparison of interest), after which empirical null distributions must be calculated through two rounds of permutation. Increasing the number of cores, using the `n_threads` argument, will decrease the runtime. Conversely, Augur runtime scales linearly with the number of cell types in the dataset, meaning the runtime will increase in datasets with more cell types. Notably, whereas the computational requirements of Augur itself can be estimated on the basis of the number of cell types and the number of threads over which the analysis is parallelized, the amount of time required for manual cell-type annotation is much more variable, and will depend on both the biological system of interest and user expertise.

A summary of the timing required to work through each of the five case studies is provided below.

**Case study 1: interferon-stimulated PBMCs (cell-type annotation and basic cell-type prioritization): ~45 min**

Steps 1–7, preprocessing and cell-type annotation: ~20 min
Steps 8–9, cell-type prioritization: ~15 min
Steps 10–11, interpret and visualize results: ~10 min

**Case study 2: walking after paralysis with spinal cord neurostimulation (RNA velocity): ~1 h 30 min**

Steps 13–16, RNA velocity calculation: ~30 min
Step 17, cell-type prioritization: ~1 h

**Case study 3: prefrontal cortex in cocaine withdrawal (multi-class classification): ~2 h**

Steps 18–20, pairwise cell-type prioritization: ~1 h 30 min
Steps 21–22, multiclass cell-type prioritization: ~30 min

**Case study 4: sex-specific differences in mouse parenting (differential prioritization): ~9 h 20 min**

Steps 23–26, cell-type prioritization in male and female mice: ~1 h 20 min
Steps 27–30, differential prioritization: ~8 h

**Case study 5: cell-type prioritization with batch effects: ~35 min**

Steps 31–35, cell-type prioritization with a confounding batch effect: ~15 min
Step 36, batch effect correction and cell-type prioritization: ~20 min

## Anticipated results

The main function in Augur, `calculate_auc`, returns an R `list` object containing the following items:

1 `X`: the numeric matrix, data frame or sparse matrix that was provided as input, containing gene expression values for each cell in the dataset
2 `y`: the vector of sample labels that the classifiers were trained to predict
3 `cell_types`: the vector of cell-type labels for each cell type in `X`
4 `parameters`: the complete set of parameters that were used in this execution of the function
5 `results`: the complete set of evaluation metrics calculated in each fold of cross-validation, for each subsample of cells
6 `feature_importance`: the importance of each feature in each fold of cross-validation
7 `AUC`: a summary of cell-type prioritization, with all cell types in the dataset ranked by their mean AUC across all subsamples and folds; for continuous sample labels, this is replaced by a `CCC` item that ranks cell types based on their mean CCC

These results can subsequently be visualized using a series of plotting functions implemented in the Augur R package, or manipulated programmatically. Here, we review the anticipated results from case studies 1–5.

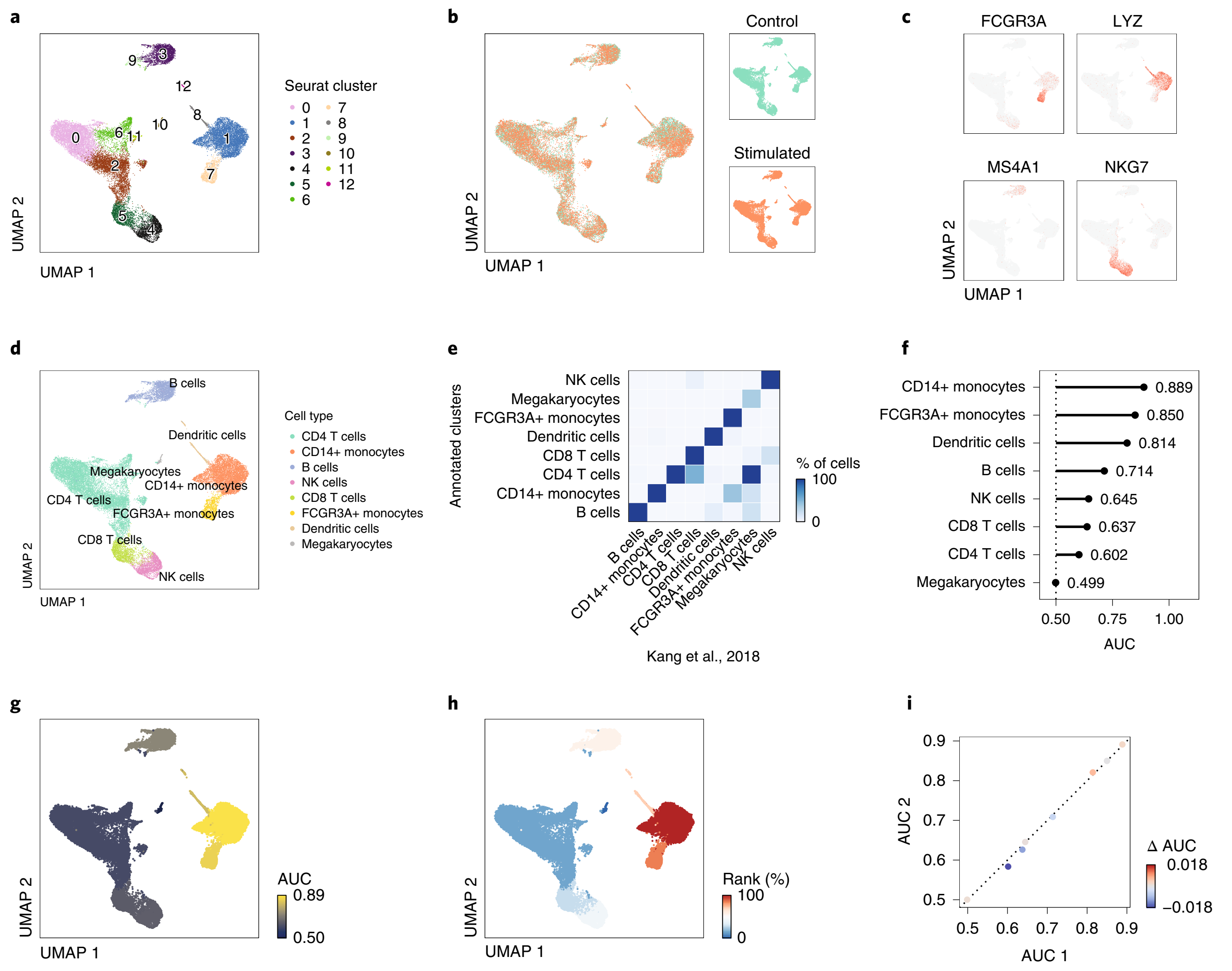


**Fig. 3 | Basic workflow for cell-type prioritization (case study 1). a**, UMAP projection of the Kang et al., 2018 dataset, colored by Seurat clusters. **b**, As in **a** but colored by experimental condition. **c**, As in **a** but colored by expression of four exemplary cell-type marker genes. **d**, As in **a** but colored by manually annotated cell type. **e**, Confusion matrix comparing cell-type annotations assigned by the authors of the original study to those reassigned here by preprocessing and clustering of the raw count matrix. **f**, Lollipop plot of cell-type prioritizations in the Kang et al., 2018 dataset. **g**, UMAP projection of the Kang et al., 2018 dataset, colored by the AUC of each cell type, as calculated by Augur. **h**, As in **g** but colored by the relative rank of each cell type, on a scale from 0% to 100%. **i**, Comparison of Augur results obtained using 50 subsamples (*x*-axis) and 100 subsamples (*y*-axis).

## Case study 1: interferon-stimulated PBMCs (cell-type annotation and basic cell-type prioritization)

We begin by reviewing the anticipated results from a typical Augur workflow, beginning from a read count matrix. A basic preprocessing workflow, including normalization, data integration, HVG selection, dimensionality reduction and graph-based clustering, provides an initial picture of the transcriptionally defined clusters of cells within this dataset. To gain a deeper appreciation for these clusters, the dataset can be visualized in two dimensions using the UMAP dimensionality reduction algorithm (Fig. 3a). Dimensionality reduction also confirms that the data integration procedure has removed any overt batch effects, with no obvious differences between cells from either of the two experimental conditions in the low-dimensional representation (Fig. 3b).

After preprocessing the dataset, each of the transcriptionally defined clusters must be manually annotated to associate them with a cell type. Manual inspection of marker genes identified by unbiased differential expression analysis provides one basis to annotate each cluster. It may also be useful to overlay the expression of some marker genes, either those suggested by unbiased differential expression analysis or known marker genes for the system of interest, directly onto the UMAP plot (Fig. 3c). After completing manual annotation of each cluster, and merging clusters that correspond to the same cell types, a final set of manually annotated cell types is obtained for input to Augur (Fig. 3d). Cross-referencing the cell types defined by this workflow (Steps 1–7) to those defined by the authors of the original publication[14] confirms the concordance between the two sets of annotations (Fig. 3e).

Applying Augur to the cell-type labels obtained from Steps 1–7 yields a list of cell types, ranked by their separability between conditions (i.e., the AUC). These results can be visualized in a number of different ways, the simplest of which is as a 'lollipop' plot (Fig. 3f). The AUCs for each cell type can also be overlaid onto the UMAP visualization, to provide a global overview of the perturbation response (Fig. 3g). In cases where the relative ranks of the AUCs assigned to each cell type are more meaningful than their absolute values, the AUCs can be converted with the rank of each cell type on the UMAP plot (Fig. 3h).

Augur optionally takes a number of user-specified parameters as arguments (Table 1). In general, Augur is robust to the specific values of these parameters (with some exceptions; Table 3). However, in cases where a user may wish to compare cell-type prioritizations obtained using different sets of parameters, this can be achieved using the `plot_scatterplot` function to compare two sets of Augur outputs. Figure 3i demonstrates the effect of increasing the number of subsamples from 50 to 100, confirming that 50 subsamples provide sufficient information for cell-type prioritizations to converge.

## Case study 2: walking after paralysis with spinal cord neurostimulation (RNA velocity)

Next, we review the application of Augur to estimates of RNA velocity, to dissect the response to acute perturbations on the timescale of transcription. Once the RNA velocity has been estimated from separate matrices of intronic and exonic read counts, Augur can be applied to the RNA velocity matrix in much the same way as to an ordinary gene expression matrix, with the exception that feature selection is disabled. In this dataset, we applied Augur to estimates of RNA velocity calculated from snRNA-seq of the lumbar spinal cord in mice with spinal cord injury. We identified a total of 38 neuronal subtypes (Fig. 4a; note that a population of contaminating dorsal root ganglion neurons was removed prior to visualization). Augur identified interneurons with the molecular profiles of V2a and V1/V2b neurons as involved in the recovery of walking after paralysis (Fig. 4b,c). Overlaying these prioritizations onto the UMAP plot highlights the localization of these prioritized ventral interneurons within the low-dimensional representation (Fig. 4b). Alternatively, a lollipop plot permits the visualization of the complete ranked list of neuronal subtypes (Fig. 4c).

Since neuronal subtypes can be defined at multiple biologically relevant resolutions, we also illustrate the use of clustering trees, in conjunction with Augur, to more deeply understand the perturbation-responsive neuron subtypes (Box 4). Here, the dataset was clustered using Seurat at seven different biological resolutions, and a clustering tree was used to establish and visualize the relationships between clusters at each resolution. Manual annotation of each resolution yielded a complete picture of the hierarchical organization of neuron subtypes in the mouse spinal cord (Fig. 4d). Relationships between adjacent resolutions were biologically coherent (for instance, dorsal inhibitory neurons split into Cdh3, Pnoc and Rorb subtypes), confirming the validity of the cell-type annotations used in cell-type prioritization. Overlaying Augur results onto the clustering tree provided further validation of cell-type annotations, with no abrupt transitions in AUC observed between adjacent resolutions (Fig. 4e). Finally, whereas spurious prioritizations may manifest in the form of an unexpectedly high AUC for a single, isolated node on the clustering tree, the clustering tree shown in Fig. 4e illustrates a coherent gradient of perturbation response over multiple biological resolutions, with Augur initially prioritizing ventral interneurons, and then progressively more specific subtypes of ventral interneurons.

## Case study 3: prefrontal cortex in cocaine withdrawal (multiclass classification)

Augur can perform cell-type prioritization in single-cell datasets with a wide variety of experimental designs. For instance, Augur can analyze data from three or more conditions simultaneously by multiclass classification. Here, we illustrate several potential approaches to cell-type prioritization in a single dataset with three experimental conditions, containing 12,936 cells grouped in eight types (Fig. 5a). These neurons were obtained from the prefrontal cortex of mice exposed to a cocaine self-addiction paradigm and subjected to either 48 h or 15 d of withdrawal, or maintained on cocaine. Data from all three experimental conditions can be analyzed jointly, by asking Augur to predict which of the three conditions each cell was obtained from (Fig. 5b,c). Alternatively, it may be more relevant to carry out all possible binary comparisons—that is, comparing cells at

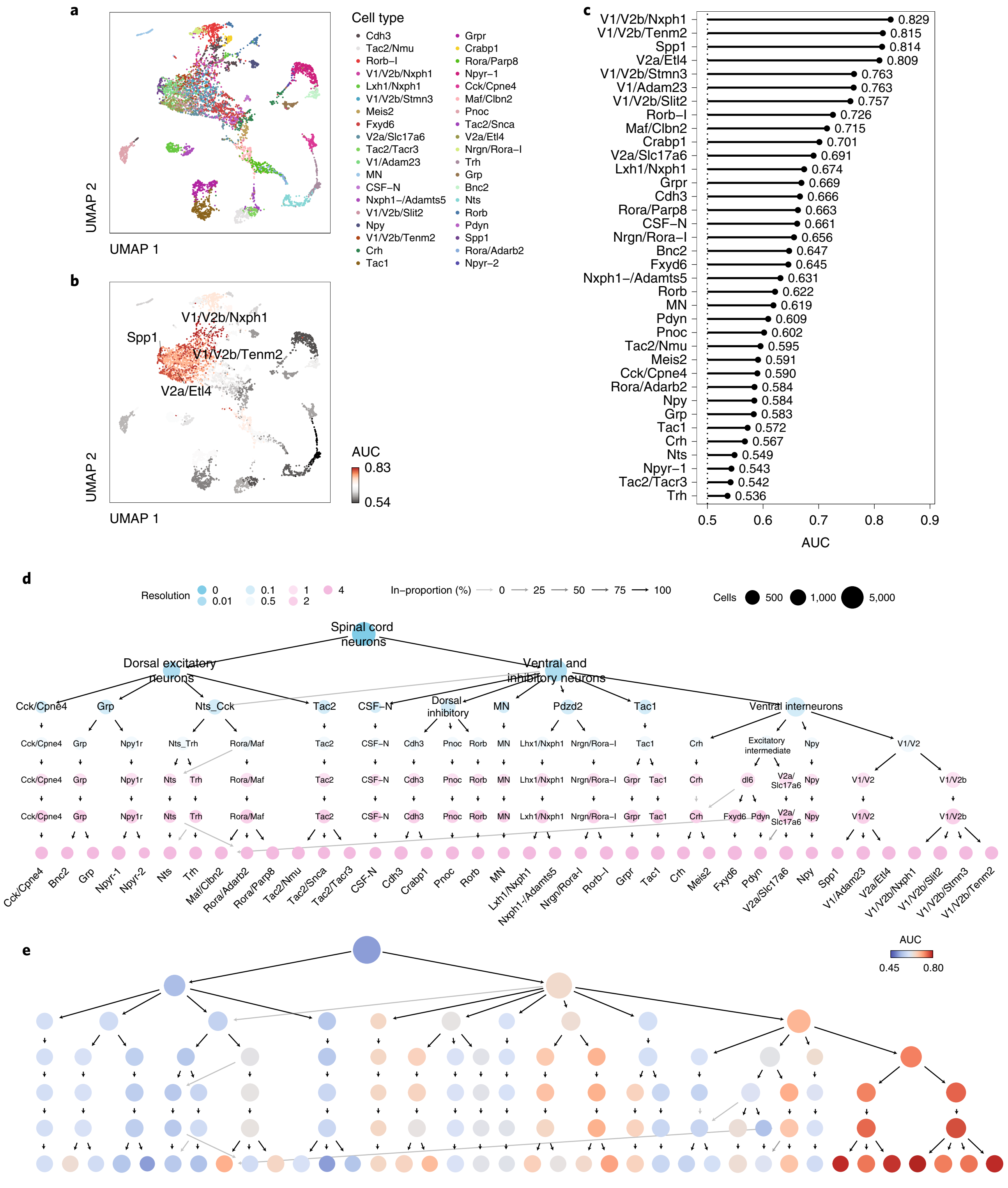


**Fig. 4 | Cell-type prioritization in RNA velocity (case study 2) and on clustering trees. a**, UMAP projection of the Skinnider et al., 2020 dataset, colored by neuronal subtype. **b**, As in **a** but colored by the AUC of each cell type, as calculated by Augur from the RNA velocity. **c**, Lollipop plot of cell-type prioritizations in the Skinnider et al. dataset. **d**, Clustering tree of the Skinnider et al. dataset at seven different resolutions, with nodes colored by the value of the resolution parameter in Seurat clustering. **e**, As in **d** but with nodes colored by the AUC as calculated by Augur.

maintenance versus 48 h, maintenance versus 15 d, and 48 h versus 15 d—or a subset of them. The results from a comparison of cells after maintenance and 48 h of cocaine withdrawal, for instance, are shown in Fig. 5d,e.

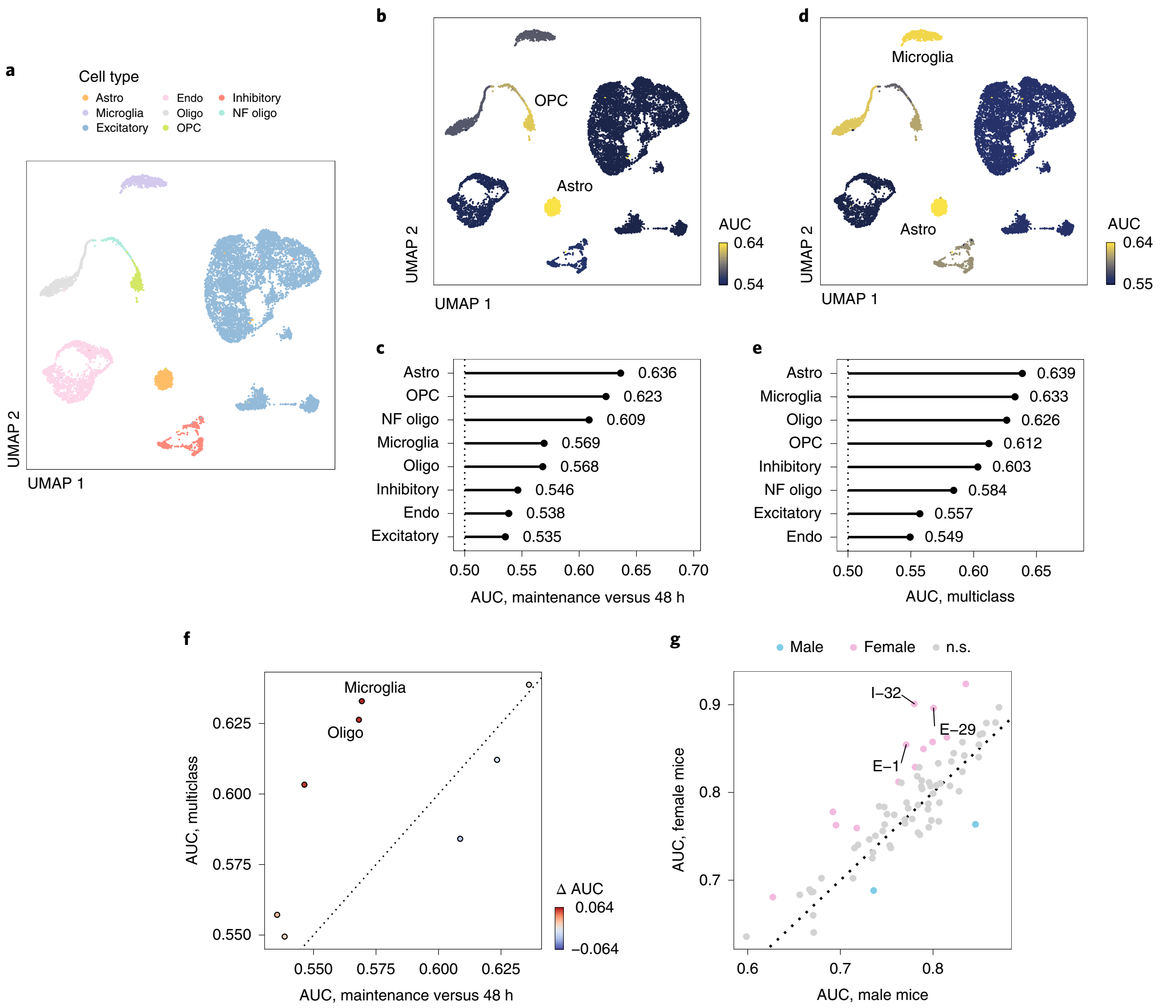


**Fig. 5 | Cell-type prioritization across multiple experimental conditions and differential prioritization (case studies 3 and 4). a**, UMAP projection of the Bhattacherjee et al., 2019 dataset, colored by annotated cell type. **b**, As in **a** but colored by the AUC in a multiclass comparison of all three experimental conditions. **c**, Lollipop plot of the cell-type prioritizations in a multiclass comparison of all three experimental conditions. **d**, As in **b** but colored by the AUC in a binary comparison of maintenance versus 48 h of withdrawal. **e**, As in **c** but in a binary comparison of maintenance versus 48 h of withdrawal. **f**, Comparison of cell-type prioritizations obtained in binary classification (maintenance versu 48 h), *x*-axis, and multiclass classification, *y*-axis. **g**, Differential prioritization of neuron subtypes activated during parenting in male versus female mice, in the Moffitt et al., 2018 dataset.

In some cases, it may not be immediately obvious whether a multiclass or binary classification framework is a more appropriate way to analyze a particular dataset. To facilitate this decision, Augur includes functionality to directly compare cell-type prioritizations from multiclass and binary frameworks, as shown in Fig. 5f. This function was illustrated above for the comparison of cell-type prioritizations obtained using different Augur parameters (Fig. 3i), but can also be used to compare cell-type prioritizations under two different designs. For instance, Fig. 5f shows that microglia are ranked more highly in a multiclass comparison. Whereas a binary classification framework entails contrasting cells from mice exposed to long and short periods of withdrawal with cells from the control population, or discarding the control population by comparing long and short periods of withdrawal, the multiclass comparison integrates information from all three experimental groups. In doing so, the multiclass approach allows the user to formulate hypotheses about what cell types might display dynamic transcriptional perturbations over the entire period of withdrawal. For instance, the prioritization of microglia in the multiclass comparison is consistent with previous literature emphasizing the dynamic responses of microglia to cocaine addiction and withdrawal; one such report implicated microglia in modulating synaptic strength and sensitization during the withdrawal period[73]. Of note, a role for microglia was not identified in the original publication, which focused primarily on the responses of individual neuron subtypes to addiction and withdrawal[62]. Collectively, this example demonstrates how cell-type prioritization across multiple experimental conditions can suggest hypotheses for follow-up experiments.

### Case study 4: sex-specific differences in mouse parenting (differential prioritization)

Next, we illustrate the use of Augur to identify cell types that exhibit different responses to two distinct perturbations, or to the same perturbation in different backgrounds, using a test for differential prioritization. We apply this test to a MERFISH dataset, comprising 302,895 neurons grouped into 69 subtypes, to identify neurons differentially prioritized during parenting in male and female mice. Initially, Augur is applied to generate separate cell-type prioritizations in male and female mice. Simply comparing the AUCs calculated for each cell type, or their relative ranks, provides an initial basis to assess differences in neuronal perturbation between mice of each sex. To gain confidence that these differences are statistically significant, an empirical null distribution of the ΔAUC between male and female mice is computed for each cell type by separately permuting each dataset. Finally, these differences, and their statistical significance, can be visualized in a differential prioritization plot (Fig. 5g). Several of the differentially prioritized neuronal subtypes also exhibit transcriptional differences between male and female naive mice—for instance, the I-32 cluster, which is enriched for aromatase expression, and expresses multiple sex steroid hormone receptors[35].

Notably, these results differ in several ways from those presented in Fig. 8 of the original publication describing this MERFISH dataset[35]. Two important factors likely account for these differences. Importantly, whereas the authors of the original study evaluated only expression of the IEG Fos, Augur considers all 155 genes that were imaged using MERFISH to prioritize neurons perturbed during parenting in mice of each sex. Considering many genes, rather than just a single IEG is particularly advantageous in sparse, droplet-based scRNA-seq datasets, where in our experience, it is uncommon to quantify lowly expressed IEGs such as Fos with more than one or two reads per cell. Moreover, in the MERFISH dataset, the Fos gene itself was omitted from the Augur analysis, because it was not measured in control mice, and therefore could not be compared between parenting and naive animals. Second, the analysis carried out by the authors of the original study tested a different null hypothesis to that evaluated by Augur. In the original study, the authors binarized Fos expression, dividing cells into Fos-positive and Fos-negative groups, then used a binomial test to identify cell types enriched for Fos-positive cells, as compared with all other cell types. They performed this analysis in male and female parents separately, then compared the results qualitatively. In contrast, the approach to differential prioritization implemented within Augur directly tests the null hypothesis that the prioritized cell types exhibit statistically significant differences in transcriptional separability, within the 155-dimensional space of gene expression, in male versus female mice during parenting. Finally, notwithstanding these discrepancies, it is important to emphasize that the prioritizations obtained using Augur reflect hypotheses suggested by the MERFISH data, and would await experimental confirmation in vivo.

### Case study 5: simulated data (batch effects)

Separation within cell types can arise not only from the cell-intrinsic response to a perturbation, but also a number of confounding technical factors. This final case study explores the impact of batch effects on cell-type prioritization using synthetic data, where the ground truth is known a priori. The dataset contains simulated scRNA-seq profiles for 5,000 cells, which collectively constitute a tissue with five cell types (Fig. 6a). These cell types display a graded response to perturbation, with the least responsive cell type displaying hardly any response at all, and the most responsive cell type undergoing a profound response to the simulated perturbation (Fig. 6b,c). The cell types are also present at uneven frequencies in the tissue, with a Gini coefficient characteristic of real scRNA-seq data. Moreover, the cells were sequenced in two different batches, and cells from stimulated or unstimulated tissues are each more likely to be in one batch than the other (Fig. 6d). Finally, one cell type (cell type D) has undergone a much more profound batch effect than the other four, perhaps because cells of this type were more sensitive to the dissociation protocol (Fig. 6e). Collectively, the parameters of this simulation reflect the fairly extreme nature of the batch effects that are necessary to undermine

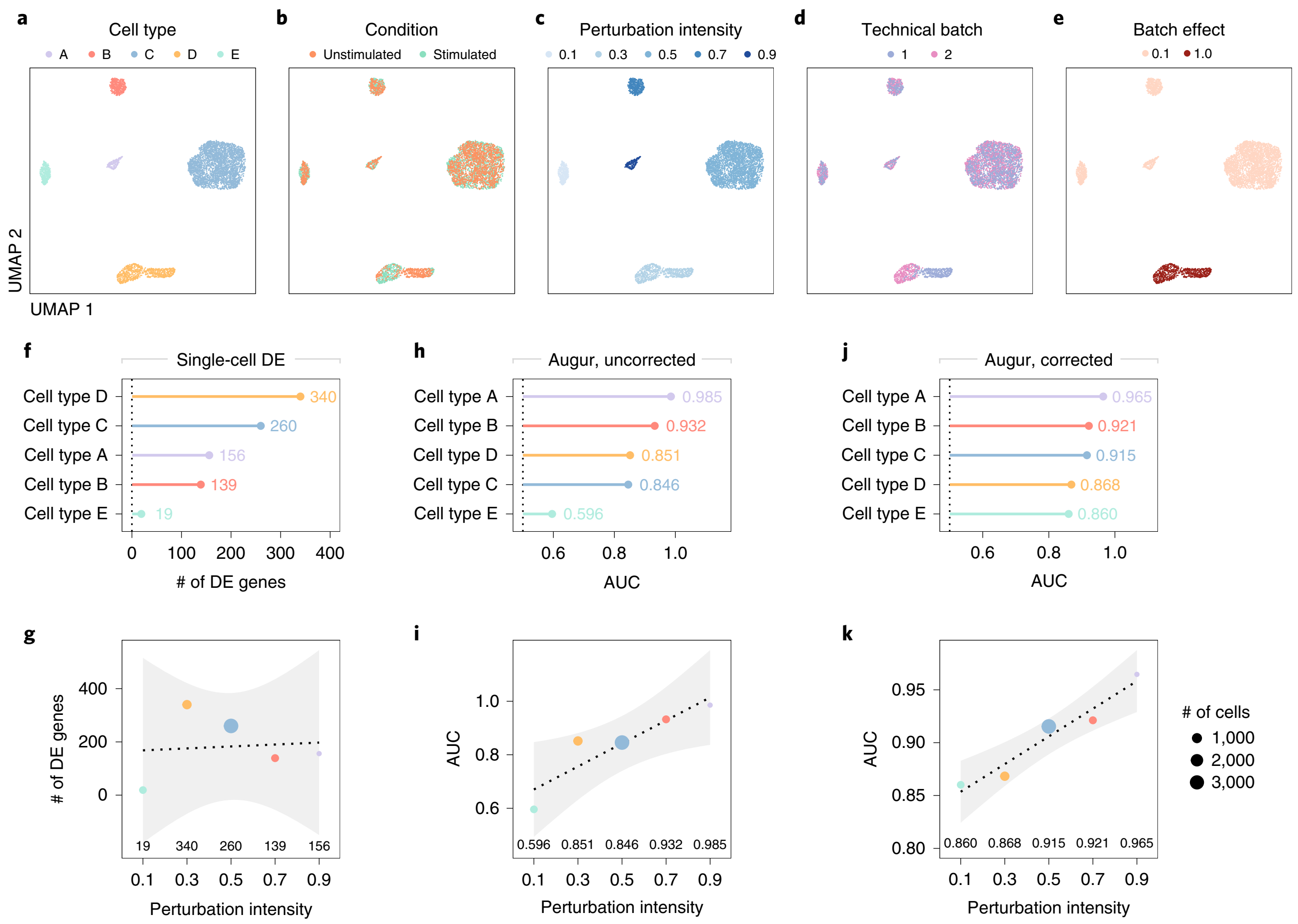


**Fig. 6 | Cell-type prioritization in datasets confounded by batch effects. a–e**, UMAP projections of the simulated dataset discussed in case study 5, colored by cell type (**a**), experimental condition (**b**), simulated perturbation intensity (**c**), technical batch (**d**) and batch effect magnitude (**e**). **f**, Lollipop plot of cell types ordered by the simulated perturbation intensity, showing the number of DE genes identified in each cell type by an exemplary single-cell DE method, the Wilcoxon rank-sum test. **g**, Correlation between the simulated perturbation intensity and the number of DE genes in each cell type. Point size shows the number of cells per type. Inset text along the *x*-axis shows the number of DE genes identified in each cell type at 5% false discovery rate. Dotted lines and shaded areas show linear regressions and 95% confidence intervals, respectively. **h**, Lollipop plot of Augur cell-type prioritizations with no batch correction. **i**, Relationship between the simulated perturbation intensity and the AUC assigned by Augur in the uncorrected data. Point size shows the number of cells per type. Inset text along the *x*-axis shows the AUC. Dotted lines and shaded areas show linear regressions and 95% confidence intervals, respectively. **j**, Lollipop plot of Augur cell-type prioritizations after batch effect correction by matching mutual nearest neighbors. **k**, Relationship between the simulated perturbation intensity and the AUC assigned by Augur in the data after mutual nearest neighbors correction. Point size shows the number of cells per type. Inset text along the *x*-axis shows the AUC. Dotted lines and shaded areas show linear regressions and 95% confidence intervals, respectively.

accurate cell-type prioritization in Augur (see also Extended Data Fig. 10 of the original Augur publication[29] for further discussion of this point).

We first take advantage of the 'ground-truth' nature of this dataset to illustrate one of the major deficiencies of cell-type prioritization based on differential expression analysis. Approaches that simply count the number of genes that are DE between perturbed and unperturbed cells are confounded by the relative abundances of each cell type. As a result, the prioritizations obtained from a representative differential expression method, the Wilcoxon rank-sum test, exhibit little correlation to the simulation ground truth (Fig. 6f,g). In contrast, applying Augur to the uncorrected data yields prioritizations that are much more in line with the simulated perturbations. The exception is cell type D, which also underwent a profound batch effect. Augur cannot distinguish the effect of the perturbation from that of the technical batch effect, and as a result, this cell type is prioritized above others that were more responsive to the perturbation itself (Fig. 6h,i). Matching between mutual nearest neighbors corrects the batch effect, and restores the expected prioritizations (Fig. 6j,k).

## Reporting Summary

Further information on research design is available in the Nature Research Reporting Summary linked to this article.

## Data availability

The datasets discussed in this protocol are available from Zenodo (https://doi.org/10.5281/zenodo.4473025). Accession codes for these datasets are provided in Table 2.

## Code availability

Augur is a freely available software package written in the R programming language and released under the MIT license. Source code can be obtained from https://github.com/neurorestore/Augur. We provide support through the GitHub issues tracker at https://github.com/neurorestore/Augur/issues.

## References

1. Tang, F. et al. mRNA-Seq whole-transcriptome analysis of a single cell. *Nat. Methods* **6**, 377–382 (2009).
2. Svensson, V., Vento-Tormo, R. & Teichmann, S. A. Exponential scaling of single-cell RNA-seq in the past decade. *Nat. Protoc.* **13**, 599–604 (2018).
3. Cao, J. et al. The single-cell transcriptional landscape of mammalian organogenesis. *Nature* **566**, 496–502 (2019).
4. Tabula Muris Consortium. et al. Single-cell transcriptomics of 20 mouse organs creates a Tabula Muris. *Nature* **562**, 367–372 (2018).
5. Han, X. et al. Mapping the mouse cell atlas by Microwell-Seq. *Cell* **173**, 1307 (2018).
6. Han, X. et al. Construction of a human cell landscape at single-cell level. *Nature* **581**, 303–309 (2020).
7. Plass, M. et al. Cell type atlas and lineage tree of a whole complex animal by single-cell transcriptomics. *Science* **360**, (2018).
8. Cao, J. et al. Comprehensive single-cell transcriptional profiling of a multicellular organism. *Science* **357**, 661–667 (2017).
9. Regev, A. et al. The Human Cell Atlas. *eLife* **6**, e27041 (2017).
10. Vieira Braga, F. A. et al. A cellular census of human lungs identifies novel cell states in health and in asthma. *Nat. Med.* **25**, 1153–1163 (2019).
11. Mathys, H. et al. Single-cell transcriptomic analysis of Alzheimer's disease. *Nature* **570**, 332–337 (2019).
12. Grubman, A. et al. A single-cell atlas of entorhinal cortex from individuals with Alzheimer's disease reveals cell-type-specific gene expression regulation. *Nat. Neurosci.* **22**, 2087–2097 (2019).
13. Smillie, C. S. et al. Intra- and inter-cellular rewiring of the human colon during ulcerative colitis. *Cell* **178**, 714–730.e22 (2019).
14. Kang, H. M. et al. Multiplexed droplet single-cell RNA-sequencing using natural genetic variation. *Nat. Biotechnol.* **36**, 89–94 (2018).
15. Wagner, D. E. et al. Single-cell mapping of gene expression landscapes and lineage in the zebrafish embryo. *Science* **360**, 981–987 (2018).
16. Tabula Muris Consortium. A single-cell transcriptomic atlas characterizes ageing tissues in the mouse. *Nature* **583**, 590–595 (2020).
17. Svensson, V., da Veiga Beltrame, E. & Pachter, L. A curated database reveals trends in single-cell transcriptomics. *Database* https://doi.org/10.1093/database/baaa073 (2020).
18. Soneson, C. & Robinson, M. D. Bias, robustness and scalability in single-cell differential expression analysis. *Nat. Methods* **15**, 255–261 (2018).
19. Crowell, H. L. et al. *muscat* detects subpopulation-specific state transitions from multi-sample multi-condition single-cell transcriptomics data. *Nat. Comun.* **11**, 6077 (2020).
20. Zimmerman, K. D., Espeland, M. A. & Langefeld, C. D. A practical solution to pseudoreplication bias in single-cell studies. *Nat. Commun.* **12**, 738 (2021).
21. Rossi, M. A. et al. Obesity remodels activity and transcriptional state of a lateral hypothalamic brake on feeding. *Science* **364**, 1271–1274 (2019).
22. Hrvatin, S. et al. Single-cell analysis of experience-dependent transcriptomic states in the mouse visual cortex. *Nat. Neurosci.* **21**, 120–129 (2018).
23. Hashikawa, Y. et al. Transcriptional and spatial resolution of cell types in the mammalian habenula. *Neuron* **106**, 743–758.e5 (2020).
24. Sathyamurthy, A. et al. Massively parallel single nucleus transcriptional profiling defines spinal cord neurons and their activity during behavior. *Cell Rep* **22**, 2216–2225 (2018).
25. Hrvatin, S. et al. Neurons that regulate mouse torpor. *Nature* **583**, 115–121 (2020).
26. Schirmer, L. et al. Neuronal vulnerability and multilineage diversity in multiple sclerosis. *Nature* **573**, 75–82 (2019).
27. Avey, D. et al. Single-cell RNA-seq uncovers a robust transcriptional response to morphine by glia. *Cell Rep* **24**, 3619–3629.e4 (2018).

28. Kotliarov, Y. et al. Broad immune activation underlies shared set point signatures for vaccine responsiveness in healthy individuals and disease activity in patients with lupus. *Nat. Med.* **26**, 618–629 (2020).
29. Skinnider, M. A. et al. Cell type prioritization in single-cell data. *Nat. Biotechnol.* https://doi.org/10.1038/s41587-020-0605-1 (2020).
30. La Manno, G. et al. RNA velocity of single cells. *Nature* **560**, 494–498 (2018).
31. Wagner, F. B. et al. Targeted neurotechnology restores walking in humans with spinal cord injury. *Nature* **563**, 65–71 (2018).
32. Formento, E. et al. Electrical spinal cord stimulation must preserve proprioception to enable locomotion in humans with spinal cord injury. *Nat. Neurosci.* **21**, 1728–1741 (2018).
33. Hagai, T. et al. Gene expression variability across cells and species shapes innate immunity. *Nature* **563**, 197–202 (2018).
34. Wang, X. et al. Three-dimensional intact-tissue sequencing of single-cell transcriptional states. *Science* **361**, eaat5691 (2018).
35. Moffitt, J. R. et al. Molecular, spatial, and functional single-cell profiling of the hypothalamic preoptic region. *Science* **362**, eaau5324 (2018).
36. Lareau, C. A. et al. Droplet-based combinatorial indexing for massive-scale single-cell chromatin accessibility. *Nat. Biotechnol.* **37**, 916–924 (2019).
37. Lin, L. I. A concordance correlation coefficient to evaluate reproducibility. *Biometrics* **45**, 255–268 (1989).
38. Bentsen, M. A. et al. Transcriptomic analysis links diverse hypothalamic cell types to fibroblast growth factor 1-induced sustained diabetes remission. *Nat. Commun.* **11**, 4458 (2020).
39. Kim, D.-W. et al. Multimodal analysis of cell types in a hypothalamic node controlling social behavior. *Cell* **179**, 713–728.e17 (2019).
40. Wu, Y. E., Pan, L., Zuo, Y., Li, X. & Hong, W. Detecting activated cell populations using single-cell RNA-seq. *Neuron* **96**, 313–329.e6 (2017).
41. Skinnider, M. A., Squair, J. W. & Foster, L. J. Evaluating measures of association for single-cell transcriptomics. *Nat. Methods* **16**, 381–386 (2019).
42. Clevers, H. et al. What is your conceptual definition of "cell type" in the context of a mature organism? *Cell Syst.* **4**, 255–259 (2017).
43. Trapnell, C. Defining cell types and states with single-cell genomics. *Genome Res* **25**, 1491–1498 (2015).
44. Zappia, L. & Oshlack, A. Clustering trees: a visualization for evaluating clusterings at multiple resolutions. *Gigascience* **7**, giy083 (2018).
45. Stuart, T. et al. Comprehensive integration of single-cell data. *Cell* **177**, 1888–1902.e21 (2019).
46. Amezquita, R. A. et al. Orchestrating single-cell analysis with Bioconductor. *Nat. Methods* **17**, 137–145 (2020).
47. Luecken, M. D. & Theis, F. J. Current best practices in single-cell RNA-seq analysis: a tutorial. *Mol. Syst. Biol.* **15**, e8746 (2019).
48. Wolf, F. A., Angerer, P. & Theis, F. J. SCANPY: large-scale single-cell gene expression data analysis. *Genome Biol.* **19**, 15 (2018).
49. Pliner, H. A., Shendure, J. & Trapnell, C. Supervised classification enables rapid annotation of cell atlases. *Nat. Methods* **16**, 983–986 (2019).
50. Zhang, A. W. et al. Probabilistic cell-type assignment of single-cell RNA-seq for tumor microenvironment profiling. *Nat. Methods* **16**, 1007–1015 (2019).
51. Abdelaal, T. et al. A comparison of automatic cell identification methods for single-cell RNA sequencing data. *Genome Biol.* **20**, 194 (2019).
52. Zheng, G. X. Y. et al. Massively parallel digital transcriptional profiling of single cells. *Nat. Commun.* **8**, 14049 (2017).
53. Petukhov, V. et al. dropEst: pipeline for accurate estimation of molecular counts in droplet-based single-cell RNA-seq experiments. *Genome Biol.* **19**, 78 (2018).
54. Melsted, P. et al. Modular, efficient and constant-memory single-cell RNA-seq preprocessing. *Nat. Biotechnol.* https://doi.org/10.1038/s41587-021-00870-2 (2021).
55. Srivastava, A., Malik, L., Smith, T., Sudbery, I. & Patro, R. Alevin efficiently estimates accurate gene abundances from dscRNA-seq data. *Genome Biol.* **20**, 65 (2019).
56. Dobin, A. et al. STAR: ultrafast universal RNA-seq aligner. *Bioinformatics* **29**, 15–21 (2013).
57. Chen, H. et al. Assessment of computational methods for the analysis of single-cell ATAC-seq data. *Genome Biol.* **20**, 241 (2019).
58. Ilicic, T. et al. Classification of low quality cells from single-cell RNA-seq data. *Genome Biol.* **17**, 29 (2016).
59. Lun, A. T. L. et al. EmptyDrops: distinguishing cells from empty droplets in droplet-based single-cell RNA sequencing data. *Genome Biol.* **20**, 63 (2019).
60. Wolock, S. L., Lopez, R. & Klein, A. M. Scrublet: computational identification of cell doublets in single-cell transcriptomic data. *Cell Syst.* **8**, 281–291.e9 (2019).
61. McGinnis, C. S., Murrow, L. M. & Gartner, Z. J. DoubletFinder: doublet detection in single-cell RNA sequencing data using artificial nearest neighbors. *Cell Syst.* **8**, 329–337.e4 (2019).
62. Bhattacherjee, A. et al. Cell type-specific transcriptional programs in mouse prefrontal cortex during adolescence and addiction. *Nat. Commun.* **10**, 4169 (2019).
63. McInnes, L., Healy, J. & Melville, J. UMAP: uniform manifold approximation and projection for dimension reduction. Preprint at https://arxiv.org/abs/1802.03426 (2018).

64. Becht, E. et al. Dimensionality reduction for visualizing single-cell data using UMAP. *Nat. Biotechnol.* **37**, 38–44 (2018).
65. Korsunsky, I. et al. Fast, sensitive and accurate integration of single-cell data with Harmony. *Nat. Methods* **16**, 1289–1296 (2019).
66. Hie, B., Bryson, B. & Berger, B. Efficient integration of heterogeneous single-cell transcriptomes using Scanorama. *Nat. Biotechnol.* **37**, 685–691 (2019).
67. Welch, J. D. et al. Single-cell multi-omic integration compares and contrasts features of brain cell identity. *Cell* **177**, 1873–1887.e17 (2019).
68. Polański, K. et al. BBKNN: fast batch alignment of single cell transcriptomes. *Bioinformatics* **36**, 964–965 (2020).
69. Tran, H. T. N. et al. A benchmark of batch-effect correction methods for single-cell RNA sequencing data. *Genome Biol.* **21**, 12 (2020).
70. Haghverdi, L., Lun, A. T. L., Morgan, M. D. & Marioni, J. C. Batch effects in single-cell RNA-sequencing data are corrected by matching mutual nearest neighbors. *Nat. Biotechnol.* **36**, 421–427 (2018).
71. Stoeckius, M. et al. Simultaneous epitope and transcriptome measurement in single cells. *Nat. Methods* **14**, 865–868 (2017).
72. Büttner, M., Miao, Z., Wolf, F. A., Teichmann, S. A. & Theis, F. J. A test metric for assessing single-cell RNA-seq batch correction. *Nat. Methods* **16**, 43–49 (2019).
73. Lewitus, G. M. et al. Microglial TNF-α suppresses cocaine-induced plasticity and behavioral sensitization. *Neuron* **90**, 483–491 (2016).
74. Zappia, L., Phipson, B. & Oshlack, A. Splatter: simulation of single-cell RNA sequencing data. *Genome Biol.* **18**, 174 (2017).
75. McDavid, A. et al. Data exploration, quality control and testing in single-cell qPCR-based gene expression experiments. *Bioinformatics* **29**, 461–467 (2013).
76. Ntranos, V., Yi, L., Melsted, P. & Pachter, L. A discriminative learning approach to differential expression analysis for single-cell RNA-seq. *Nat. Methods* **16**, 163–166 (2019).
77. Finak, G. et al. MAST: a flexible statistical framework for assessing transcriptional changes and characterizing heterogeneity in single-cell RNA sequencing data. *Genome Biol.* **16**, 278 (2015).
78. Erhard, F. et al. scSLAM-seq reveals core features of transcription dynamics in single cells. *Nature* **571**, 419–423 (2019).
79. Phipson, B. & Smyth, G. K. Permutation P-values should never be zero: calculating exact P-values when permutations are randomly drawn. *Stat. Appl. Genet. Mol. Biol.* **9**, 39 (2010).
80. Schwartz, G. W. et al. TooManyCells identifies and visualizes relationships of single-cell clades. *Nat. Methods* **17**, 405–413 (2020).

## Acknowledgements
M.A.S. acknowledges support from Wings for Life, the Canadian Institutes of Health Research (CIHR) (Vanier Canada Graduate Scholarship, Michael Smith Foreign Study Supplement), an Izaak Walton Killam Memorial Pre-Doctoral Fellowship, a UBC Four Year Fellowship, a Vancouver Coastal Health–CIHR–UBC MD/PhD Studentship, a Brain Canada Hubert van Tol fellowship and a BCRegMed Collaborative Research Travel Grant. J.W.S. is supported by a CIHR Banting postdoctoral fellowship and a Marie Skłodowska-Curie individual fellowship (No. 842578). Work in L.J.F.'s group is supported by Genome Canada/Genome BC (Project 264PRO). The present work was supported by a Consolidator Grant from the European Research Council (ERC-2015-CoG HOW2WALKAGAIN 682999) and the Swiss National Science Foundation (subside 310030_192558).

## Author contributions
J.W.S. and M.A.S. implemented all procedures and developed Augur. M.G. contributed to the procedures. L.J.F. and G.C. supervised the work. J.W.S., M.A.S. and G.C. wrote the manuscript. All authors edited the manuscript.

## Competing interests
G.C. is a founder and shareholder of ONWARD Medical, a company with no direct relationships with the present work.

## Additional information
**Extended data** is available for this paper at https://doi.org/10.1038/s41596-021-00561-x.

**Supplementary information** The online version contains supplementary material available at https://doi.org/10.1038/s41596-021-00561-x.

**Correspondence and requests for materials** should be addressed to J.W.S., M.A.S. or G.C.

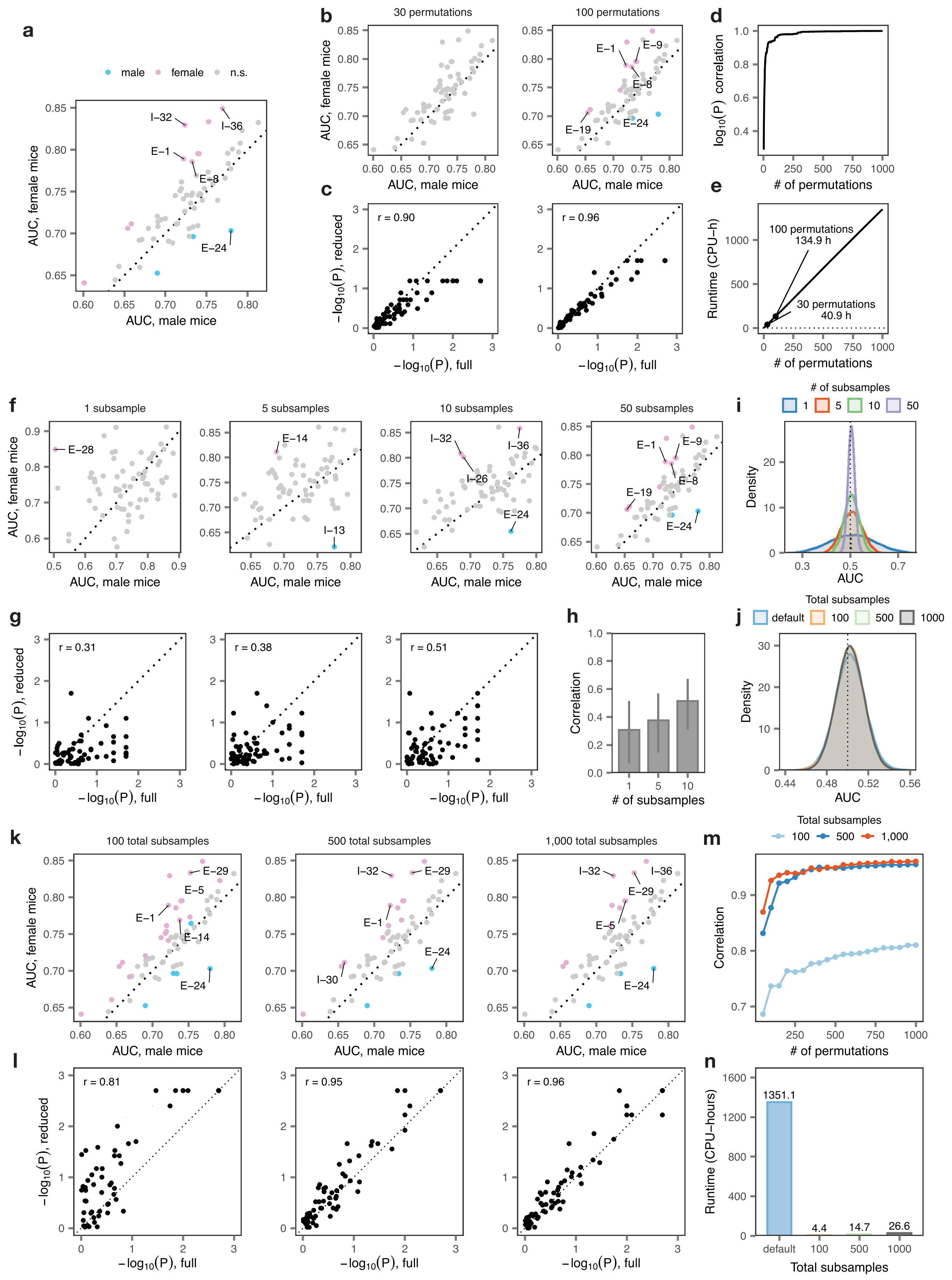
**Peer review information** *Nature Protocols* thanks Lyla Atta, Jean Fan, Brendan Miller, Joshua Welch and the other, anonymous reviewer (s) for their contribution to the peer review of this work.

Received: 14 October 2020; Accepted: 14 April 2021;
Published online: 25 June 2021

## Related links

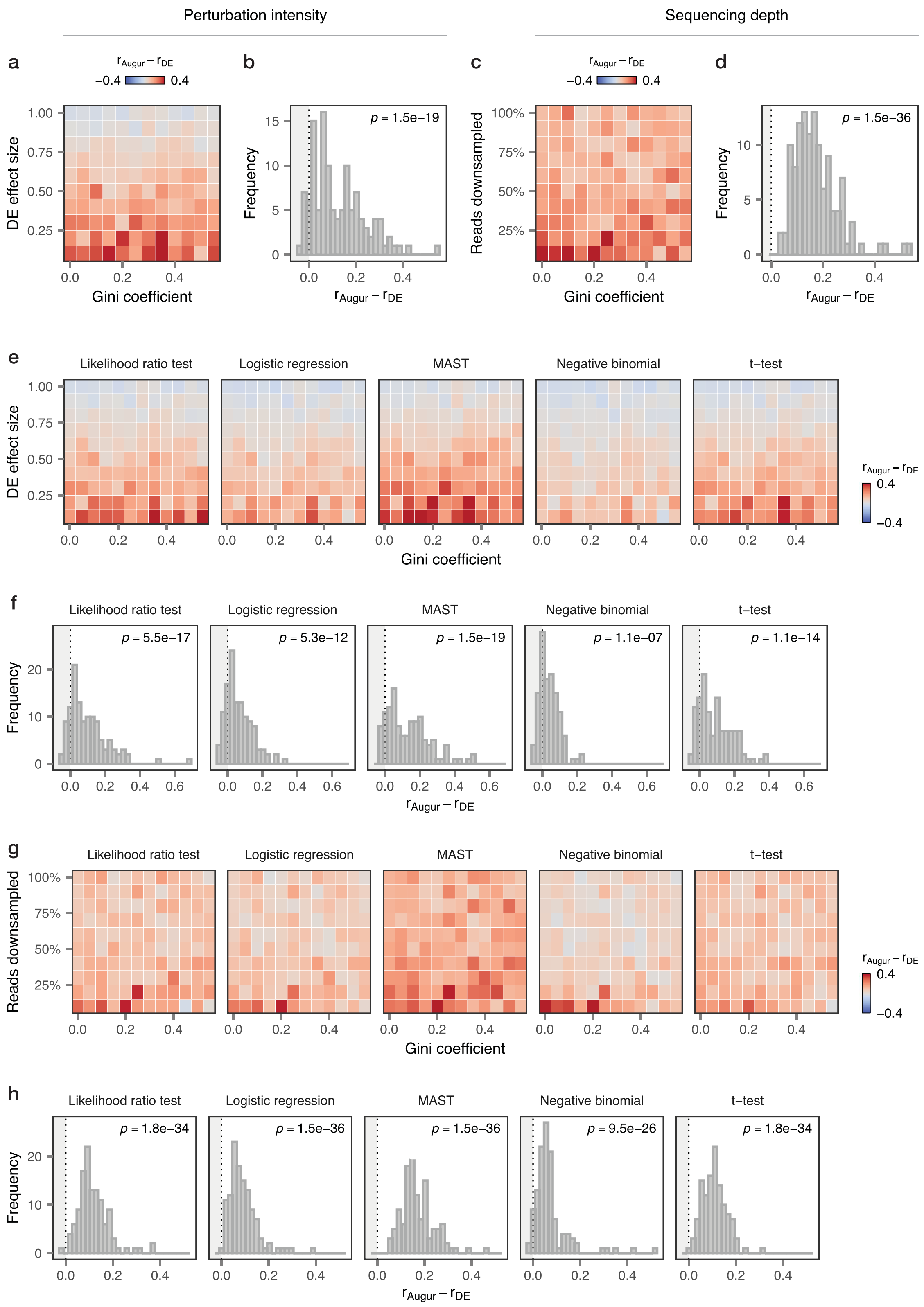
**Key reference using this protocol**
Skinnider, M. et al. *Nat. Biotechnol.* **39**, 30–34 (2021): https://doi.org/10.1038/s41587-020-0605-1

**Key data used in this protocol**
Kang, H. et al. *Nat. Biotechnol.* **36**, 89–94 (2018): https://doi.org/10.1038/nbt.4042
Bhattacherjee, A. et al. *Nat. Commun.* **10**, 4169 (2019): https://doi.org/10.1038/s41467-019-12054-3
Moffitt, J. R. et al. *Science* **362**, eaau5324 (2018): https://doi.org/10.1126/science.aau5324



**Extended Data Fig. 1 | See next page for caption.**

◀ **Extended Data Fig. 1 | Optimized workflow for differential cell-type prioritization.** Differential cell-type prioritization in the Moffitt et al., 2018[35] MERFISH dataset with variable numbers of permutations, subsamples per permutation, and total subsamples. **a**, Differential prioritization in the full dataset, with a background of 1,000 independent permutations. The top five cell types with a permutation $P$-value $< 0.05$ are shown throughout. **b**–**e**, Impact of reducing the number of permutations on differential prioritization. Differential prioritization yields stable results with the number of permutations decreased to 100, but becomes noisier below this threshold. Moreover, with 100 permutations, over 134 core-hours are required. **b**, Differential prioritization with 30 permutations (left) or 100 permutations (right). **c**, Correlations between differential prioritization $-\log_{10}$ $P$-values for each cell type in the reduced datasets with 30 permutations (left) or 100 permutations (right), compared with the full dataset of 1,000 permutations shown in **a**. **d**, Correlation of $-\log_{10}$ $P$-values to the full dataset for between 2 and 999 permutations. **e**, Total runtime required to perform between 1 and 1,000 permutations. **f**,**g**, Impact of reducing the number of subsamples on differential prioritization. A full 50 subsamples are required in each permuted dataset for accurate differential prioritization. **f**, Differential prioritization with one, five or ten subsamples per permutation. **g**, Correlations between differential prioritization $-\log_{10}$ $P$-values for each cell type in the reduced datasets with one, five or ten subsamples per permutation, compared with the complete dataset shown in **a**. **h**, Correlation coefficients to the full dataset with one, five or ten subsamples per permutation. Error bars show 95% confidence interval. **i**, Distribution of mean AUCs with 1, 5, 10 or 50 subsamples per permutation. The variance of null distribution is inflated with <50 subsamples per permutation, which precludes differential prioritization. **j**, Distribution of mean AUCs in the complete dataset of 1,000 permutations (‘default’), or an equivalent number of mean AUCs sampled with replacement from a background of 100, 500 or 1,000 total subsamples, with 50 subsamples per permutation. The null distribution using sampling with replacement is indistinguishable from the null distribution in the complete dataset. **k**–**n**, Sampling with replacement enables accurate differential prioritization at dramatically reduced computational cost, providing an optimized workflow for differential prioritization. **k**, Differential prioritization after sampling with replacement from a background of 100, 500 or 1,000 total subsamples. The original results from the complete dataset are approximated with 500 or more subsamples. **l**, Correlation of $-\log_{10}$ $P$-values to the full dataset for between 100, 500 and 1,000 total permutations. **m**, Correlation coefficients to the full dataset with between 50 and 1,000 mean AUCs drawn from a background of 100, 500 or 1,000 subsamples. **n**, Total runtime required to perform the full permutation analysis versus 100, 500 or 1,000 total permutations using `augur_mode = "permute”`.



**Extended Data Fig. 2 | See next page for caption.**

◀ **Extended Data Fig. 2 | Augur outperforms DE-based methods with subsampling.** Cell-type prioritization in simulated scRNA-seq data[74] from a tissue with eight cell types and increasingly unequal numbers of cells per type, as quantified by the Gini coefficient[29]. The average number of DE genes at 5% false discovery rate in 50 subsamples of 20 cells per condition was tallied using six different statistical tests (*t*-test, Wilcoxon rank-sum test, likelihood ratio test[75], logistic regression[76], MAST[77] and a negative binomial generalized linear model), implemented through the Seurat 'FindMarkers' function. The accuracy of cell-type prioritization was quantified as the Pearson correlation between the cell-type prioritizations (AUC or average number of DE genes, for Augur and single-cell differential expression tests, respectively) and the true proportion of DE genes under the simulation ground truth. The mean of five simulation replicates is shown throughout. Insets show binomial *P*-values for the sign of the difference in correlations (that is, the frequency with which Augur outperforms single-cell differential expression with subsampling), all with $n = 120$. **a**,**b**, Impact of perturbation intensity (differential expression effect size) on cell-type prioritization for a representative test for single-cell differential gene expression (Wilcoxon rank-sum test). Augur outperforms single-cell differential expression with subsampling in prioritizing cell types in the context of by subtler perturbations. **c**,**d**, Impact of sequencing depth (% of reads downsampled) on cell-type prioritization for a representative test for single-cell differential gene expression (Wilcoxon rank-sum test), with the location parameter of the differential expression factor log-normal distribution set to 0.5. Augur outperforms single-cell differential expression with subsampling in more sparsely sequenced datasets. **e**,**f**, Impact of perturbation intensity on cell-type prioritization for five additional tests for single-cell differential gene expression. **g**,**h**, Impact of sequencing depth on cell-type prioritization for five additional tests for single-cell differential gene expression.