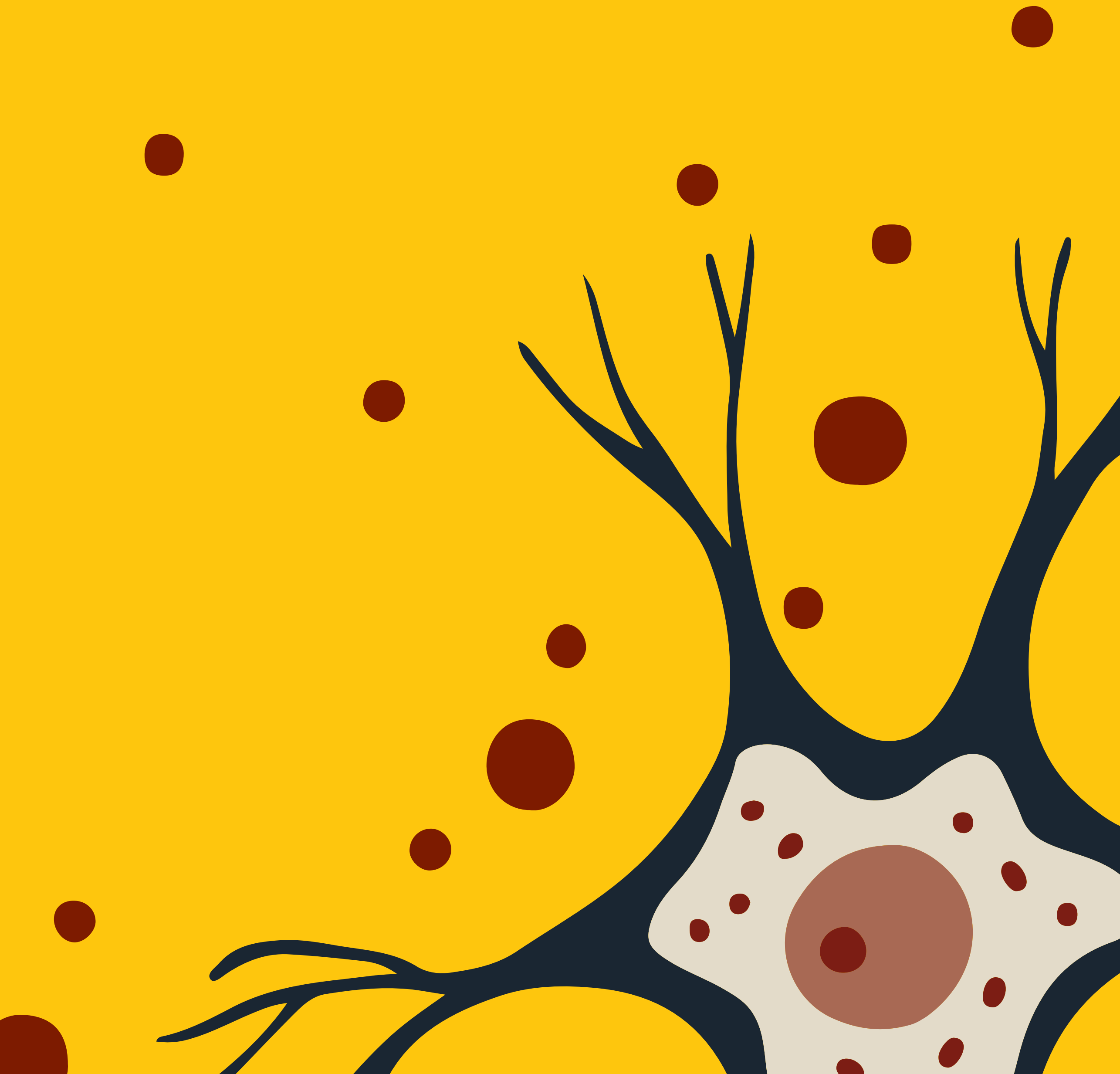
LEARNING CELL IDENTITIES AND (POST-)TRANSCRIPTIONAL REGULATION USING SINGLE-CELL DATA

Lieke Michielsen



Learning cell identities and (post-)transcriptional regulation using single-cell data

Lieke Michielsen

ISBN: 978-94-6506-008-8

Layout & cover design: Jim Iliohan, Cato van Lieshout, Lieke Michielsen
Printing: Ridderprint, the Netherlands

© 2024, Lieke Michielsen.

The research described in this thesis was financially supported by an NWO Gravitation project: BRAINSCAPES: A Roadmap from Neurogenetics to Neurobiology.

All rights reserved. No part of this thesis may be reprinted, reproduced, stored in a retrievel system or transmitted in any other form by any means, without the permission of the author, or when appropriate of the publisher of the represented published articles.

Learning cell identities and (post-)transcriptional regulation using single-cell data

Proefschrift

ter verkrijging van
de graad van doctor aan de Universiteit Leiden,
op gezag van rector magnificus prof.dr.ir. H. Bijl
volgens besluit van het college voor promoties
te verdedigen op donderdag 13 juni 2024
klokke 11:15 uur

door

Lieke Catharina Maria Michielsen

geboren te 's Hertogenbosch
in 1996

Promotors:
Prof.dr.ir. B.P.F. Lelieveldt
Prof.dr.ir. M.J.T. Reinders

Co-promotor:
Dr. A. Mahfouz

Promotiecommissie:
Prof.dr. W.M.C. van Roon-Mom
Prof.dr. D. Posthuma
Vrije Universiteit Amsterdam
Dr. K.P. Kenna
Universitair Medisch Centrum Utrecht
Dr. L. Cantini
Institut Pasteur, France

TABLE OF CONTENTS

Summary 7

Samenvatting 9

Chapter 1 Introduction 11

PART I - Learning cell identities in scRNA-seq data

Chapter 2 A comparison of automatic cell identification methods for single-cell RNA sequencing data 35

Chapter 3 Hierarchical progressive learning of cell identities in single-cell data 67

Chapter 4 Single-cell reference mapping to construct and extend cell-type hierarchies 95

Chapter 5 Cell type matching across species using protein embeddings and transfer learning 115

PART 2 - Using scRNA-seq data to understand (post-) transcriptional regulation

Chapter 6 Predicting cell population-specific gene expression from genomic sequence 133

Chapter 7 Predicting cell-type-specific exon inclusion in the human brain reveals more complex splicing mechanisms in neurons than glia 153

Chapter 8 Discussion 177

Acknowledgements 189

Curriculum vitae 191

List of publications 193

SUMMARY

Tissues in the human body, and especially the brain, are heterogeneous and consist of many different cell types. Cell types can be defined by the genes expressed in a cell, and these expressions are controlled by unique cell-type-specific (post-)transcriptional mechanisms. Diseases can perturb these control mechanisms, and thus affect cell types differently. Consequently, understanding which cell type is affected by a disease is crucial information when developing new drugs or treatments. Single nucleotide polymorphisms (SNPs) in the DNA can be associated with diseases, but approximately 95% of such SNPs fall in the non-coding region of the DNA. Usually, it is unknown whether these variants are causal, and which gene and cell type they affect. Studying gene expression at the single-cell level could reveal such disrupted mechanisms.

Current advances in single-cell RNA sequencing have greatly improved our understanding of heterogeneous tissues and led to the discovery of many new cell types. However, this new technology also presents computational challenges. For example, when comparing datasets from different cohorts (e.g., across many different individuals) it is important to annotate cells consistently. To ensure such consistency, it is essential to annotate cells using classification methods instead of currently practiced clustering methods that are subjective and time-consuming. To facilitate this transition, in this thesis, we benchmarked cell-type classification methods and developed computational methods to automatically build reference atlases using multiple already labeled single-cell datasets. We show how such reference atlases can be deployed to automatically annotate new (unlabeled) single-cell datasets, as well as how they can be updated continuously using new labeled single-cell datasets.

Having established a more consistent cell-type annotation across single-cell datasets, we return to establishing a relationship between mutations and their effect on gene expression. Hereto, we study sequence-to-expression models that can predict an alteration in expression when a mutation is observed. Given that gene expression mechanisms are cell-type specific, we introduce sequence-to-expression models based on single-cell data to make cell-type-specific predictions. We use these models to show that certain mutations are indeed changing gene expression, increasing our understanding of transcriptional regulation.

Next to differences in gene expression between cell types, cell types might express different isoforms of a gene (i.e., different combinations of exons included in an mRNA molecule). Again, this can be altered by mutations in the DNA. Advances in single-cell long-read sequencing enabled measuring which cell types express which isoforms. We leveraged this data and propose a novel approach in which we adapted our sequence-to-expression models to predict cell-type-specific isoform usage. This opens a new avenue for looking at cell-type-specific alterations.

Taken together, we introduce a variety of computational methods to enhance single-cell RNA sequencing data to improve our understanding of cellular heterogeneity.

SAMENVATTING

Weefsels in het menselijk lichaam, in het bijzonder de hersenen, zijn heterogeen en bestaan uit veel verschillende celtypen. Celtypen kunnen worden gedefinieerd door de genen die tot expressie komen in een cel, wat gecontroleerd wordt door unieke celtypespecifieke (post-) transcriptionele mechanismen. Ziekten kunnen deze controlemechanismen verstoren en dus een verschillend effect hebben op celtypes. Begrijpen welk celtype wordt beïnvloed door een ziekte is daarom cruciaal bij het ontwikkelen van nieuwe medicijnen. Single-nucleotide-polymorfismen (SNPs) in het DNA kunnen geassocieerd worden met ziekten, maar ongeveer 95% van de SNPs maakt deel uit van het niet-coderende DNA. Meestal is het onbekend of een SNP de oorzaak van een ziekte is en welk gen en celtype wordt beïnvloed. Het bestuderen van genexpressie op celniveau zou zulke verstoorde mechanismen kunnen onthullen.

De vooruitgang in single-cell RNA sequencing heeft ons begrip van heterogene weefsels sterk verbeterd en geleid tot de ontdekking van veel nieuwe celtypes. Deze nieuwe technologie brengt echter ook computationele uitdagingen met zich mee. Bij het vergelijken van datasets van verschillende cohorten (bijvoorbeeld van veel verschillende individuen) is het belangrijk om cellen consistent te annoteren. Om deze consistentie te garanderen, is het essentieel om cellen te annoteren met behulp van classificatiemethoden in plaats van de huidige clustermethoden, die subjectief en tijdrovend zijn. Om deze overgang te vergemakkelijken, hebben we in dit proefschrift classificatiemethoden voor celtypen gebenchmarkt en computationele methoden ontwikkeld om automatisch referentie-atlassen te bouwen met behulp van meerdere reeds gelabelde single-cell datasets. We laten zien hoe dergelijke referentie-atlassen kunnen worden ingezet om automatisch nieuwe (ongelabelde) single-cell datasets te annoteren en hoe ze continu kunnen worden bijgewerkt met behulp van nieuwe single-cell datasets.

Met de meer consistente annotatie van celtypen in single-cell data, gaan we terug naar de relatie tussen mutaties en hun effect op genexpressie. Hiertoe bestuderen we sequentie-naar-expressie modellen die een verandering in expressie kunnen voorspellen wanneer een mutatie wordt waargenomen. Aangezien genexpressie celtypespecifiek is, introduceren we sequentie-naar-expressiemodellen getraind op single-cell data om celtypespecifieke voorspellingen te doen. We gebruiken deze modellen om aan te tonen dat bepaalde mutaties inderdaad genexpressie veranderen, wat ons begrip van transcriptionele regulatie vergroot.

Naast verschillen in genexpressie tussen celtypen, kunnen celtypen ook verschillende isovormen van een gen tot expressie brengen (d.w.z. verschillende combinaties van exonen in een mRNA-molecuul). Ook dit kan worden veranderd door mutaties in het DNA. Single-cell long-read sequencing maakt het mogelijk om expressie van isovormen in celtypes te meten. We gebruiken deze data en stellen een nieuwe aanpak voor waarbij we onze sequentie-naar-expressiemodellen gebruiken om celtypespecifiek isovormgebruik te voorspellen. Dit opent een nieuwe weg voor het bekijken van celtypespecifieke veranderingen.

Alles bij elkaar introduceren we een scala aan computationele methoden om single-cell RNA sequencing data te gebruiken om ons begrip van cellulaire heterogeniteit te verbeteren.

chapter 1

Introduction

In the 17th century, Robert Hooke discovered something fascinating when analyzing a piece of cork under a microscope: the cork consists of tiny pores. This reminded him of the cells in a monastery and therefore he called these pores ‘cells’ [1]. Almost two centuries later, Matthias Jakob Schleiden and Theodor Schwann formulated the first concept of cell theory: every organism consists of either one or multiple cells, and cells are the building blocks of life [2,3]. We estimate that the human body consists of ~3.7e13 cells [4].

Looking at our own human body, we know that cells have different functions. For example, immune cells fight against pathogens, skeletal muscle cells help us move, and sensory nerve cells receive information from the outside world. How is it possible that all these cells share the same DNA yet execute such a variety of functions? To explain this, we must understand the central dogma of molecular biology that describes the genetic flow of information in a cell (Figure 1) [5,6]. In every cell, there are chromosomes, very long DNA molecules, that provide the genetic code for an organism. Some parts of the DNA sequence, called genes, are transcribed into RNA molecules. Even though many different types of RNA exist with all important functions, we will focus on messenger RNA (mRNA) here. As the name already suggests, these mRNA molecules come from protein-coding genes and are translated into proteins. The resulting protein has a specific function in a cell.

Except for some somatic mutations, however, every cell in an organism has the same DNA. How could a cell know which genes have to be transcribed? Different control mechanisms tightly regulate transcription and translation to ensure the expression of the correct genes and proteins in a cell. For instance, transcription of protein-coding genes starts when RNA polymerase II and auxiliary factors bind the promotor region, the DNA sequence around the transcription start site (TSS) (Figure 2). A group of proteins, transcription factors (TFs), can bind parts of the DNA sequence, called enhancers and silencers, and either activate or repress the binding of RNA polymerase II or the auxiliary factors. This way, transcription factors control

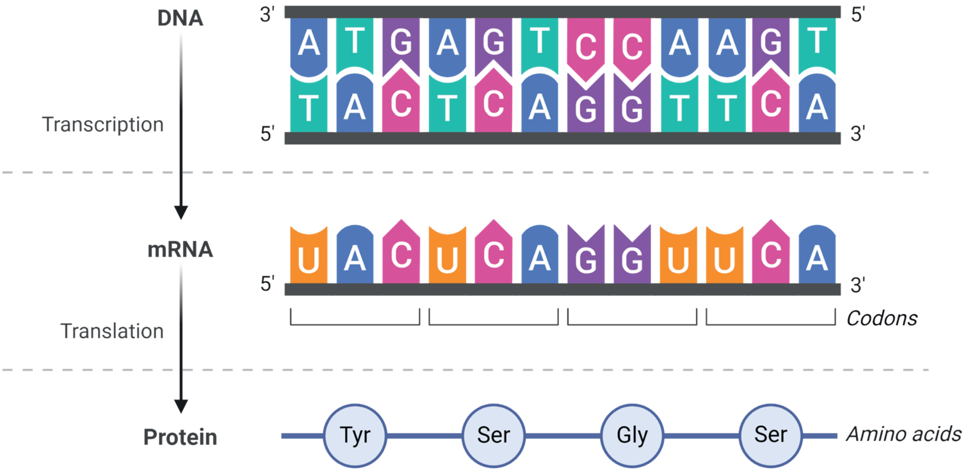


Figure 1. The central dogma of molecular biology. DNA is transcribed into mRNA, which is translated into proteins. [7]

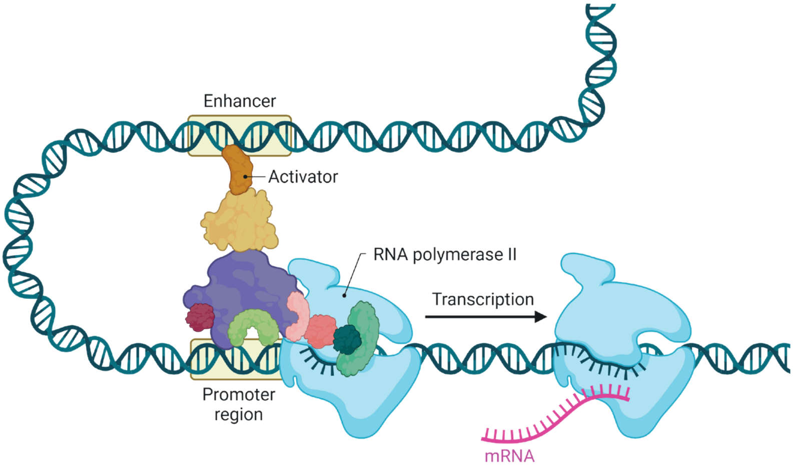


Figure 2. Transcriptional regulation. RNA polymerase II and co-factors must bind to the promoter region to start transcription. Other proteins, called activators, can bind enhancer regions and stimulate this process. The opposite can happen as well. Repressors can bind a silencer region and prevent the RNA polymerase II complex from binding and thus inhibit transcription. [8]

which genes are transcribed in a cell and to which extent. Since humans have approximately 1,400 TFs [9] that can also act combinatorially, the exact regulation mechanisms for each gene are incompletely understood. Understanding transcriptional regulation is important since a mutation in a TF, aberrant expression of a TF, or a mutation in a TF binding site can cause diseases and disorders ranging from cancer, autoimmune diseases, and diabetes to neurological disorders [10-14].

Humans have approximately 20,000 protein-coding genes [15,16]. Some genes, however, can produce different proteins with different functions [17,18]. How can the same mRNA molecule encode different proteins? After transcription, the resulting mRNA molecule has to be processed and spliced before the mature mRNA is transported to the nucleus and translated into a protein (Figure 3A). During the processing, the head and tail are modified to promote stability and export to the nucleus. Splicing, on the other hand, can lead to different proteins. The pre-mRNA molecule consists of exons, the coding regions, and introns. During splicing, the spliceosome, an RNA-protein complex, binds the RNA and catalyzes the removal of the introns. Exons from the same gene can be joined in different combinations, which we call alternative splicing (Figure 3B). Multiple forms of alternative splicing are recognized (Figure 3C). For instance, exons can be included or skipped completely, but alternative splice sites can be used as well. In humans, approximately 90-95% of the genes are alternatively spliced [19,20], which occurs most often in the brain [21].

We can draw a parallel between the regulation of (alternative) splicing and transcription. Where TFs binding the DNA sequence regulate transcription, RNA binding proteins (RBPs) regulate splicing. RBPs can either activate or repress the binding of the spliceosome and thereby control the splicing of exons or introns [22]. Aberrant splicing, for instance, caused by mutations in RBP binding sites, is a hallmark of many neurological diseases [23,24].

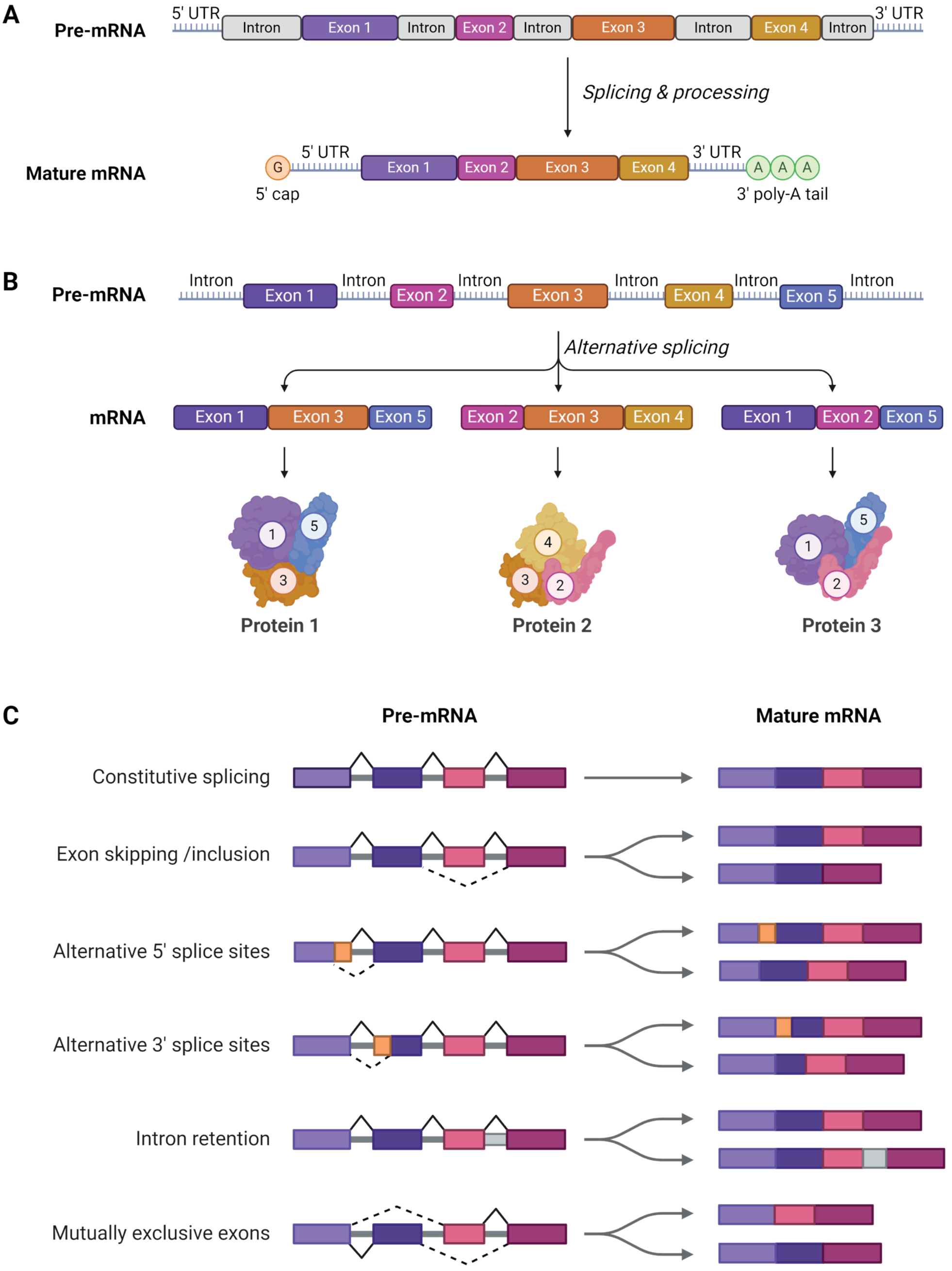


Figure 3. mRNA processing and splicing. **A)** The 5’ cap is added, the tail is polyadenylated, and the introns are spliced out. Afterwards, the mRNA can be transported to the nucleus and translated into a protein. [25] **B)** Alternative splicing. The pre-mRNA can be spliced in different ways. Different combinations of exons can be included in the mRNA molecule which will result in different proteins after translation. [25,26] **C)** Overview of different mRNA splicing types. [27]

1.1 Measuring transcription

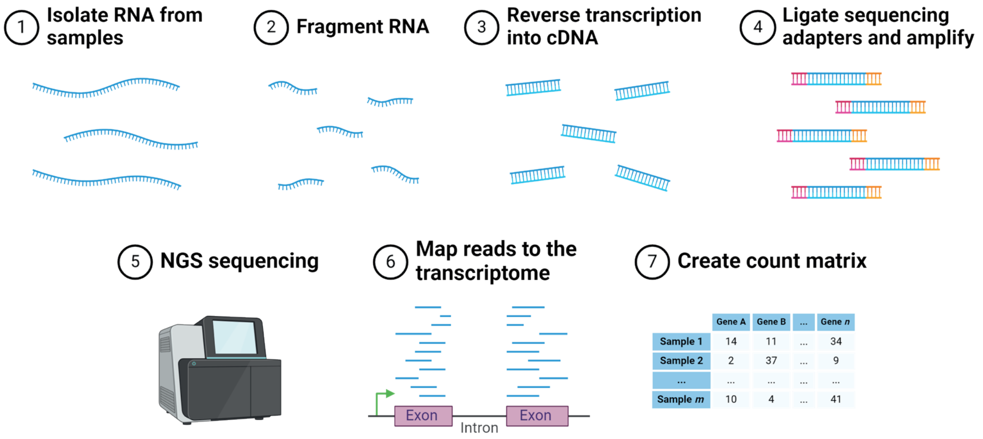
To increase our understanding of cells in health and disease, we quantify which genes are expressed. RNA sequencing is a high-throughput technique to measure the number of mRNA molecules in a sample. This is often done using next-generation sequencing (NGS) technologies such as Illumina and Ion Torrent [28]. The general workflow consists of the following steps: 1) isolating the RNA from the cells, 2) fragmenting the RNA, 3) converting the mRNA into cDNA using reverse transcription, 4) ligating sequence adapters, 5) sequencing using a sequencing platform, 6) mapping the reads to the reference transcriptome, 7) constructing a count matrix (Figure 4). The final count matrix indicates how often a gene was measured in a sample.

NGS technologies generate relatively short reads. For instance, the read length is only 150 bp for most Illumina platforms. This short read length makes it impossible to study complete isoforms since the average length of human protein-coding transcripts is approximately 2.8kb [29]. Some reads map to splice junctions, so from such reads, we can extract whether exons are skipped or if alternative 3' or 5' splice sites are used.

1.1.1 Single-cell RNA sequencing

NGS techniques have been developed to measure transcription in groups of cells. This has the downside that the signal is evened out. If a gene's expression differs between two samples, it is impossible to know whether the sample consists of the same cells with altered expression or whether the cell-type composition changed (Figure 5A). This is especially disadvantageous when analyzing heterogeneous tissues, such as the brain.

In 2009, a new revolution began: single-cell RNA sequencing (scRNA-seq) [30]. Using scRNA-seq, the tissue is dissociated and the gene expression of individual cells can be measured


① Isolate RNA from samples
② Fragment RNA
③ Reverse transcription into cDNA
④ Ligate sequencing adapters and amplify
⑤ NGS sequencing
⑥ Map reads to the transcriptome
Exon
Intron
Exon
⑦ Create count matrix

	Gene A	Gene B	...	Gene *n*
Sample 1	14	11	...	34
Sample 2	2	37	...	9
...	...	...	...	...
Sample *m*	10	4	...	41

Figure 4. Overview of next-generation sequencing. [36]

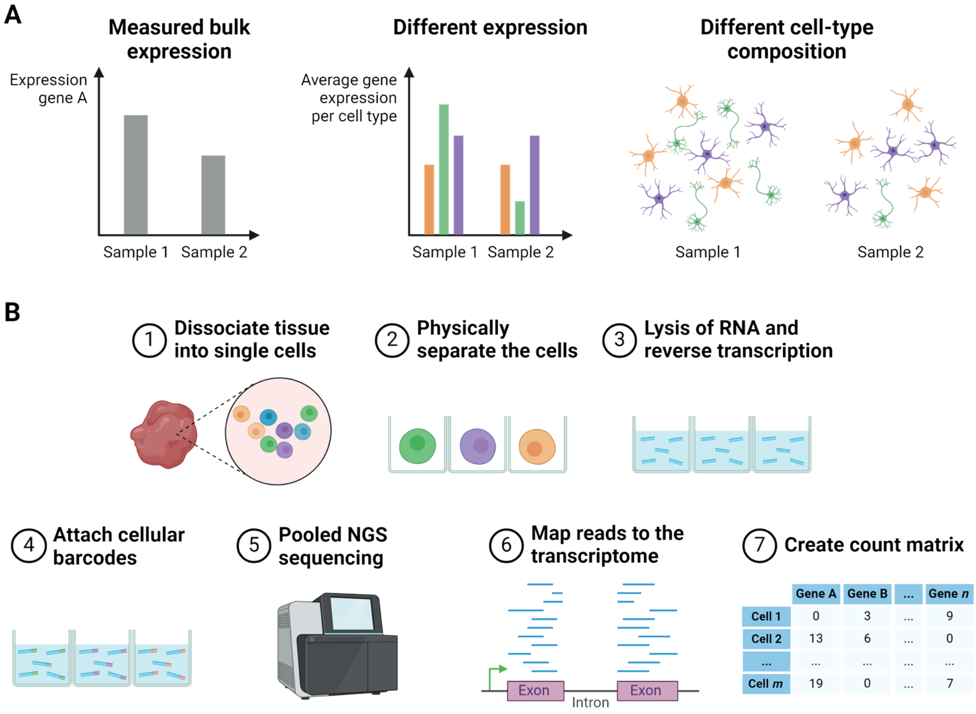

A
Measured bulk expression
Expression gene A
Sample 1
Sample 2
Different expression
Average gene expression per cell type
Sample 1
Sample 2
Different cell-type composition
Sample 1
Sample 2
B
1 Dissociate tissue into single cells
2 Physically separate the cells
3 Lysis of RNA and reverse transcription
4 Attach cellular barcodes
5 Pooled NGS sequencing
6 Map reads to the transcriptome
Exon
Intron
Exon
7 Create count matrix
Gene A
Gene B
...
Gene n
Cell 1
0
3
...
9
Cell 2
13
6
...
0
...
Cell m
19
0
...
7

Figure 5. Single-cell RNA sequencing. **A)** The disadvantage of bulk RNA sequencing. Multiple scenarios can explain the decreased expression of gene A in sample 2. For instance, the expression of gene A decreased in the green cell type, or the cell-type composition changed which resulted in fewer green cells in sample 2. **B)** The general pipeline of single-cell RNA sequencing. This is similar to bulk RNA sequencing, except that cells are physically separated and cellular barcodes are attached to the cDNA. [36,37]

instead [31-35] (Figure 5B). The process is quite similar to sequencing in bulk, except that the cells are physically separated from each other and a barcode is attached to every cDNA molecule after reverse transcription. This barcode informs which reads originated from the same cell during the mapping step later. After barcoding, all material is pooled and sequenced together using an NGS platform. During the mapping step, the reads are split into the barcode and cDNA sequence. Based on the barcode, we know which cell the molecule came from, and based on the cDNA we know which gene was expressed.

In general, the data generated by all scRNA-seq protocols is sparse -around 90% of the values in the count matrix are zeros. Furthermore, when more cells are measured during an experiment, the sparser the data becomes [38]. Both biological and technical limitations explain this sparsity. Even essential genes will not always be expressed in a cell. Transcriptional bursting is the phenomenon in which genes are actively transcribed for a short period followed by a longer period of silence, which causes temporal fluctuation in gene levels [33]. Furthermore, since the mRNA content in a cell is low, it is difficult to capture all molecules.

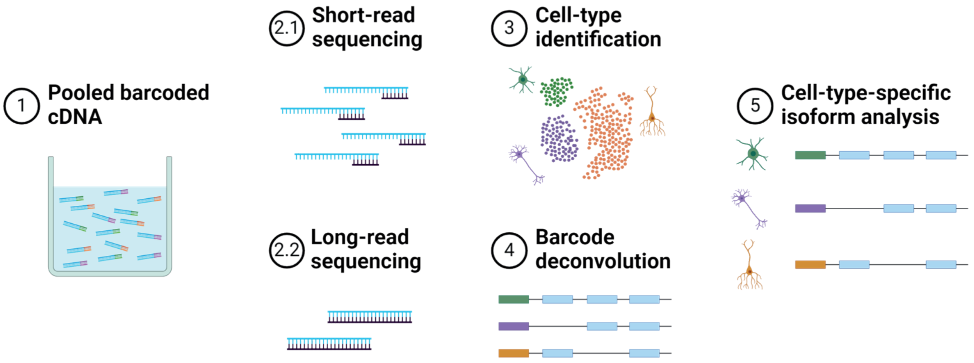
Broadly, scRNA-seq methods can be split into two groups: either the full transcript is sequenced, which is similar to bulk analysis (e.g., using Smart-Seq2 [34]), or only the 3’ or

5' end of the molecule can be captured and counted (e.g., using 10x Chromium [35]). An advantage of Smart-Seq2 is that the cells are sequenced deeper, which results in less sparse data. Furthermore-similar to bulk RNA sequencing- the reads can cover splice junctions. On the other hand, 10x optimized their pipeline for sequencing many cells simultaneously at a low cost. Up to hundreds of thousands of cells can be sequenced per experiment compared to thousands with Smart-Seq2. However, 10x only captures the 3' or 5' end of the mRNA molecule and ~100 nucleotides are measured. This short part of the sequence is enough to differentiate between all genes but lacks information about splice sites.

1.1.2 Long-read single-cell sequencing

To study alternative splicing, one would ideally sequence the whole mRNA molecule instead of looking at short fragments. Two technologies facilitate this nowadays: Oxford Nanopore [39,40] and PacBio [41]. Using Oxford Nanopore either the RNA molecule or the cDNA passes through a pore, which creates a changing electrical current. A base caller deciphers the order of nucleotides that generated these currents. PacBio uses single-molecule real-time (SMRT) sequencing which means that the cDNA molecule of interest is replicated using DNA polymerase. The incorporated new nucleotides are all fluorescent, with the four different bases each having a different fluorescent tag. When a nucleotide is incorporated, the fluorescent tag is cut off and a detector detects the fluorescent signal to decode the order of nucleotides.

Many different human tissues have been sequenced using such long-read protocols, which enhanced the discovery of more than 70.000 new transcripts [42]. This, however, is all in bulk. These protocols have been applied to single cells as well, but initially, only up to a hundred cells could be sequenced [43,44]. Several protocols have been developed to increase the throughput of long-read single-cell sequencing methods [45–47]. For example, some protocols combine short- and long-read sequencing (Figure 6) [48,49]. The single cells are barcoded using the 10x approach. After amplification, the cDNA is split into two pools. One pool is sequenced using Illumina and the other using Oxford Nanopore or PacBio. Due to



1 Pooled barcoded cDNA
2.1 Short-read sequencing
2.2 Long-read sequencing
3 Cell-type identification
4 Barcode deconvolution
5 Cell-type-specific isoform analysis

Figure 6. Schematic overview of long-read single-cell sequencing. The pooled barcoded cDNA is split into two pools. The first part is sequenced using short-read technologies, which can be used for cell-type identification. The second part is sequenced using long-read technologies. Since the barcodes of the short- and long-read data are similar, the data can be combined to study cell-type-specific isoforms. Figure adapted from Joglekar et al. (2021) [50].

the high costs of long-read sequencing, the coverage of the short reads generated by Illumina is usually higher, which results in better gene quantification and can be used to group the cells into specific cell types (see Section 1.2.1). The short-read barcodes are also present in the long-read data and can assign a cell and a cell type to every long-read. The long reads can be grouped per cell type and be used to study cell-type-specific isoform usage.

1.2 Cell types

Studying individual cells in scRNA-seq data is challenging since the data is sparse. Therefore, cells are grouped into cell types, which greatly reduces the complexity of the analysis, especially for organisms with as many cells as humans. But what is a cell type? How do we define them? The concept of a cell type might seem intuitive, but a clear definition is still missing.

In the past, cells were mainly studied under the microscope, so cell types were defined based on morphology. Camillo Golgi, for instance, developed a staining technique to visualize neurons that could later be used to classify them based on their dendritic patterns [51]. Nowadays, more and more features are measured, which changes our groupings of cells into cell types. With these new techniques, we can define a cell type based on which genes or proteins are expressed in a cell [52].

Even though the definition of cell types is dynamic, Cell Ontology [53] attempts to structure all identifiable cell types into a hierarchy. Most cells can be classified at different levels. For instance, a cell can be a blood cell, a lymphoid cell, a T cell, and so on (Figure 7). This hierarchical structure is inherent to cell types since all cells develop from the same cell and become gradually more specialized. The hierarchy shows that some cell types are more similar to one another. However, the cell-type hierarchy does not always align with development.

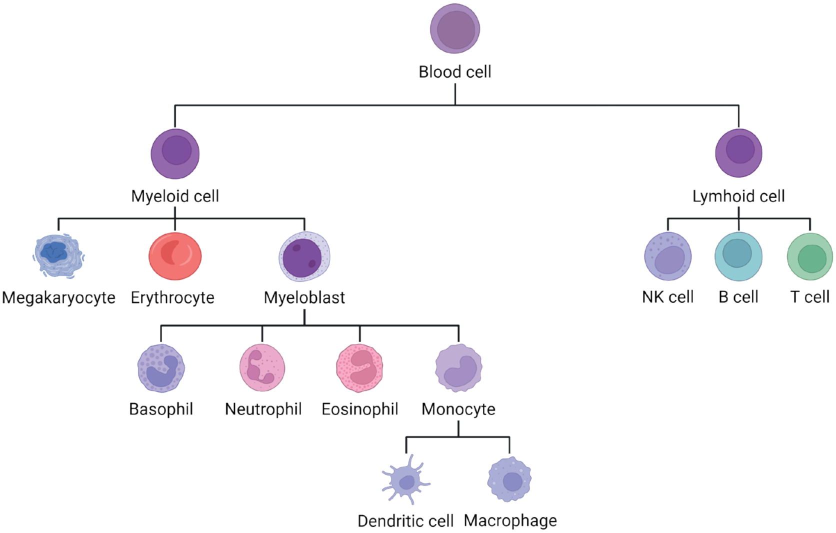


Figure 7. Example of a cell-type hierarchy for blood cells. Figure adapted from Monga et al. (2022) [54]

1.2.1 Discovering cell types in scRNA-seq data

In scRNA-seq data, the cell type of a cell is defined based on which genes are measured in a cell. Because the data is sparse, we cannot determine the cell type of individual cells by looking only at the expression of marker genes. As a solution, we first group cells with a similar gene expression profile and annotate these groups based on the expression of the marker genes (Figure 8A). The standard pipeline from a raw (short-read) scRNA-seq count matrix consists of different preprocessing, clustering, and visualization steps to annotate the clusters, which we will discuss in more detail below [55,56]. Several computational toolkits, such as Scanpy [57] and Seurat [58], have been developed to analyze scRNA-seq data, and all steps discussed below can be performed with these tools. After annotating the cells, other downstream analysis tasks, such as testing for differentially expressed genes between cell types, can be applied.

1.2.1.1 Preprocessing scRNA-seq data

Preprocessing starts with quality control to ensure that only high-quality, viable cells are in the data. Here, for instance, we filter out apoptotic cells based on the high content of mitochondrial genes [59,60]. Next, we normalize the data to remove differences in read depth between the cells. Most often, the data is normalized using library size normalization and log-transformed. After these steps, the dimensionality of the count matrix is still huge since ~20.000 genes are measured. Some of these genes are uniformly expressed across all the cells and uninformative for downstream tasks. We select 1000-5000 genes that show

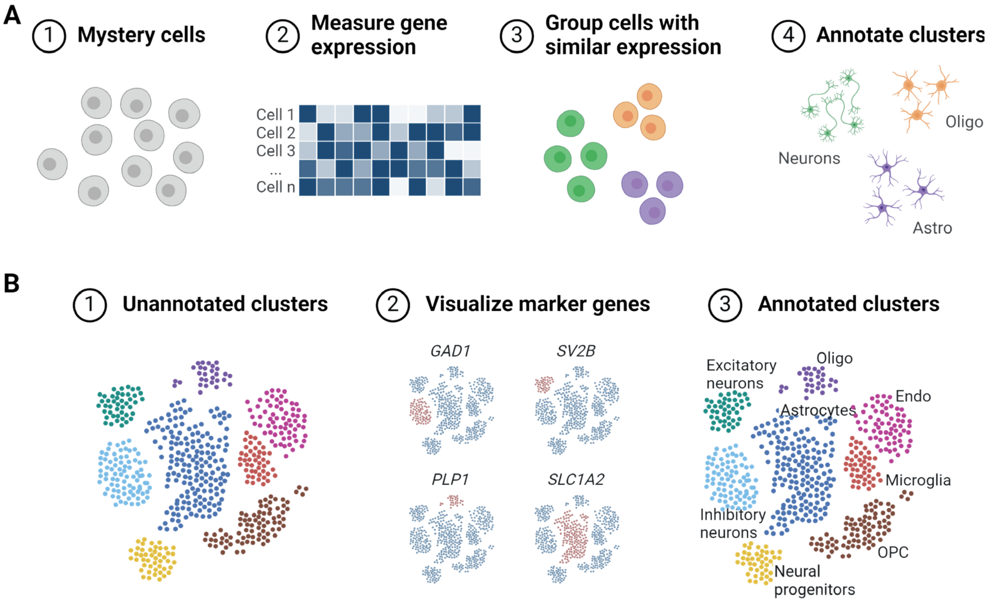


Figure 8. Annotating cell types in single-cell RNA-sequencing data. **A)** Mystery cells are grouped based on their expression pattern and these groups are annotated. **B)** Clusters are annotated by visualizing the expression of marker genes in two dimensions using t-SNE or UMAP.

the most variance in the dataset. Usually, the genes with the highest variance-to-mean ratio are selected. Next, we reduce the dimensions to 30-50 using principal component analysis (PCA). PCA is a linear dimensionality reduction method that reduces the data to a new set of features that is a linear combination of the old features that explain most of the variance. Instead of linear dimensionality reduction methods, non-linear methods can be applied as well. For instance, scVI [61], a variational autoencoder, can map the cells to a latent space of 10-50 dimensions.

1.2.1.2 Identifying cell types in scRNA-seq data

After preprocessing, the data is ready for downstream analysis such as cell-type identification. First, we cluster the data into groups of similar cells. We construct a *k*-nearest-neighbor graph in which every cell is connected to the *k* cells with the most similar gene expression pattern. Next, we detect clusters in this graph using Louvain [62] or Leiden [63] community detection. Here, the resolution parameter influences the number of clusters found. The resulting clusters can be visualized in two dimensions using t-SNE [64] or UMAP [65] (Figure 8B). To annotate the clusters, we visualize the expression of marker genes in, for instance, the two-dimensional space or a dot plot. However, marker genes might be unknown or not clearly expressed in scRNA-seq data, which makes annotating some clusters challenging.

1.3 Supervised learning for scRNA-seq data

In scRNA-seq data, cells are commonly annotated using clustering methods, an example of unsupervised learning. Unsupervised learning means that the data itself is unlabeled (i.e., the cell types are unknown) and the goal is to find groups in the data. However, unsupervised methods have drawbacks: they are subjective and time-consuming. Different parameters yield different clusterings, and the number of clusters or cell types discovered in scRNA-seq data is even correlated with the number of sequenced cells [66-68].

A shift towards supervised methods is needed to overcome this subjectiveness. Supervised models learn the relation between input data (e.g., the measured gene expression) and the label (e.g., the cell type). The trained model can annotate new, unlabeled data automatically. In this example, we predict the cell types that are discrete categories (classification), but supervised models can also be used to predict continuous outcomes (regression).

Many different types of supervised methods exist. Some rely on relatively simple principles and try to find a linear decision boundary between different groups, such as linear discriminant analysis or the linear support vector machine (SVM) (Figure 9A). Other methods, such as a *k*-nearest neighbor (kNN) or nearest mean classifier, look at which samples of the different groups of samples are closest and transfer the closest-group label to the new, unlabeled sample (Figure 9B). Deep learning models, such as neural networks, convolutional neural networks (CNN), and recurrent neural networks (RNN), can learn more complex relationships between the input features and the label (Figure 9C). Deep learning models have the disadvantage that much training data is needed and the models are difficult to interpret. With the linear models, we can easily see which input features guided the decision while

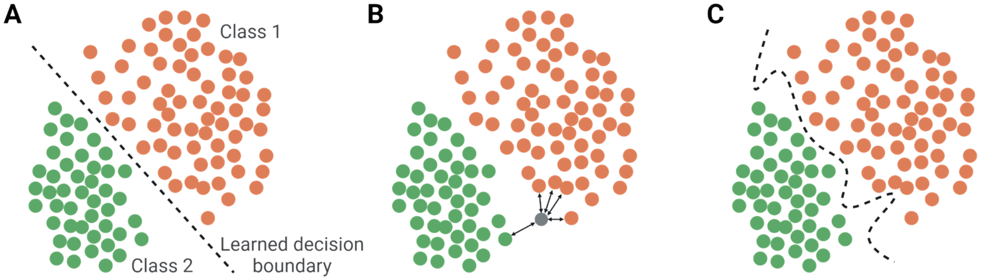


Figure 9. Supervised learning. **A)** Linear classifiers learn a linear decision boundary between the class 1 and class 2 samples. **B)** The *k*-nearest neighbor classifier looks at the neighboring samples and classifies new samples, for example, using a majority vote. In this case, the gray unlabeled sample would be classified as class 1. **C)** Deep learning models can learn complex decision boundaries.

this is impossible to know exactly for deep learning models. Approximation methods, such as Shapley values, exist though [69,70].

Automatic cell-type identification is one example of applying supervised models on scRNA-seq data. In this thesis, we will focus on two types of models: either we use the measured gene expression to predict the cell-type label (Section 1.4), or we know the cell-type label and use a generic input (e.g., the DNA sequence) to predict gene expression or splicing (Section 1.5).

1.4 Part I - Learning cell identities in scRNA-seq data

Ideally, we want to annotate the cells in a new scRNA-seq dataset automatically and consistently by using a classifier trained on an annotated dataset to transfer the labels to this new dataset. Several methods have been developed for this task, each varying considerably in their underlying principles. Some rely on relatively simple machine learning techniques such as a kNN classifier [71,72], SVM [73,74], or random forest (RF) [75–77], while others rely on more complex deep learning architectures [78,79]. We can also categorize methods by whether their approach is flat or hierarchical. Hierarchical methods exploit the inherent hierarchical structure of cell types; instead of learning the differences between all cell types in one go, they split the problem into smaller subproblems. Flat classifiers, on the other hand, do not benefit from this advantage. Another notable example of classifiers is methods that leverage the Cell Ontology [80,81]. Leveraging this ontology might be beneficial in the future, but currently, many newly discovered cell types are still missing in their hierarchy.

1.4.1 Challenges for cell-type identification

Even though many classification methods exist, we still face several challenges when automatically annotating cells.

1.4.1.1 Choosing the training dataset

An enormous amount of scRNA-seq datasets is publicly available, but it remains unclear which one is most optimal to train the classifier. Even datasets from the same tissue will contain different cell types since these datasets are annotated using unsupervised methods. Most research groups are interested in different cell compartments. Their cells of interest might be annotated at a fine-grained resolution, while the other cells are annotated at a low resolution-again relating to the inherent hierarchy of cell types. Comparing the annotations of different datasets can be challenging since a naming convention is missing.

An extra challenge is that most individual studies are incomplete. Rare cell types might be missing completely, or more difficult to discover when looking at one study only. Therefore, multiple datasets should be combined into a reference atlas, as demonstrated by initiatives like the Human Lung Cell Atlas [82]. Here, scRNA-seq data from 14 studies, 107 individuals, and different anatomical locations of the respiratory system is combined into one reference atlas. The cell-type labels of the datasets were manually harmonized using a group of experts, which is very time-consuming. Ideally, annotated datasets from the same tissue could be automatically combined to create a reference atlas.

1.4.1.2 Batch effects between datasets

Unwanted technical variations between datasets pose a second challenge for automatic cell-type identification. These batch effects are caused by variations in sequencing depths, handling of the cells, protocols, laboratories, etc. Consequently, batch effects between datasets should be removed before a classifier can be trained (Figure 10).

Removing batch effects is a trade-off between removing technical variation and preserving biological variation. Methods developed for this task can be categorized into three groups: 1) methods that correct the original gene space, 2) methods that project the data to a corrected latent space, and 3) methods that construct a batch-corrected graph. Methods in the second group usually yield the most optimal performance [83,84]. Another grouping of the current methods depends on whether they adjust all input datasets or allow users to pick one reference and project the query datasets onto it [72,85]. Even though the latter is more difficult, it has the advantage that the reference remains unchanged. As such, a classifier

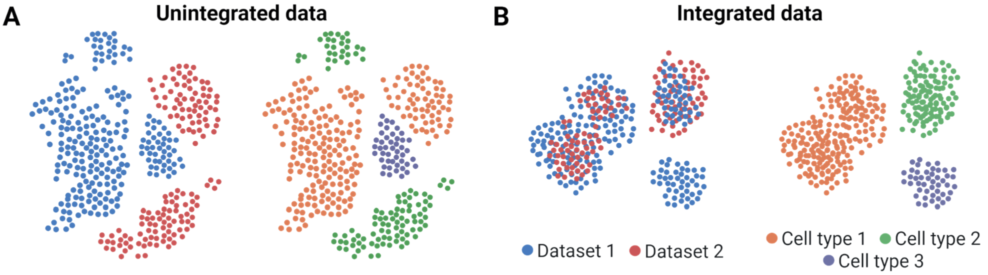


Figure 10. Schematic showing **A)** unintegrated and **B)** integrated scRNA-seq data. In the integrated data, the cells from datasets 1 and 2 overlap.

trained on this reference dataset can be used to annotate any query dataset. Combining these reference mapping methods with an accurate classifier would thus yield a more consistent annotation of the query datasets.

1.4.1.3 Identifying unknown cells

The third challenge for current classifiers is classifying cells as 'unknown' when the label is uncertain. This can be achieved by implementing a rejection option in the classifier. A correctly working rejection option is important for two reasons. First, the border between two cell types might not always be very distinct (Figure 11A). If a cell is close to the decision boundary, the label might be ambiguous and we prefer to keep it unlabeled. A low posterior probability of the classifier is a good indicator of this. Second, some datasets contain new or rare cell types that are not in the training data (Figure 11B). Here, the posterior probability might not work since this only indicates which cell types look most similar to the new cells, but not how similar they are. In this case, a distance metric is required. To correctly identify unknown cells in both scenarios, a classifier needs to use both the posterior probability and a distance metric to reject cells.

1.4.2 Learning cell identities across species

Model organisms, such as mice and rats, are often used to provide insights into biological mechanisms inside a cell or test the effect of new drugs or treatments. Knowing what aspects are similar or different between model organisms and humans is crucial for understanding how results translate. Comparing and matching cell types across species is one fundamental step in this process. Some cell types might be well conserved, while others might be species-specific. Matching cell types is thus interesting from an evolutionary point of view as well and aids in understanding cell-type evolution.

Besides the batch effects described in Section 1.4.1.2, an extra challenge during cross-species comparisons is that the measured gene sets are different. Throughout evolution, genes have been duplicated, deleted, and modified, which results in complex many-to-many relations. Relations between genes of different species are established based on their protein sequence similarity, with the underlying idea that proteins with a similar amino acid sequence will probably execute a similar function [86]. Traditionally, BLAST [87] is used for this task. However, a disadvantage of BLAST is that the whole protein sequence is weighed equally, while certain domains are more important for a specific function. More recently, large language

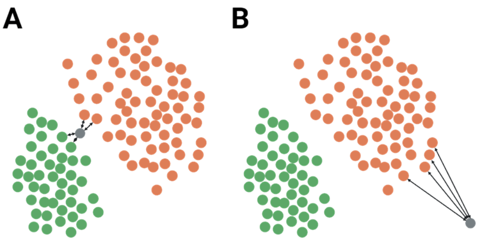


Figure 11. Examples of cells that should remain unlabeled. **A)** The gray cell is close to the decision boundary and therefore it is unclear whether it should be labeled a green or orange. The posterior probability of, for instance, the kNN classifier will be ~0.5, since about half of the neighbors are green and half are orange. **B)** The gray cell is far from the other cell types, which could indicate that it is a new cell type. The closest cells, however, are all orange so the posterior probability will be around one. In this case, a distance metric is needed to reject this cell.

models, such as SeqVec [88] and ProtBERT [89], have been trained to learn a representation of proteins in a lower dimensional space. These embeddings capture functional similarities between proteins and could be used to define homologous genes [90-92].

After matching the genes across species, only one-to-one orthologous genes, which are genes with exactly one match, are commonly used to compare cell types. The scRNA-seq methods developed for same-species data can be applied, which eases downstream analysis. A downside, however, is that much information is ignored. Some methods have been developed for cross-species analysis and use the many-to-many relationships between genes [93,94]. However, these methods currently all rely on the BLAST similarity. Using many-to-many orthologs defined by the protein embeddings would thus greatly enrich the cell type matches made.

1.5 Part II - Using scRNA-seq data to understand (post-) transcriptional regulation

(Post-)transcriptional regulation ensures that every cell expresses the correct genes and isoforms. Since a cell's gene expression level determines its cell type, these regulation mechanisms must be cell-type specific. Which TF or RBP binding sites are used on the DNA or RNA sequence will thus differ per cell type.

Understanding cell-type-specific regulation aids in understanding the underlying fundamental biological processes in a cell, which is, amongst others, essential for drug development. Furthermore, this enables us to predict the effect of mutations in non-coding regions. Mutations in a TF or RBP binding site will only affect gene expression or splicing if that binding site is normally used in that cell type. Knowing which mutations affect which cell types and how, will help to find new targets for drugs or therapies.

1.5.1 Genomic feature prediction models

Training genomic feature prediction models can help to unravel (post-)transcriptional regulation. These models use a generic input, such as the DNA sequence, to predict genomic features, such as gene expression or splicing, that were measured in a sample using RNA-seq. Why is it interesting to train these models though? The model cannot be extrapolated to new genes, as the expression of all genes was measured in the RNA-seq experiment. However, if a model can accurately predict the measured gene expression, interpreting why the model makes a high or low prediction for a gene improves our understanding of regulation. Current genomic feature prediction models can be divided into two groups: 1) feature-extraction-based and 2) sequence-based methods.

1.5.1.1 Feature-extraction-based models

Feature-extraction-based models extract features from the DNA sequence around the TSS of a gene or the RNA sequence around the splice site. These extracted features are used to train a relatively simple model, such as a linear regressor, to predict expression or splicing

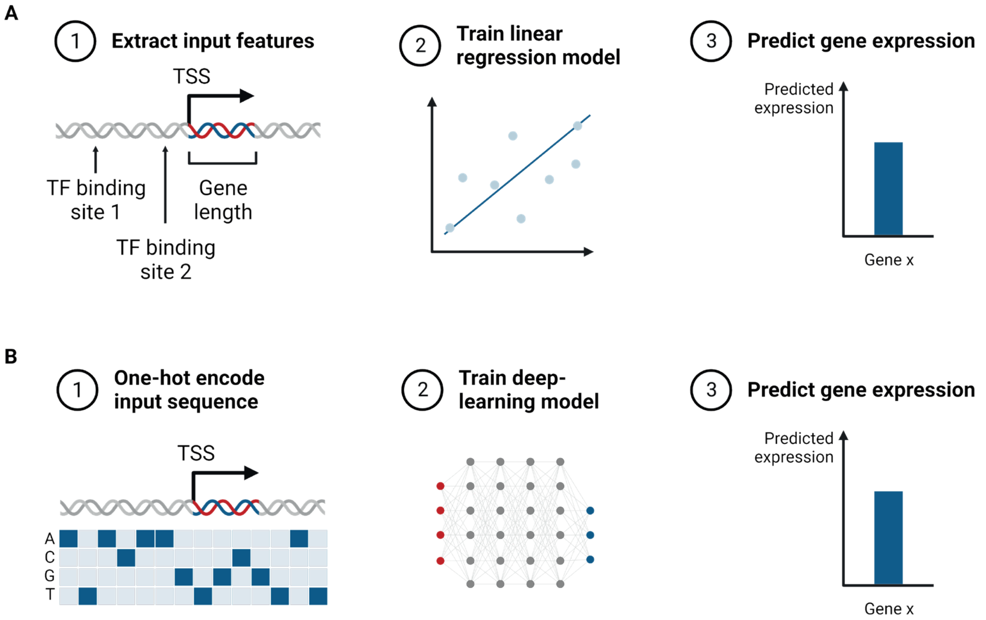

A
1 Extract input features
TSS
TF binding site 1
Gene length
TF binding site 2
2 Train linear regression model
3 Predict gene expression
Predicted expression
Gene x
B
1 One-hot encode input sequence
TSS
A
C
G
T
2 Train deep-learning model
3 Predict gene expression
Predicted expression
Gene x

Figure 12. Schematic of **A)** feature-extraction-based and **B)** sequence-based models to predict genomic features. In this example, the DNA sequence is used to predict gene expression, but the RNA sequence could be used to predict splicing similarly.

(Figure 12A) [95–97]. Examples of extracted features are the gene length, GC content of the gene, and measured or predicted TF or RBP binding sites. The coefficients in the linear regressor directly inform us which features were most important for the predictions, which makes these models easy to interpret. However, we need prior knowledge about extracted features to train a model. If the preferred binding motif for a TF or RBP is unknown, we cannot incorporate it into our models either. Furthermore, evaluating how individual variants affect the prediction is more complicated since the sequence is not directly fed into the model.

1.5.1.2 Sequence-based models

Rapid developments in the deep learning field enabled a shift towards sequence-based methods. Sequence-based methods directly use the (one-hot encoded) DNA or RNA sequence as input to predict gene expression or splicing (Figure 12B) [98–100]. Depending on the task, a window varying from 400bp to 100kb around the TSS or the splice site is used as input. This input is unbiased towards known TFs, RBPs, or other extracted features. More complex models, such as CNNs, RNNs, or transformers, are used to learn the relation between the sequence and expression or splicing.

Training these deep learning models can be challenging since they tend to have millions of free parameters, and the sample size of the training data is limited. The training data size cannot be increased since the number of genes per organism is limited. As a solution, models can be trained on multiple species simultaneously, assuming that the regulatory mechanisms are at least partially conserved [101].

A second challenge is interpreting these black-box models. Model interpretation methods can give insights into what the model learned. One example is examining the initial layer of a CNN. The weights learned by these convolutional weight matrices are comparable to position-weight matrices, which indicate which sequences a TF or RBP prefers to bind [102]. Another option is using *in-silico* saturation mutagenesis (ISM) to systematically predict how nucleotide substitutions in the input sequence affect the predicted value [103,104]. Doing this for many input sequences can reveal interesting patterns that can be detected using TF-MoDISco [105]. TF-MoDISco discovers motifs that are predicted to positively or negatively affect the prediction.

1.5.1.3 Tissue-specific models

In the past, these models were trained using data from cell lines and only learned the basic principles of regulation. The models became more specific by training them, for instance, on bulk RNA-seq data from different tissues. In such cases, either a model per tissue or a multitask model can be trained. The regulation mechanisms, however, are cell-type-specific. Thus, there is a need for training these models on scRNA-seq data instead.

1.6 Contributions of this thesis

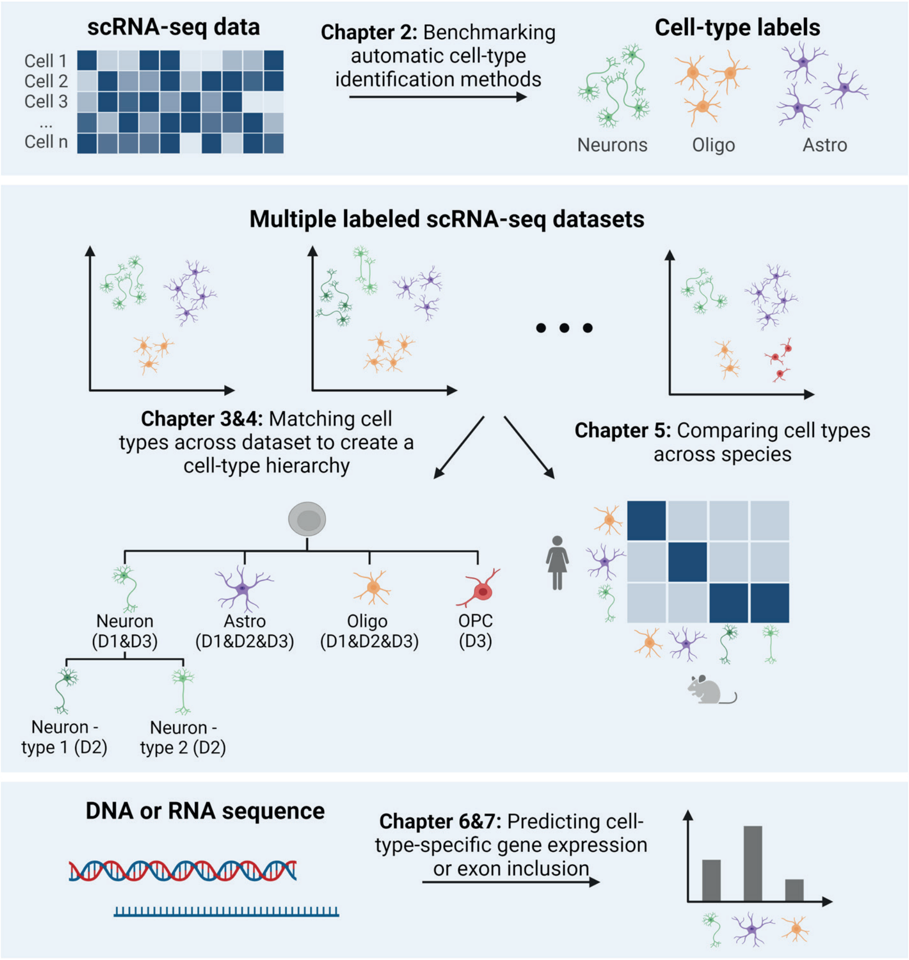
In this thesis, we address several challenges regarding identifying cell types in scRNA-seq data (Part I, Chapters 2-5) and using scRNA-seq datasets to improve our understanding of (post-) transcriptional regulation (Part II, Chapters 6-7).

Part I - Learning cell identities in scRNA-seq data

Chapter 2: In Chapter 2, we benchmark sixteen cell-type identification methods designed for scRNA-seq data and six off-the-shelf Python classifiers. We compare their performance on 27 scRNA-seq datasets of different sizes, number of cell types, species, and technologies. Almost all methods perform well on most datasets, but their performance correlates negatively with the complexity of the data. Most classifiers suffer if a dataset contains many or very similar cell types. Overall, the linear SVM, one of the off-the-shelf Python classifiers, outperforms the methods designed for scRNA-seq data. Furthermore, when benchmarking the rejection options of the classifiers, we noticed that designing a proper rejection option is challenging and that relying on the posterior probability alone is not optimal.

Chapter 3: In Chapter 3, we present single-cell Hierarchical Progressive Learning (scHPL). scHPL combines multiple labeled scRNA-seq datasets into one classifier. We exploit the unharmonized labels of the input datasets to automatically create a cell-type hierarchy by matching the cell types of the different datasets. This hierarchy can either be updated progressively using new, labeled datasets or used as a classifier to annotate the cells in an unlabeled dataset. For every node in the hierarchy, we train a linear SVM since this performed best in the benchmark in Chapter 2. Furthermore, we implemented two rejection options using the posterior probability to reject cells between two cell types and the reconstruction error of the PCA to identify new cell types. We show that scHPL can accurately construct the cell-type hierarchy for PBMC and brain datasets and that scHPL outperforms the flat linear SVM when annotating an unlabeled dataset.

Chapter 4: In Chapter 4, we combine scHPL and scArches [84] into a computational pipeline called treeArches. Before running scHPL, we require datasets to be batch-corrected. A downside of most batch-correction tools is that the complete alignment has to be repeated when adding a new dataset to update the hierarchy. Consequently, the complete hierarchy has to be rebuilt in the new integrated space. Since scArches is a reference-mapping method, it projects a new dataset on top of the reference, which ensures that the reference and corresponding hierarchy do not change. treeArches thus facilitates easy building and extending of reference atlases and the corresponding cell-type hierarchy.


scRNA-seq data
Cell 1
Cell 2
Cell 3
...
Cell n
Chapter 2: Benchmarking automatic cell-type identification methods
Cell-type labels
Neurons
Oligo
Astro
Multiple labeled scRNA-seq datasets
Chapter 3&4: Matching cell types across dataset to create a cell-type hierarchy
Chapter 5: Comparing cell types across species
Neuron (D1&D3)
Astro (D1&D2&D3)
Oligo (D1&D2&D3)
OPC (D3)
Neuron - type 1 (D2)
Neuron - type 2 (D2)
DNA or RNA sequence
Chapter 6&7: Predicting cell-type-specific gene expression or exon inclusion

Chapter 5: In Chapter 5, we propose a model to transfer and align cell types in cross-species analysis (TACTiCS). TACTiCS matches genes of different species using protBERT [89], an NLP model, while allowing for many-to-many matches. Next, it employs a neural network to train species-specific cell-type classifiers. Afterwards, it cross-predicts the other species' labels and compares the predicted to the original labels. TACTiCS outperforms state-of-the-art methods when matching human, mouse, and marmoset cell types in the primary motor cortex.

Part II - Using scRNA-seq data to understand (post-)transcriptional regulation

Chapter 6: In Chapter 6, we extend Xpresso, a tool to predict gene expression in bulk RNA-seq samples, to scXpresso which is a multitask model trained on scRNA-seq data to predict cell-type-specific gene expression. We show that cell-type-specific predictions are especially useful in heterogeneous tissues. In all experiments, cell-type-specific models outperform the tissue-specific models. The difference becomes most apparent when the gene expression of a cell type and the corresponding tissue are dissimilar. Furthermore, we show that scXpresso learns TF binding sites and envision that it will be useful for unraveling cell-type-specific transcriptional regulation mechanisms.

Chapter 7: In Chapter 7, we leverage long-read single-cell data to predict exon inclusion in glia and neurons in the human hippocampus and frontal cortex. We show that splicing is more difficult to predict in neurons than glia. Comparing RBP binding sites for exons with high and low exon inclusion between variable and non-variable exons, we found that these differ more in neurons than in glia, indicating that splicing mechanisms in variable exons in neurons diverged more from the standard mechanisms. Furthermore, we could pinpoint interesting RBPs regulating alternative splicing between glia and neurons.

Chapter 8: Finally, we discuss the contribution of our work in both research directions. First, we discuss how consistent cell-type classification can be improved. Next, we discuss the limitations of current genomic feature prediction models and suggest how these could be tackled.

Bibliography

1. Hooke R. Micrographia: or some physiological descriptions of minute bodies made by magnifying glasses. With observations and inquiries thereupon. London : Printed by Jo. Martyn, and Ja. Allestry ... and are to be sold at their shop ..., 1665.; 1665. Available: https://search.library.wisc.edu/catalog/999581426802121
2. Schwann T, Hünseler F. Mikroskopische Untersuchungen über die Ubereinstimmung in der Struktur und dem Wachstume der Tiere und Pflanzen. W. Engelmann; 1910.
3. Schleiden MJ. Arch Anat Physiol. Wiss Med. 1838.
4. Bianconi E, Piovesan A, Facchin F, Beraudi A, Casadei R, Frabetti F, et al. An estimation of the number of cells in the human body. Ann Hum Biol. 2013;40: 463–471. doi:10.3109/03014460.2013.807878
5. Crick FH. On protein synthesis. Symp Soc Exp Biol. 1958;12: 138–163. Available: https://www.ncbi.nlm.nih.gov/pubmed/13580867
6. Cobb M. 60 years ago, Francis Crick changed the logic of biology. PLoS Biol. 2017;15: e2003243. doi:10.1371/journal.pbio.2003243
7. Reprinted from “Central Dogma”, by BioRender.com (2023). Retrieved from https://app.biorender.com/biorender-templates.
8. Adapted from “Eukaryotic Gene Regulation - Transcriptional Initiation”, created by “Shadma Nafis” using BioRender.com (2023). Retrieved from https://app.biorender.com/biorender-templates.
9. Vaquerizas JM, Kummerfeld SK, Teichmann SA, Luscombe NM. A census of human transcription factors: function, expression and evolution. Nat Rev Genet. 2009;10: 252–263. doi:10.1038/nrg2538
10. Lee TI, Young RA. Transcriptional regulation and its misregulation in disease. Cell. 2013;152: 1237–1251. doi:10.1016/j.cell.2013.02.014
11. Maurano MT, Humbert R, Rynes E, Thurman RE, Haugen E, Wang H, et al. Systematic localization of common disease-associated variation in regulatory DNA. Science. 2012;337: 1190–1195. doi:10.1126/science.1222794
12. Van Houdt JKJ, Nowakowska BA, Sousa SB, van Schaik BDC, Seuntjens E, Avonce N, et al. Heterozygous missense mutations in SMARCA2 cause Nicolaides-Baraitser syndrome. Nat Genet. 2012;44: 445–9, S1. doi:10.1038/ng.1105
13. Lin CY, Lovén J, Rahl PB, Paranal RM, Burge CB, Bradner JE, et al. Transcriptional amplification in tumor cells with elevated c-Myc. Cell. 2012;151: 56–67. doi:10.1016/j.cell.2012.08.026
14. Nie Z, Hu G, Wei G, Cui K, Yamane A, Resch W, et al. c-Myc is a universal amplifier of expressed genes in lymphocytes and embryonic stem cells. Cell. 2012;151: 68–79. doi:10.1016/j.cell.2012.08.033
15. Frankish A, Diekhans M, Jungreis I, Lagarde J, Loveland JE, Mudge JM, et al. GENCODE 2021. Nucleic Acids Res. 2021;49: D916–D923. doi:10.1093/nar/gkaa1087
16. Martin FJ, Amode MR, Aneja A, Austine-Orimoloye O, Azov AG, Barnes I, et al. Ensembl 2023. Nucleic Acids Res. 2023;51: D933–D941. doi:10.1093/nar/gkac958
17. Sabate MI, Stolarsky LS, Polak JM, Bloom SR, Varndell IM, Ghatei MA, et al. Regulation of neuroendocrine gene expression by alternative RNA processing. Colocalization of calcitonin and calcitonin gene-related peptide in thyroid C-cells. J Biol Chem. 1985;260: 2589–2592. doi:10.1016/S0021-9258(18)89396-6
18. Muntoni F, Torelli S, Ferlini A. Dystrophin and mutations: one gene, several proteins, multiple phenotypes. Lancet Neurol. 2003;2: 731–740. doi:10.1016/s1474-4422(03)00585-4
19. Pan Q, Shai O, Lee LJ, Frey BJ, Blencowe BJ. Deep surveying of alternative splicing complexity in the human transcriptome by high-throughput sequencing. Nat Genet. 2008;40: 1413–1415. doi:10.1038/ng.259
20. Wang ET, Sandberg R, Luo S, Khrebtukova I, Zhang L, Mayr C, et al. Alternative isoform regulation in human tissue transcriptomes. Nature. 2008;456: 470–476. doi:10.1038/nature07509
21. Yeo G, Holste D, Kreiman G, Burge CB. Variation in alternative splicing across human tissues. Genome Biol. 2004;5: R74. doi:10.1186/gb-2004-5-10-r74
22. Fisher E, Feng J. RNA splicing regulators play critical roles in neurogenesis. Wiley Interdiscip Rev RNA. 2022;13: e1728. doi:10.1002/wrna.1728
23. Licatalosi DD, Darnell RB. Splicing regulation in neurologic disease. Neuron. 2006;52: 93–101. doi:10.1016/j.neuron.2006.09.017
24. Dredge BK, Polydorides AD, Darnell RB. The splice of life: alternative splicing and neurological disease. Nat Rev Neurosci. 2001;2: 43–50. doi:10.1038/35049061
25. Adapted from “RNA Processing in Eukaryotes”, using BioRender.com (2023). Retrieved from https://app.biorender.com/biorender-templates.
26. Adapted from “Gene Splicing”, using BioRender.com (2023). Retrieved from https://app.biorender.com/biorender-templates.
27. Reprinted from “mRNA Splicing Types”, by BioRender.com (2023). Retrieved from https://app.biorender.com/biorender-templates.

28. Hu T, Chitnis N, Monos D, Dinh A. Next-generation sequencing technologies: An overview. Hum Immunol. 2021;82: 801–811. doi:10.1016/j.humimm.2021.02.012

29. Piovesan A, Caracausi M, Antonaros F, Pelleri MC, Vitale L. GeneBase 1.1: a tool to summarize data from NCBI gene datasets and its application to an update of human gene statistics. Database . 2016;2016. doi:10.1093/database/baw153

30. Tang F, Barbacioru C, Wang Y, Nordman E, Lee C, Xu N, et al. mRNA-Seq whole-transcriptome analysis of a single cell. Nat Methods. 2009;6: 377–382. doi:10.1038/nmeth.1315

31. Carangelo G, Magi A, Semeraro R. From multitude to singularity: An up-to-date overview of scRNA-seq data generation and analysis. Front Genet. 2022;13: 994069. doi:10.3389/fgene.2022.994069

32. Mereu E, Lafzi A, Moutinho C, Ziegenhain C, McCarthy DJ, Álvarez-Varela A, et al. Benchmarking single-cell RNA-sequencing protocols for cell atlas projects. Nat Biotechnol. 2020;38: 747–755. doi:10.1038/s41587-020-0469-4

33. Haque A, Engel J, Teichmann SA, Lönnberg T. A practical guide to single-cell RNA-sequencing for biomedical research and clinical applications. Genome Med. 2017;9: 75. doi:10.1186/s13073-017-0467-4

34. Picelli S, Björklund ÅK, Faridani OR, Sagasser S, Winberg G, Sandberg R. Smart-seq2 for sensitive full-length transcriptome profiling in single cells. Nat Methods. 2013;10: 1096–1098. doi:10.1038/nmeth.2639

35. Zheng GXY, Terry JM, Belgrader P, Ryvkin P, Bent ZW, Wilson R, et al. Massively parallel digital transcriptional profiling of single cells. Nat Commun. 2017;8: 14049. doi:10.1038/ncomms14049

36. Adapted from “RNA Sequencing”, using BioRender.com (2023). Retrieved from https://app.biorender.com/biorender-templates.

37. Adapted from “Single-Cell Sequencing”, using BioRender.com (2023). Retrieved from https://app.biorender.com/biorender-templates.

38. Bouland GA, Mahfouz A, Reinders MJT. Consequences and opportunities arising due to sparser single-cell RNA-seq datasets. Genome Biol. 2023;24: 86. doi:10.1186/s13059-023-02933-w

39. Oikonomopoulos S, Wang YC, Djambazian H, Badescu D, Ragoussis J. Benchmarking of the Oxford Nanopore MinION sequencing for quantitative and qualitative assessment of cDNA populations. Sci Rep. 2016;6: 31602. doi:10.1038/srep31602

40. Sereika M, Kirkegaard RH, Karst SM, Michaelsen TY, Sørensen EA, Wollenberg RD, et al. Oxford Nanopore R10.4 long-read sequencing enables the generation of near-finished bacterial genomes from pure cultures and metagenomes without short-read or reference polishing. Nat Methods. 2022;19: 823–826. doi:10.1038/s41592-022-01539-7

41. Eid J, Fehr A, Gray J, Luong K, Lyle J, Otto G, et al. Real-time DNA sequencing from single polymerase molecules. Science. 2009;323: 133–138. doi:10.1126/science.1162986

42. Glinos DA, Garborcauskas G, Hoffman P, Ehsan N, Jiang L, Gokden A, et al. Transcriptome variation in human tissues revealed by long-read sequencing. Nature. 2022;608: 353–359. doi:10.1038/s41586-022-05035-y

43. Byrne A, Beaudin AE, Olsen HE, Jain M, Cole C, Palmer T, et al. Nanopore long-read RNAseq reveals widespread transcriptional variation among the surface receptors of individual B cells. Nat Commun. 2017;8: 16027. doi:10.1038/ncomms16027

44. Karlsson K, Linnarsson S. Single-cell mRNA isoform diversity in the mouse brain. BMC Genomics. 2017;18: 126. doi:10.1186/s12864-017-3528-6

45. Lebrigand K, Magnone V, Barbry P, Waldmann R. High throughput error corrected Nanopore single cell transcriptome sequencing. Nat Commun. 2020;11: 4025. doi:10.1038/s41467-020-17800-6

46. Al’Khafaji AM, Smith JT, Garimella KV, Babadi M, Popic V, Sade-Feldman M, et al. High-throughput RNA isoform sequencing using programmed cDNA concatenation. Nat Biotechnol. 2023. doi:10.1038/s41587-023-01815-7

47. Tian L, Jabbari JS, Thijssen R, Gouil Q, Amarasinghe SL, Voogd O, et al. Comprehensive characterization of single-cell full-length isoforms in human and mouse with long-read sequencing. Genome Biol. 2021;22: 310. doi:10.1186/s13059-021-02525-6

48. Gupta I, Collier PG, Haase B, Mahfouz A, Joglekar A, Floyd T, et al. Single-cell isoform RNA sequencing characterizes isoforms in thousands of cerebellar cells. Nat Biotechnol. 2018;36: 1197–1202. doi:10.1038/nbt.4259

49. Hardwick SA, Hu W, Joglekar A, Fan L, Collier PG, Foord C, et al. Single-nuclei isoform RNA sequencing unlocks barcoded exon connectivity in frozen brain tissue. Nat Biotechnol. 2022. doi:10.1038/s41587-022-01231-3 50. Joglekar A, Prjibelski A, Mahfouz A, Collier P, Lin S, Schlusche AK, et al. A spatially resolved brain region- and cell type-specific isoform atlas of the postnatal mouse brain. Nat Commun. 2021;12: 463. doi:10.1038/s41467-020-20343-5

50. Joglekar A, Prjibelski A, Mahfouz A, Collier P, Lin S, Schlusche AK, et al. A spatially resolved brain region- and cell type-specific isoform atlas of the postnatal mouse brain. Nat Commun. 2021;12: 463. doi:10.1038/s41467-020-20343-5

51. Abdel-Maguid TE, Bowsher D. Classification of neurons by dendritic branching pattern. A categorisation based on Golgi impregnation of spinal and cranial somatic and visceral afferent and efferent cells in the adult human. J Anat. 1984;138 (Pt 4): 689–702. Available: https://www.ncbi.nlm.nih.gov/pubmed/6204961

52. Wagner A, Regev A, Yosef N. Revealing the vectors of cellular identity with single-cell genomics. Nat Biotechnol. 2016;34: 1145–1160. doi:10.1038/nbt.3711

53. Diehl AD, Meehan TF, Bradford YM, Brush MH, Dahdul WM, Dougall DS, et al. The Cell Ontology 2016: enhanced content, modularization, and ontology interoperability. J Biomed Semantics. 2016;7: 44. doi:10.1186/s13326-016-0088-7

54. Monga I, Kaur K, Dhanda SK. Revisiting hematopoiesis: applications of the bulk and single-cell transcriptomics dissecting transcriptional heterogeneity in hematopoietic stem cells. Brief Funct Genomics. 2022;21: 159–176. doi:10.1093/bfgp/elac002
55. Luecken MD, Theis FJ. Current best practices in single-cell RNA-seq analysis: a tutorial. Mol Syst Biol. 2019;15: e8746. doi:10.15252/msb.20188746
56. Clarke ZA, Andrews TS, Atif J, Pouyabahar D, Innes BT, MacParland SA, et al. Tutorial: guidelines for annotating single-cell transcriptomic maps using automated and manual methods. Nat Protoc. 2021;16: 2749–2764. doi:10.1038/s41596-021-00534-0
57. Wolf FA, Angerer P, Theis FJ. SCANPY: Large-scale single-cell gene expression data analysis. Genome Biol. 2018;19: 15. doi:10.1186/s13059-017-1382-0
58. Stuart T, Butler A, Hoffman P, Hafemeister C, Papalexi E, Mauck WM, et al. Comprehensive Integration of Single-Cell Data. Cell. 2019;177: 1888–1902.e21. doi:10.1016/j.cell.2019.05.031
59. Griffiths JA, Scialdone A, Marioni JC. Using single-cell genomics to understand developmental processes and cell fate decisions. Mol Syst Biol. 2018;14: e8046. doi:10.15252/msb.20178046
60. Ilicic T, Kim JK, Kolodziejczyk AA, Bagger FO, McCarthy DJ, Marioni JC, et al. Classification of low quality cells from single-cell RNA-seq data. Genome Biol. 2016;17: 29. doi:10.1186/s13059-016-0888-1
61. Gayoso A, Lopez R, Xing G, Boyeau P, Wu K, Jayasuriya M, et al. scvi-tools: a library for deep probabilistic analysis of single-cell omics data. bioRxiv. 2021. p. 2021.04.28.441833. doi:10.1101/2021.04.28.441833
62. Blondel VD, Guillaume J-L, Lambiotte R, Lefebvre E. Fast unfolding of communities in large networks. J Stat Mech: Theory Exp. 2008. doi:10.1088/1742-5468/2008/10/P10008
63. Traag VA, Waltman L, van Eck NJ. From Louvain to Leiden: guaranteeing well-connected communities. Sci Rep. 2019;9: 5233. doi:10.1038/s41598-019-41695-z
64. van der Maaten L, Hinton G. Visualizing Data using t-SNE. J Mach Learn Res. 2008;9: 2579–2605. Available: https://jmlr.org/papers/v9/vandermaaten08a.html
65. McInnes L, Healy J, Melville J. UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction. 2018. Available: http://arxiv.org/abs/1802.03426
66. Svensson V, da Veiga Beltrame E, Pachter L. A curated database reveals trends in single-cell transcriptomics. Database. 2020;2020. doi:10.1093/database/baaa073
67. Tasic B, Menon V, Nguyen TN, Kim TK, Jarsky T, Yao Z, et al. Adult mouse cortical cell taxonomy revealed by single cell transcriptomics. Nat Neurosci. 2016;19: 335–346. doi:10.1038/nn.4216
68. Tasic B, Yao Z, Graybuck LT, Smith KA, Nguyen TN, Bertagnolli D, et al. Shared and distinct transcriptomic cell types across neocortical areas. Nature. 2018;563: 72–78. doi:10.1038/s41586-018-0654-5
69. Štrumbelj E, Kononenko I. Explaining prediction models and individual predictions with feature contributions. Knowl Inf Syst. 2014;41: 647–665. doi:10.1007/s10115-013-0679-x
70. Shrikumar A, Greenside P, Kundaje A. Learning Important Features Through Propagating Activation Differences. arXiv [cs.CV]. 2017. Available: http://arxiv.org/abs/1704.02685
71. Kiselev VY, Yiu A, Hemberg M. scmap: projection of single-cell RNA-seq data across data sets. Nat Methods. 2018;15: 359. Available: https://doi.org/10.1038/nmeth.4644
72. Hao Y, Hao S, Andersen-Nissen E, Mauck WM, Zheng S, Butler A, et al. Integrated analysis of multimodal single-cell data. Cell. 2021;0. doi:10.1016/j.cell.2021.04.048
73. Wagner F, Yanai I. Moana: A robust and scalable cell type classification framework for single-cell RNA-Seq data. bioRxiv. 2018; 456129. doi:10.1101/456129
74. Alquicira-Hernandez J, Sathe A, Ji HP, Nguyen Q, Powell JE. ScPred: Accurate supervised method for cell-type classification from single-cell RNA-seq data. Genome Biol. 2019;20: 264. doi:10.1186/s13059-019-1862-5
75. Johnson TS, Wang T, Huang Z, Yu CY, Wu Y, Han Y, et al. LAmbDA: Label Ambiguous Domain Adaptation Dataset Integration Reduces Batch Effects and Improves Subtype Detection. Bioinformatics. 2019. doi:10.1093/bioinformatics/btz295
76. Lieberman Y, Rokach L, Shay T. CaSTLe – Classification of single cells by transfer learning: Harnessing the power of publicly available single cell RNA sequencing experiments to annotate new experiments. Kaderali L, editor. PLoS One. 2018;13: e0205499. doi:10.1371/journal.pone.0205499
77. Tan Y, Cahan P. SingleCellNet: A Computational Tool to Classify Single Cell RNA-Seq Data Across Platforms and Across Species. Cell Syst. 2019;9: 207–213.e2. doi:10.1016/j.cels.2019.06.004
78. Ma F, Pellegrini M. ACTINN: automated identification of cell types in single cell RNA sequencing. Bioinformatics. 2020;36: 533–538. doi:10.1093/bioinformatics/btz592
79. Xu C, Lopez R, Mehlman E, Regier J, Jordan MI, Yosef N. Probabilistic harmonization and annotation of single-cell transcriptomics data with deep generative models. Mol Syst Biol. 2021;17: e9620. doi:10.15252/msb.20209620
80. Wang S, Pisco AO, McGeever A, Brbic M, Zitnik M, Darmanis S, et al. Leveraging the Cell Ontology to classify unseen cell types. Nat Commun. 2021;12: 5556. doi:10.1038/s41467-021-25725-x
81. Bernstein MN, Ma Z, Gleicher M, Dewey CN. CellO: comprehensive and hierarchical cell type classification of human cells with the Cell Ontology. iScience. 2021;24: 101913. doi:10.1016/J.ISCI.2020.101913

82. Sikkema L, Ramírez-Suástegui C, Strobl DC, Gillett TE, Zappia L, Madissoon E, et al. An integrated cell atlas of the lung in health and disease. Nat Med. 2023;29: 1563–1577. doi:10.1038/s41591-023-02327-2

83. Luecken MD, Büttner M, Chaichoompu K, Danese A, Interlandi M, Mueller MF, et al. Benchmarking atlas-level data integration in single-cell genomics. Nat Methods. 2022;19: 41–50. doi:10.1038/s41592-021-01336-8

84. Tran HTN, Ang KS, Chevrier M, Zhang X, Lee NYS, Goh M, et al. A benchmark of batch-effect correction methods for single-cell RNA sequencing data. Genome Biol. 2020;21: 1–32. doi:10.1186/s13059-019-1850-9

85. Lotfollahi M, Naghipourfar M, Luecken MD, Khajavi M, Büttner M, Wagenstetter M, et al. Mapping single-cell data to reference atlases by transfer learning. Nat Biotechnol. 2022;40: 121–130. doi:10.1038/s41587-021-01001-7

86. Kimura M, Ohta T. On some principles governing molecular evolution. Proc Natl Acad Sci U S A. 1974;71: 2848–2852. doi:10.1073/pnas.71.7.2848

87. Altschul SF, Gish W, Miller W, Myers EW, Lipman DJ. Basic local alignment search tool. J Mol Biol. 1990;215: 403–410. doi:10.1016/S0022-2836(05)80360-2

88. Heinzinger M, Elnaggar A, Wang Y, Dallago C, Nechaev D, Matthes F, et al. Modeling aspects of the language of life through transfer-learning protein sequences. BMC Bioinformatics. 2019;20: 723. doi:10.1186/s12859-019-3220-8

89. Elnaggar A, Heinzinger M, Dallago C, Rehawi G, Wang Y, Jones L, et al. ProtTrans: Toward Understanding the Language of Life Through Self-Supervised Learning. IEEE Trans Pattern Anal Mach Intell. 2022;44: 7112–7127. doi:10.1109/TPAMI.2021.3095381

90. Kilinc M, Jia K, Jernigan RL. Protein Language Model Performs Efficient Homology Detection. bioRxiv. 2022. p. 2022.03.10.483778. doi:10.1101/2022.03.10.483778

91. Villegas-Morcillo A, Makrodimitris S, van Ham RCHJ, Gomez AM, Sanchez V, Reinders MJT. Unsupervised protein embeddings outperform hand-crafted sequence and structure features at predicting molecular function. Bioinformatics. 2020. doi:10.1093/bioinformatics/btaa701

92. Rives A, Meier J, Sercu T, Goyal S, Lin Z, Liu J, et al. Biological structure and function emerge from scaling unsupervised learning to 250 million protein sequences. Proc Natl Acad Sci U S A. 2021;118. doi:10.1073/pnas.2016239118

93. Tarashansky AJ, Musser JM, Khariton M, Li P, Arendt D, Quake SR, et al. Mapping single-cell atlases throughout Metazoa unravels cell type evolution. Elife. 2021;10. doi:10.7554/eLife.66747

94. Liu X, Shen Q, Zhang S. Cross-species cell-type assignment from single-cell RNA-seq data by a heterogeneous graph neural network. Genome Res. 2023;33: 96–111. doi:10.1101/gr.276868.122

95. Barash Y, Calarco JA, Gao W, Pan Q, Wang X, Shai O, et al. Deciphering the splicing code. Nature. 2010;465: 53–59. doi:10.1038/nature09000

96. Barash Y, Vaquero-Garcia J, González-Vallinas J, Xiong HY, Gao W, Lee LJ, et al. AVISPA: a web tool for the prediction and analysis of alternative splicing. Genome Biol. 2013;14: R114. doi:10.1186/gb-2013-14-10-r114

97. McLeay RC, Lesluyes T, Cuellar Partida G, Bailey TL. Genome-wide in silico prediction of gene expression. Bioinformatics. 2012;28: 2789–2796. doi:10.1093/bioinformatics/bts529

98. Agarwal V, Shendure J. Predicting mRNA Abundance Directly from Genomic Sequence Using Deep Convolutional Neural Networks. Cell Rep. 2020;31: 107663. doi:10.1016/j.celrep.2020.107663

99. Jaganathan K, Kyriazopoulou Panagiotopoulou S, McRae JF, Darbandi SF, Knowles D, Li YI, et al. Predicting Splicing from Primary Sequence with Deep Learning. Cell. 2019;176: 535–548.e24. doi:10.1016/j.cell.2018.12.015

100. Avsec Ž, Agarwal V, Visentin D, Ledsam JR, Grabska-Barwinska A, Taylor KR, et al. Effective gene expression prediction from sequence by integrating long-range interactions. Nat Methods. 2021;18: 1196–1203. doi:10.1038/s41592-021-01252-x

101. Kelley DR. Cross-species regulatory sequence activity prediction. Ma J, editor. PLoS Comput Biol. 2020;16: e1008050. doi:10.1371/journal.pcbi.1008050

102. Novakovsky G, Dexter N, Libbrecht MW, Wasserman WW, Mostafavi S. Obtaining genetics insights from deep learning via explainable artificial intelligence. Nat Rev Genet. 2023;24: 125–137. doi:10.1038/s41576-022-00532-2

103. Kelley DR, Snoek J, Rinn JL. Basset: learning the regulatory code of the accessible genome with deep convolutional neural networks. Genome Res. 2016;26: 990–999. doi:10.1101/gr.200535.115

104. Zhou J, Troyanskaya OG. Predicting effects of noncoding variants with deep learning-based sequence model. Nat Methods. 2015;12: 931–934. doi:10.1038/nmeth.3547

105. Shrikumar A, Tian K, Avsec Ž, Shcherbina A, Banerjee A, Sharmin M, et al. Technical Note on Transcription Factor Motif Discovery from Importance Scores (TF-MoDISco) version 0.5.6.5. arXiv [cs.LG]. 2018. Available: http://arxiv.org/abs/1811.00416

chapter 2

A comparison of automatic cell identification methods for single-cell RNA sequencing data

Tamim Abdelaal*, Lieke Michielsen*, Davy Cats, Dylan Hoogduin,
Hailiang Mei, Marcel J.T. Reinders, Ahmed Mahfouz

This chapter is published in: *Genome Biology* (2019) 20: 194, doi: 10.1186/s13059-019-1795-z.
Supplementary material is available online at:
https://genomebiology.biomedcentral.com/articles/10.1186/s13059-019-1795-z#Sec36

*Equal contribution

Single cell transcriptomics is rapidly advancing our understanding of the cellular composition of complex tissues and organisms. A major limitation in most analysis pipelines is the reliance on manual annotations to determine cell identities, which are time-consuming and irreproducible. The exponential growth in the number of cells and samples has prompted the adaptation and development of supervised classification methods for automatic cell identification. Here, we benchmarked 22 classification methods that automatically assign cell identities including single cell-specific and general-purpose classifiers. The performance of the methods was evaluated using 27 publicly available single cell RNA-sequencing datasets of different sizes, technologies, species, and levels of complexity. We used two experimental setups to evaluate the performance of each method for within dataset predictions (intra-dataset) and across datasets (inter-dataset) based on accuracy, percentage of unclassified cells, and computation time. We further evaluated the methods' sensitivity to the input features, number of cells per population, their performance across different annotation levels and datasets. We found that most classifiers performed well on a variety of datasets with decreased accuracy for complex datasets with overlapping classes or deep annotations. The general-purpose *SVM* classifier has overall the best performance across the different experiments. In conclusion, we present a comprehensive evaluation of automatic cell identification methods for single cell RNA-sequencing data. All the code used for the evaluation is available on GitHub (https://github.com/tabdelaal/scRNAseq_Benchmark). Additionally, we provide a Snakemake workflow to facilitate the benchmarking and to support extension of new methods and new datasets.

2.1 Background

Single-cell RNA-sequencing (scRNA-seq) provides unprecedented opportunities to identify and characterize the cellular composition of complex tissues. Rapid and continuous technological advances over the past decade has allowed scRNA-seq technologies to scale to thousands of cells per experiment [1]. A common analysis step in analyzing single cell data involves the identification of cell populations presented in a given dataset . This task is typically solved by unsupervised clustering of cells into groups based on the similarity of their gene expression profiles, followed by cell population annotation by assigning labels to each cluster. This approach proved very valuable in identifying novel cell populations and resulted in cellular maps of entire cell lineages, organs and even whole organisms [2–7]. However, the annotation step is cumbersome and time-consuming as it involves manual inspection of cluster-specific marker-genes. Additionally, manual annotations, which are often not based on standardized ontologies of cell labels, are not reproducible across different experiments within and across research groups. These caveats become even more pronounced as the number of cells and samples increases, preventing fast and reproducible annotations.

To overcome these challenges, a growing number of classification approaches are being adapted to automatically label cells in scRNA-seq experiments. scRNA-seq classification methods predict the identity of each cell by learning these identities from annotated training data (e.g. a reference atlas). scRNA-seq classification methods are relatively new compared to the plethora of methods addressing different computational aspects of single cell analysis (such as normalization, clustering, and trajectory inference). However, the number of classification methods is rapidly growing to address the aforementioned challenges [8,9].

While all scRNA-seq classification methods share a common goal, i.e. accurate annotation of cells, they differ in terms of their underlying algorithms and the incorporation of prior knowledge (e.g. cell type marker gene tables).

In contrast to the extensive evaluations of clustering, differential expression, and trajectory inference methods [10–12], there is currently one single attempt comparing methods to assign cell type labels to cell clusters [13]. The lack of a comprehensive comparison of scRNA-seq classification methods leaves users without indications as to which classification method best fits their problem. More importantly, a proper assessment of existing approaches in comparison to baseline methods can greatly benefit new developments in the field and prevent unnecessary complexity.

Here, we benchmarked 22 classification methods to automatically assign cell identities including single cell-specific and general-purpose classifiers. The methods were evaluated using 27 publicly available single cell RNA-sequencing datasets of different sizes, technologies, species, and complexity. The performance of the methods was evaluated based on their accuracy, percentage of unclassified cells, and computation time. We performed several experiments to cover different levels of challenge in the classification task, and to test specific features or tasks such as the feature selection, scalability and rejection experiments. We evaluated the classification performance through two experimental setups, 1) intra-dataset in which we applied 5-fold cross-validation within each dataset, and 2) inter-dataset involving across datasets comparisons. The inter-dataset comparison is more realistic and more practical, where a reference dataset (e.g. atlas) is used to train a classifier which can then be applied to identify cells in new unannotated datasets. However, in order to perform well across datasets, the classifier should also perform well using the intra-dataset setup on the reference dataset. The intra-dataset experiments, albeit artificial, provide an ideal scenario to evaluate different aspects of the classification process (e.g. feature selection, scalability and different annotation levels), regardless of the technical and biological variations across datasets. In general, most classifiers perform well across all datasets in both experimental setups (inter- and intra-dataset), including the general-purpose classifiers. In our experiments, incorporating prior knowledge in the form of marker-genes does not improve the performance. We observed large variation across different methods in the computation time and classification performance in response to changing the input features and the number of cells. Our results highlight the general-purpose support vector machine (*SVM*) classifier as the best performer overall.

2.2 Results

2.2.1 Benchmarking automatic cell identification methods (intra-dataset evaluation)

We benchmarked the performance and computation time of all 22 classifiers (Table 1) across 11 datasets used for intra-dataset evaluation (Table 2). Classifiers were divided into two categories: 1) supervised methods which require a training dataset labeled with the corresponding cell populations in order to train the classifier, or 2) prior-knowledge methods,

Table 1. Automatic cell identification methods included in this study.

Name	Version	Language	Underlying classifier	Prior knowledge	Rejection option	Ref.
Garnett	0.1.4	R	Generalized linear model	Yes	Yes	[14]
Moana	0.1.1	Python	SVM with linear kernel	Yes	No	[15]
DigitalCellSorter	Github version: e369a34	Python	Voting based on cell type markers	Yes	No	[16]
SCINA	1.1.0	R	Bimodal distr. fitting for marker-genes	Yes	No	[17]
scVI	0.3.0	Python	Neural Network	No	No	[18]
Cell-Blast	0.1.2	Python	Cell-to-cell similarity	No	Yes	[19]
ACTINN	GitHub version: 563bcc1	Python	Neural Network	No	No	[20]
LAmbDA	GitHub version: 3891d72	Python	Random Forest	No	No	[21]
Scmapcluster	1.5.1	R	Nearest median classifier	No	Yes	[22]
Scmapcell	1.5.1	R	kNN	No	Yes	[22]
scPred	0.0.0.9000	R	SVM with radial kernel	No	Yes	[23]
CHETAH	0.99.5	R	Correlation to training set	No	Yes	[24]
CaSTLe	Github version: 258b278	R	Random Forest	No	No	[25]
SingleR	0.2.2	R	Correlation to training set	No	No	[26]
scID	0.0.0.9000	R	LDA	No	Yes	[27]
singleCellNet	0.1.0	R	Random Forest	No	No	[28]
LDA	0.19.2	Python	LDA	No	No	[29]
NMC	0.19.2	Python	NMC	No	No	[29]
RF	0.19.2	Python	RF (50 trees)	No	No	[29]
SVM	0.19.2	Python	SVM (linear kernel)	No	No	[29]
SVM$_{rejection}$	0.19.2	Python	SVM (linear kernel)	No	Yes	[29]
kNN	0.19.2	Python	kNN (k = 9)	No	No	[29]

Table 2. Overview of the datasets used during this study.

Dataset	No. of cells	No. of genes	No. of cell populations (>10 cells)	Description	Protocol	Ref.
Baron (Mouse)^{a}	1,886	14,861	13 (9)	Mouse Pancreas	inDrop	[30]
Baron (Human)^{a,b}	8,569	17,499	14 (13)	Human Pancreas	inDrop	[30]
Muraro^{a,b}	2,122	18,915	9 (8)	Human Pancreas	CEL-Seq2	[31]
Segerstolpe^{a,b}	2,133	22,757	13 (9)	Human Pancreas	SMART-Seq2	[32]
Xin^{a,b}	1,449	33,889	4 (4)	Human Pancreas	SMARTer	[33]
CellBench 10X^{a,b}	3,803	11,778	5 (5)	Mixture of five human lung cancer cell lines	10X Chromium	[34]
CellBench CEL-Seq2^{a,b}	570	12,627	5 (5)	Mixture of five human lung cancer cell lines	CEL-Seq2	[34]

TM[a]	54,865	19,791	55 (55)	Whole Mus musculus	SMART-Seq2	[6]
AMB[a]	12,832	42,625	4/22/110 (3/16/92)	Primary mouse visual cortex	SMART-Seq v4	[35]
Zheng sorted[a]	20,000	21,952	10 (10)	FACS sorted PBMC	10X Chromium	[36]
Zheng 68K[a]	65,943	20,387	11 (11)	PBMC	10X Chromium	[36]
VISp[b] (Mouse)	12,832	42,625	3/36 (3/34)	Primary Visual Cortex	SMART-Seq v4	[35]
ALM[b] (Mouse)	8,758	42,461	3/37 (3/34)	Anterior Lateral Motor Area	SMART-Seq v4	[35]
MTG[b] (Human)	14,636	16,161	3/35 (3/34)	Middle Temporal Gyrus	SMART-Seq v4	[37]
PbmcBench pbmc1.10Xv2[b]	6,444	33,694	9 (9)	PBMC	10X version 2	[38]
PbmcBench pbmc1.10Xv3[b]	3,222	33,694	8 (8)	PBMC	10X version 3	[38]
PbmcBench pbmc1.CL[b]	253	33,694	7 (7)	PBMC	CEL-Seq2	[38]
PbmcBench pbmc1.DR[b]	3,222	33,694	9 (9)	PBMC	Drop-Seq	[38]
PbmcBench pbmc1.iD[b]	3,222	33,694	7 (7)	PBMC	inDrop	[38]
PbmcBench pbmc1.SM2[b]	253	33,694	6 (6)	PBMC	SMART-Seq2	[38]
PbmcBench pbmc1.SW[b]	3,176	33,694	7 (7)	PBMC	Seq-Well	[38]
PbmcBench pbmc2.10Xv[b]	3,362	33,694	9 (9)	PBMC	10X version 2	[38]
PbmcBench pbmc2.CL[b]	273	33,694	5 (5)	PBMC	CEL-Seq2	[38]
PbmcBench pbmc2.DR[b]	3,362	33,694	6 (6)	PBMC	Drop-Seq	[38]
PbmcBench pbmc2.iD[b]	3,362	33,694	9 (9)	PBMC	inDrop	[38]
PbmcBench pbmc2.SM2[b]	273	33,694	6 (6)	PBMC	SMART-Seq2	[38]
PbmcBench pbmc2.SW[b]	551	33,694	4 (4)	PBMC	Seq-Well	[38]

a: used for intra-dataset evaluation
b: used for inter-dataset evaluation

for which either a marker-genes file is required as an input or a pre-trained classifier for specific cell populations is provided.

The datasets used in this study vary in the number of cells, genes and cell populations (annotation level), in order to represent different levels of challenges in the classification task and to evaluate how each classifier performs in each case (Table 2). They include relatively typical sized scRNA-seq datasets (1,500–8,500 cells), such as the five pancreatic datasets (Baron Mouse and Human, Muraro, Segerstolpe and Xin), which include both mouse and human pancreatic cells and vary in the sequencing protocol used. The Allen Mouse Brain (AMB) dataset is used to evaluate how the classification performance changes when dealing with different levels of cell population annotation as the AMB dataset contains three levels of annotations for each cell (3, 16 or 92 cell populations), denoted as AMB3, AMB16, and AMB92. The Tabula Muris (TM) and Zheng 68K datasets represent relatively large scRNA-seq datasets (>50,000 cells), and are used to assess how well the classifiers scale with large datasets. For all previous datasets, cell populations were obtained through clustering. To assess how the classifiers perform when dealing with sorted populations, we included the CellBench dataset and the Zheng sorted dataset, representing sorted populations for lung cancer cell lines and PBMC, respectively. Including the Zheng sorted and Zheng 68K datasets, allows the benchmarking of four prior-knowledge classifiers, since the marker-genes files or pre-trained classifiers are available for the four classifiers for peripheral blood mononuclear cells (PBMCs).

2.2.2 All classifiers perform well in intra-dataset experiments

Generally, all classifiers perform well in the intra-dataset experiments, including the general-purpose classifiers (Figure 1). However, *Cell-BLAST* performs poorly for the Baron Mouse and Segerstople pancreatic datasets. Further, *scVI* has low performance on the deeply annotated datasets TM (55 cell populations) and AMB92 (92 cell populations), and *kNN* produces low performance for the Xin and AMB92 datasets.

For the pancreatic datasets, the best-performing classifiers are *SVM*, $SVM_{rejection}$, *scPred*, *scmapcell*, *scmapcluster*, *scVI*, *ACTINN*, *singleCellNet*, *LDA* and *NMC*. *SVM* is the only classifier to be in the top five list for all five pancreatic datasets, while *NMC*, for example, appears only in the top five list for the Xin dataset. The Xin dataset contains only four pancreatic cell types (alpha, beta, delta and gamma) making the classification task relatively easy for all classifiers, including *NMC*. Considering the median F1-score alone to judge the classification performance can be misleading since some classifiers incorporate a rejection option (e.g. $SVM_{rejection}$, *scmapcell*, *scPred*), by which a cell is assigned as 'unlabeled' if the classifier is not confident enough. For example, for the Baron Human dataset, the median F1-score for $SVM_{rejection}$, *scmapcell*, *scPred* and *SVM* is 0.991, 0.984, 0.981, and 0.980, respectively (Figure 1B). However, $SVM_{rejection}$, *scmapcell* and *scPred* assigned 1.5%, 4.2% and 10.8% of the cells, respectively, as unlabeled while *SVM* (without rejection) classified 100% of the cells with a median F1-score of 0.98. This shows an overall better performance for *SVM* and $SVM_{rejection}$, with higher performance and less unlabeled cells.

The CellBench 10X and CEL-Seq2 datasets represent an easy classification task, where the five sorted lung cancer cell lines are quite separable [34]. All classifiers have an almost perfect performance on both CellBench datasets (median F1-score ≈ 1).

For the TM dataset, the top five performing classifiers are $SVM_{rejection}$, *SVM*, *scmapcell*, *Cell-BLAST* and *scPred* with a median F1-score > 0.96, showing that these classifiers can perform well and scale to large scRNA-seq datasets with a deep level of annotation. Furthermore, *scmapcell* and *scPred* assigned 9.5% and 17.7% of the cells, respectively, as unlabeled, which shows a superior performance for $SVM_{rejection}$ and *SVM*, with a higher median F1-score and 2.9% and 0% unlabeled cells, respectively.

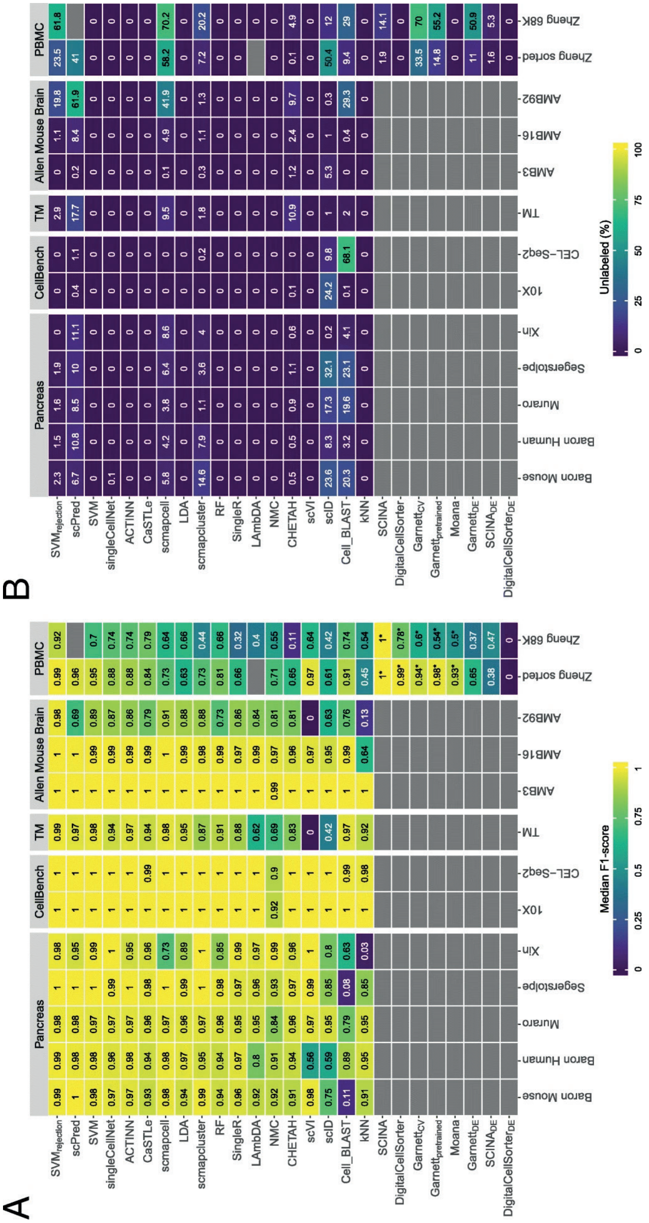


Figure 1. Performance comparison of supervised classifiers for cell identification using different scRNA-seq datasets. Heatmap of the **A)** median F1-scores and **B)** percentage of unlabeled cells across all cell populations per classifier (rows) per dataset (columns). Grey boxes indicate that the corresponding method could not be tested on the corresponding dataset. Classifiers are ordered based on the mean of the median F1-scores. Asterix (*) indicates that the prior-knowledge classifiers, *SCINA*, *DigitalCellSorter*, *Garnett*$_{CV}$, *Garnett*$_{pretrained}$, and *Moana*, could not be tested on all cell populations of the PBMC datasets. *SCINA*$_{DE}$, *Garnett*$_{DE}$, and *DigitalCellSorter*$_{DE}$ are the versions of *SCINA*, *Garnett*$_{CV}$, and *DigitalCellSorter* were the marker-genes are defined using differential expression from the training data. Different numbers of marker-genes, 5, 10, 15, and 20, were tested and the best result is shown here. *SCINA*, *Garnett*, and *DigitalCellSorter* produced the best result for the Zheng sorted dataset using 20, 15 and 5 markers, and for the Zheng 68K dataset using 10, 5 and 5 markers, respectively.

2.2.3 Performance evaluation across different annotation levels

We used the AMB dataset with its three different levels of annotations, to evaluate the classifiers' performance behavior with an increasing number of smaller cell populations within the same dataset. For AMB3, the classification task is relatively easy, differentiating between three major brain cell types (GABAergic, Glutamatergic and Non-Neuronal). All classifiers perform almost perfectly with a median F1-score > 0.99 (Figure 1A). For AMB16, the classification task becomes slightly more challenging and the performance of some classifiers drops, especially *kNN*. The top five classifiers are $SVM_{rejection}$, *scmapcell*, *scPred*, *SVM* and *ACTINN*, where $SVM_{rejection}$, *scmapcell* and *scPred* assigned 1.1%, 4.9% and 8.4% of the cells as unlabeled, respectively. For the deeply annotated AMB92 dataset, the performance of all classifiers drops further, specially for *kNN* and *scVI*, where the median F1-score is 0.130 and zero, respectively. The top five classifiers are $SVM_{rejection}$, *scmapcell*, *SVM*, *LDA*, and *scmapcluster*, with $SVM_{rejection}$ assigning less cells as unlabeled compared to *scmapcell* (19.8% vs 41.9%) and once more $SVM_{rejection}$ shows improved performance over *scmapcell* (median F1-score of 0.981 vs 0.906). These results show an overall superior performance for general-purpose classifiers ($SVM_{rejection}$, *SVM* and *LDA*) compared to other scRNA-seq specific classifiers across different levels of cell population annotation.

Instead of only looking at the median F1-score, we also evaluated the F1-score per cell population for each classifier (Figure S1). We confirmed previous conclusions, *kNN* performance drops with deep annotations which include smaller cell populations (Figure S1B-C), and *scVI* poorly performs on the deeply annotated AMB92 dataset. Additionally, we observed that some cell populations are much harder to classify compared to other populations. For example, most classifiers had a low performance on the Serpinf1 cells in the AMB16 dataset.

2.2.4 Incorporating marker-genes does not improve intra-dataset performance on PBMC data

For the two PBMC datasets (Zheng 68K and Zheng sorted), the prior-knowledge classifiers *Garnett*, *Moana*, *DigitalCellSorter* and *SCINA* could be evaluated and benchmarked with the rest of the classifiers. Although the best performing classifier on Zheng 68K is *SCINA* with a median F1-score of 0.998, this performance is based only on 3, out of 11, cell populations (Monocytes, B cells and NK cells) for which marker-genes are provided. Table S1 summarizes which PBMC cell populations can be classified by the prior-knowledge methods. Interestingly, none of the prior-knowledge methods showed superior performance compared to other classifiers, despite the advantage these classifiers have over other classifiers given they are tested on fewer cell populations due to the limited availability of marker-genes. *Garnett, Moana,* and *DigitalCellSorter,* could be tested on seven, seven, and five cell populations respectively (Table S1). Beside *SCINA*, the top classifiers for the Zheng 68K dataset are *CaSTLe*, *ACTINN*, *singleCellNet* and *SVM*. $SVM_{rejection}$ and *Cell-BLAST* show high performance, at the expense of high rejection rate of 61.8% and 29%, respectively (Figure 1). Moreover, *scPred* failed when tested on the Zheng 68K dataset. Generally, all classifiers show relatively lower performance on the Zheng 68K dataset compared to other datasets, as the Zheng 68K dataset contains 11 immune cell populations which are harder to differentiate, particularly the T cell compartment (6 out of 11 cell populations). This difficulty of separating these populations was previously noted in the original study [36]. Also, the confusion matrices for *CaSTLe*, *ACTINN*, *singleCellNet* and *SVM* clearly indicate the high similarity between cell populations, such as 1) monocytes with dendritic cells, 2) the two CD8+ T populations, and 3) the four CD4+ T populations (Figure S2).

The classification of the Zheng sorted dataset is relatively easier compared to the Zheng 68K dataset, as almost all classifiers show improved performance (Figure 1), with the exception that *LAmbDA* failed while being tested on the Zheng sorted dataset. The prior-knowledge methods show high performance (median F1-score > 0.93), which is still comparable to other classifiers such as $SVM_{rejection}$, *scVI*, *scPred* and *SVM*. Yet, the supervised classifiers do not require any marker-genes, and they can predict more (all) cell populations.

2.2.5 The performance of prior-knowledge classifiers strongly depends on the selected marker-genes

Some prior-knowledge classifiers, *SCINA*, *DigitalCellSorter* and $Garnett_{CV}$, used marker-genes to classify the cells. For the PBMC datasets, the number of marker-genes per cell population varies across classifiers (2-161 markers) and the marker-genes show very little overlap. Only one B cell marker gene, CD79A, is shared by all classifiers while none of the marker-genes for the other cell populations is shared by the three classifiers. We analyzed the effect of the number of marker-genes, mean expression, dropout rate, and the specificity of each marker gene (beta score, see Methods), on the performance of the classifier (Figure S3). The dropout rate and marker specificity (beta-score) are strongly correlated with the median F1-score, highlighting that the performance does not only depend on biological knowledge, but also on technical factors.

The difference between the marker-genes used by each method underscores the challenge of marker-genes selection, especially for smaller cell populations. Moreover, public databases of cell type markers (e.g. PanglaoDB [39] and CellMarker [40]) often provide different markers for the same population. For example, CellMarker provides 33 marker-genes for B cells, while PanglaoDB provides 110 markers, with only 11 marker-genes overlap between the two databases.

Given the differences between “expert-defined” markers and the correlation of classification performance and technical dataset-specific features (e.g. dropout rate), we tested if the performance of prior-knowledge methods can be improved by automatically selecting marker-genes based on differential expression. Through the cross-validation scheme, we used the training folds to select the marker-genes of each cell population based on differential expression (see Methods) and later used these markers to evaluate the classifiers’ performance on the testing fold. We tested this approach on the two PBMC datasets, Zheng sorted and Zheng 68K for different numbers of marker-genes (5, 10, 15, and 20 markers). In Figure 1, the best result across the number of markers for $SCINA_{DE}$, $Garnett_{DE}$, and $DigitalCellSorter_{DE}$ are shown.

The median F1-score obtained using the differential expression-defined markers is significantly lower compared to the original versions of classifiers using the markers defined by the authors. This lower performance is in part due to the low performance on challenging populations, such as subpopulations of CD4+ and CD8+ T cell populations (F1-score ≤ 0.68) (Figure S4). These challenging populations are not identified by the original classifiers since the markers provided by the authors only considered annotations at a higher level (Table S1). For example, the median F1-score of $SCINA_{DE}$ on Zheng sorted is 0.38, compared to a median F1-score of 1.0 for *SCINA* (using the original markers defined by the authors). However, *SCINA* only considers three cell populations: CD14+ monocytes, CD56+ NK cells, and CD19+ B cells. If we only consider these cell populations for $SCINA_{DE}$, this results in a median F1-score of 0.95.

We observed that the optimal number of marker-genes varies per classifier and dataset. For the Zheng sorted dataset the optimal number of markers is 5, 15, and 20 for $DigitalCellSorter_{DE}$, $Garnett_{DE}$, and, $SCINA_{DE}$ respectively, while for Zheng 68K this is 5, 5, and 10. All together, these results illustrate the dependence of the classification performance on the careful selection of marker genes which is evidently a challenging task.

2.2.6 Classification performance depends on dataset complexity

A major aspect affecting the classification performance is the complexity of the dataset at hand. We described the complexity of each dataset in terms of the pairwise similarity between cell populations (see Methods) and compared the complexity to the performance of the classifiers and the number of cell populations in a dataset (Figure 2). When the complexity and/or the number of cell populations of the dataset increases, the performance generally decreases. The performance of all classifiers is relatively low on the Zheng 68K dataset, which can be explained by the high pairwise correlations between the mean expression profiles of each cell population (Figure S5). These correlations are significantly lower for the TM and AMB92 datasets, justifying the higher performance of the classifiers on these two datasets (Figure S6-7). While both TM and AMB92 have more cell populations (55 and 92, respectively) compared to Zheng 68K (11 populations), these populations are less correlated to one another, making the task easier for all the classifiers.

2.2.7 Evaluation across datasets

While evaluating the classification performance within a dataset (intra-dataset) is important, the realistic scenario in which a classifier is useful requires cross-dataset (i.e. inter-dataset) classification. We used 22 datasets (Table 2) to test the classifiers' ability to predict cell identities in a dataset that was not used for training. First, we tested the classifiers' performance across different sequencing protocols, applied to the same samples within the same lab using the two CellBench datasets. We evaluated the classification performance when training on one protocol and testing on the other. Similar to the intra-dataset evaluation result, all classifiers performed well in this case (Figure S8).

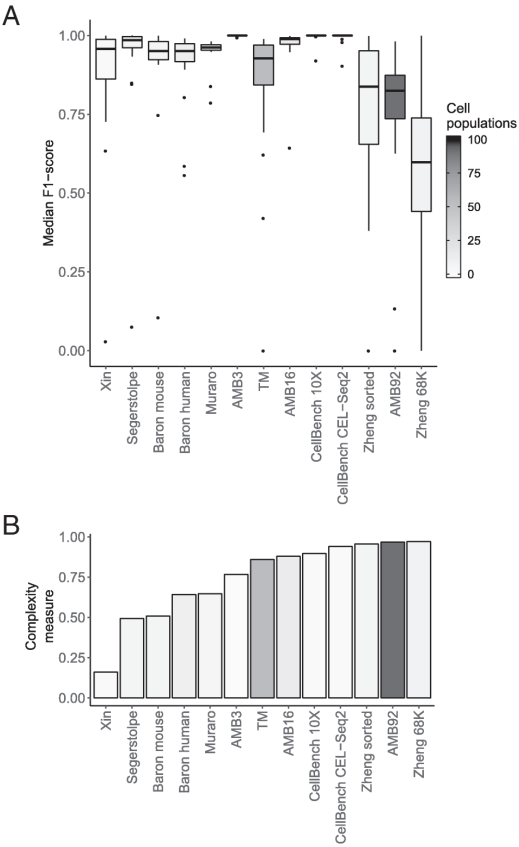


Figure 2. Complexity of the datasets compared to the performance of the classifiers. A) Boxplots of the median F1-scores of all classifiers for each dataset used during the intra-dataset evaluation. **B)** Barplots describing the complexity of the datasets (see Methods). Datasets are ordered based on complexity. Box- and barplots are colored according to the number of cell populations in each dataset.

Second, we tested the classification performance on the PbmcBench datasets, which represent a more extensive protocol comparison. PbmcBench consists of two samples (pbmc1 and pbmc2), sequenced using seven different protocols (Table 2) with the exception that 10Xv3 was not applied to the pbmc2 sample. We used the pbmc1 datasets to evaluate the classification performance of all pairwise train-test combinations between the seven protocols (42 experiments, see Methods). Moreover, we extended the evaluation to include comparisons across different samples for the same protocol, using pbmc1 and pbmc2 (6 experiments, see Methods). All 48 experiments results are summarized in Figure 3. Overall, several classifiers performed well including *SCINA*$_{DE}$ using 20 marker-genes, *singleCellNet*, *scmapcell*, *scID* and *SVM*, with an average median F1-score > 0.75 across all 48 experiments (Figure 3A, S9A). *SCINA*$_{DE}$, *Garnett*$_{DE}$, and *DigitalCellSorter*$_{DE}$ were tested using 5, 10, 15 and 20 marker-genes, Figure 3A shows the best result for each classifier, where *SCINA*$_{DE}$ and *Garnett*$_{DE}$ performed best using 20 and 5 marker-genes, respectively, while *DigitalCellSorter*$_{DE}$ had a median F1-score of zero during all experiments using all different numbers of marker-genes. *DigitalCellSorter*$_{DE}$ could only identify B-cells in the test sets, usually with an F1-score between 0.8 and 1.0, while the F1-score for all other cell populations was zero.

We also tested the prior-knowledge classifiers on all 13 PbmcBench datasets. The prior-knowledge classifiers showed lower performance compared to other classifiers (average

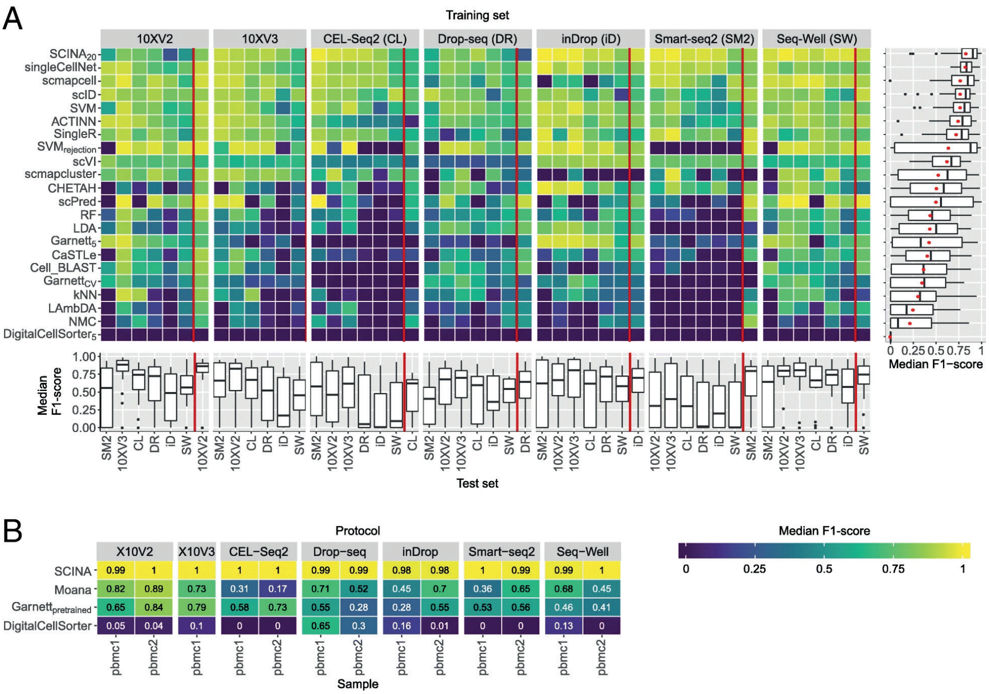


Figure 3. Classification performance across the PbmcBench datasets. A) Heatmap showing the median F1-scores of the supervised classifiers for all train-test pairwise combination across different protocols. The training set is indicated in the grey box on top of the heatmap, the test set is indicated using the column labels below. Results showed to the left of the red line represent the comparison between different protocol using sample pbmc1. Sample pbmc2 was used as test set then. Results showed to the right of the red line represent the comparison between different samples using the same protocol, with pbmc 1 used for training and pbmc2 used for testing. Boxplots on the right side of the heatmap summarize the performance of each classifier across all experiments. The mean of the median F1-scores, also used to order the classifiers, is indicated in the boxplots using a red dot. Boxplots underneath the heatmap summarize the performance of the classifiers per experiment. For SCINA_{DE}, $\mathit{Garnett}_{DE}$, and $\mathit{DigitalCellSorter}_{DE}$ different numbers of marker-genes were tested. Only the best result is shown here. **B)** Median F1-score of the prior-knowledge classifiers on both samples of the different protocols. The protocol is indicated in the grey box on top of the heatmap, the sample is indicated with the labels below. Classifiers are ordered based on their mean performance across all datasets.

median F1-score < 0.6), with the exception of *SCINA* which was only tested on three cell populations (Figure 3B, S9B). These results are inline with our previous conclusions from the Zheng sorted and Zheng 68K datasets in the intra-dataset evaluation.

Comparing the performance of the classifiers across the different protocols, we observed a higher performance for all classifiers for specific pairs of protocols. For example, all classifiers performed well when trained on 10Xv2 and tested on 10Xv3, and vice versa. On the other hand, other pairs of protocols had good performance only in one direction, training on Seq-Well produced good predictions on 10Xv3, but not the other way around. Compared to all other protocols, the performance of all classifiers was low when they were either trained or tested on Smart-seq2 data. This can, in part, be due to the fact that Smart-seq2 data does not contain Unique Molecular Identifier (UMI), in contrast to all other protocols.

We also tested the classification performance using the three brain datasets, VISp, ALM and MTG (Table 2), which allowed us to compare performances across species (mouse and human) as well as single-cell RNA-seq (used in VISp and ALM) versus single-nucleus RNA-seq (used for MTG). We tested all possible train-test combinations for both levels of annotation, three major brain cell types (inhibitory neurons, excitatory neurons and non-neuronal cells) and the deeper annotation level with 34 cell populations (18 experiments, see Methods). Prediction of the three major cell types was easy, where almost all classifiers showed high performance (Figure 4A) with some exceptions. For example, *scPred* failed the classification task completely when testing on the MTG dataset, producing 100% unlabeled cells (Figure S10A). Predicting the 34 cell populations turned out to be a more challenging task, especially when the MTG human dataset is included either as training or testing data, resulting in significantly lower performance across all classifiers (Figure 4B). Across all nine experiments at the deeper annotation, the top performing classifiers were *SVM*, *ACTINN*, *singleCellNet*, *SingleR* and *LAmbDA*, with almost 0% unlabeled cells (Figure S10B).

Finally, to evaluate the classification performance across different protocols and different labs, we used the four human pancreatic datasets: Baron Human, Muraro, Segerstople and Xin. We tested four combinations by training on three datasets and test on one dataset, in which case the classification performance can be affected by batch differences between datasets. We evaluated the performance of the classifiers when trained using the original data as well as aligned data using the mutual nearest neighbour (MNN) method [41]. Figure S11 shows UMAPs [42] of the combined dataset before and after alignment, demonstrating better grouping of pancreatic cell types after alignment.

For the original (unaligned) data, the best performing classifiers across all four experiments are *scVI, SVM*, *ACTINN, scmapcell* and *SingleR* (Figure 5A, S12A). For the aligned data, the best performing classifiers are *kNN*, $SVM_{rejection}$, *singleCellNet, SVM* and *NMC* (Figure 5B, S12B). Some classifiers benefit from aligning datasets such as $SVM_{rejection}$, *kNN*, *NMC* and *singleCellNet*, resulting in higher median F1-scores (Figure 5). On the other hand, some other classifiers failed the classification task completely, such as *scmapcell* which labels all cells as unlabeled. Some other classifiers failed to run over the aligned datasets, such as *ACTINN*, *scVI*, *Cell-BLAST*, *scID*, *scmapcluster* and *scPred*. These classifiers work only with positive gene expression data, while the aligned datasets contains positive and negative gene expression values.

2.2.8 Rejection option evaluation

Classifiers developed for scRNA-seq data often incorporate a rejection option to identify cell populations in the test set that were not seen during training. These populations cannot be predicted correctly and therefore should remain unassigned. To test whether the classifiers indeed leave these unseen populations unlabeled, we applied two different experiments using negative controls of different tissues and using unseen populations of the same tissue.

First, the classifiers were trained on a data set from one tissue (e.g. pancreas) and used to predict cell populations of a completely different tissue (e.g. brain) [22]. The methods

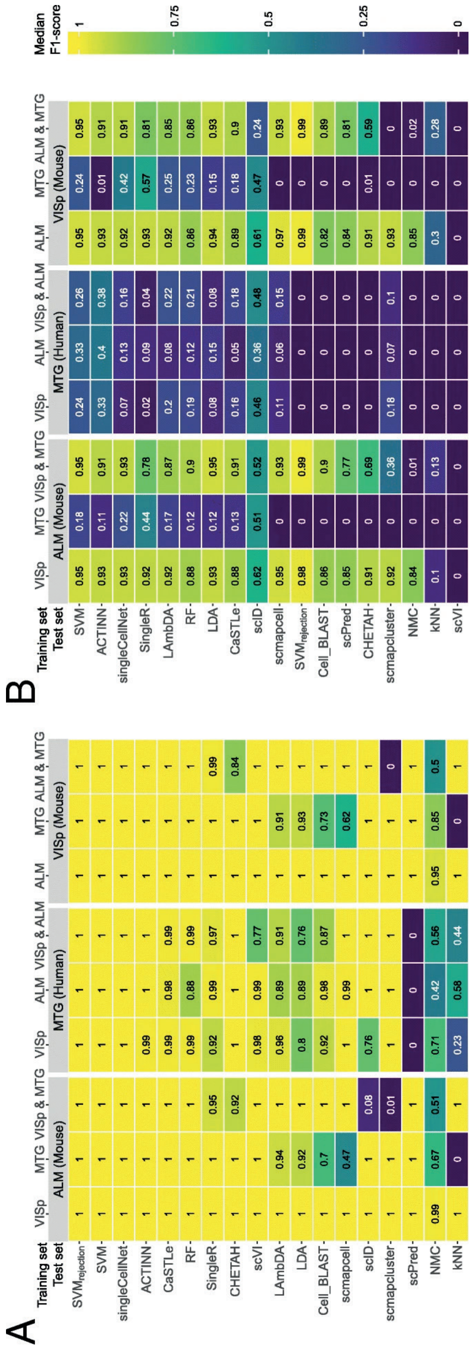
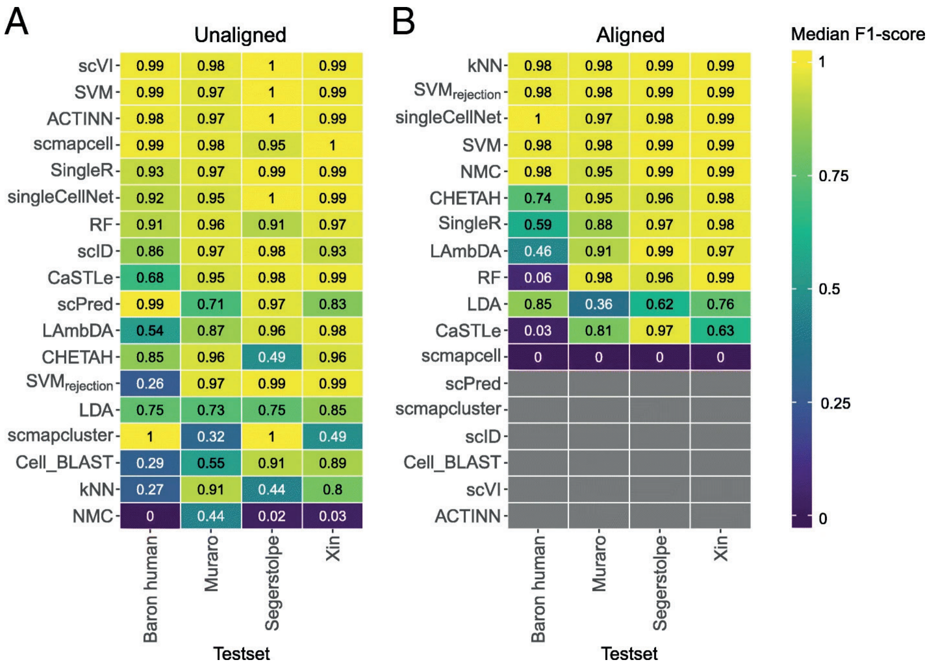


Figure 4. Classification performance across brain datasets. Heatmaps show the median F1-scores of the supervised classifiers when tested on **A)** major lineage annotation with three cell populations, and **B)** deeper level of annotation with 34 cell populations. The training set(s) are indicated using the column labels on top of the heatmap. The test set is indicated in the grey box. In each heatmap the classifiers are ordered based on their mean performance across all experiments.



	Baron human	Muraro	Segerstolpe	Xin
scVI	0.99	0.98	1	0.99
SVM	0.99	0.97	1	0.99
ACTINN	0.98	0.97	1	0.99
scmapcell	0.99	0.98	0.95	1
SingleR	0.93	0.97	0.99	0.99
singleCellNet	0.92	0.95	1	0.99
RF	0.91	0.96	0.91	0.97
scID	0.86	0.97	0.98	0.93
CaSTLe	0.68	0.95	0.98	0.99
scPred	0.99	0.71	0.97	0.83
LAmbDA	0.54	0.87	0.96	0.98
CHETAH	0.85	0.96	0.49	0.96
$SVM_{rejection}$	0.26	0.97	0.99	0.99
LDA	0.75	0.73	0.75	0.85
scmapcluster	1	0.32	1	0.49
Cell_BLAST	0.29	0.55	0.91	0.89
kNN	0.27	0.91	0.44	0.8
NMC	0	0.44	0.02	0.03

	Baron human	Muraro	Segerstolpe	Xin
kNN	0.98	0.98	0.99	0.99
$SVM_{rejection}$	0.98	0.98	0.99	0.99
singleCellNet	1	0.97	0.98	0.99
SVM	0.98	0.98	0.99	0.99
NMC	0.98	0.95	0.99	0.99
CHETAH	0.74	0.95	0.96	0.98
SingleR	0.59	0.88	0.97	0.98
LAmbDA	0.46	0.91	0.99	0.97
RF	0.06	0.98	0.96	0.99
LDA	0.85	0.36	0.62	0.76
CaSTLe	0.03	0.81	0.97	0.63
scmapcell	0	0	0	0
scPred				
scmapcluster				
scID				
Cell_BLAST				
scVI				
ACTINN				

Figure 5. Classification performance across pancreatic datasets. Heatmaps showing the median F1-score for each classifier for the **A)** unaligned and **B)** aligned datasets. The column labels indicate which of the four datasets was used as a test set, in which case the other three datasets were used as training. Grey boxes indicate that the corresponding method could not be tested on the corresponding dataset. In each heatmap, the classifiers are ordered based on their mean performance across all experiments.

should thus reject all (100%) of the cells in the test dataset. We carried out four different negative control experiments (see Methods, Figure 6A). *scmapcluster* and *scPred* have an almost perfect score for all four combinations, rejecting close 100% of the cells. Other top performing methods for this task, *SVM*$_{rejection}$ and *scmapcell,* failed when trained on mouse pancreatic data and tested on mouse brain data. All labeled cells of the AMB16 dataset are predicted to be beta cells in this case. The prior-knowledge classifiers, *SCINA, Garnett*$_{pretrained}$, and *DigitalCellSorter*, could only be tested on the Baron Human pancreatic dataset. *Garnett*$_{CV}$ could, on top of that, also be trained on the Baron Human dataset and tested on the Zheng 68K dataset. During the training phase, *Garnett*$_{CV}$ tries to find representative cells for the cell populations described in the marker-genes file. Being trained on Baron Human using the PBMC marker-genes file, it should not be able to find any representatives and therefore all cells in the Zheng 68K dataset should be unassigned. Surprisingly, *Garnett*$_{CV}$ still finds representatives for PBMC cells in the pancreatic data and thus the cells in the test set are labeled. However, being trained on the PBMC dataset and tested on the pancreatic dataset, it does have a perfect performance.

To test the rejection option in more realistic and challenging scenario, we trained the classifiers on some cell populations from one dataset, and used the held out cell populations in the test set (see Methods). Since the cell populations in the test set were not seen during training, they should remain unlabeled. Here, the difficulty of the task was gradually increased

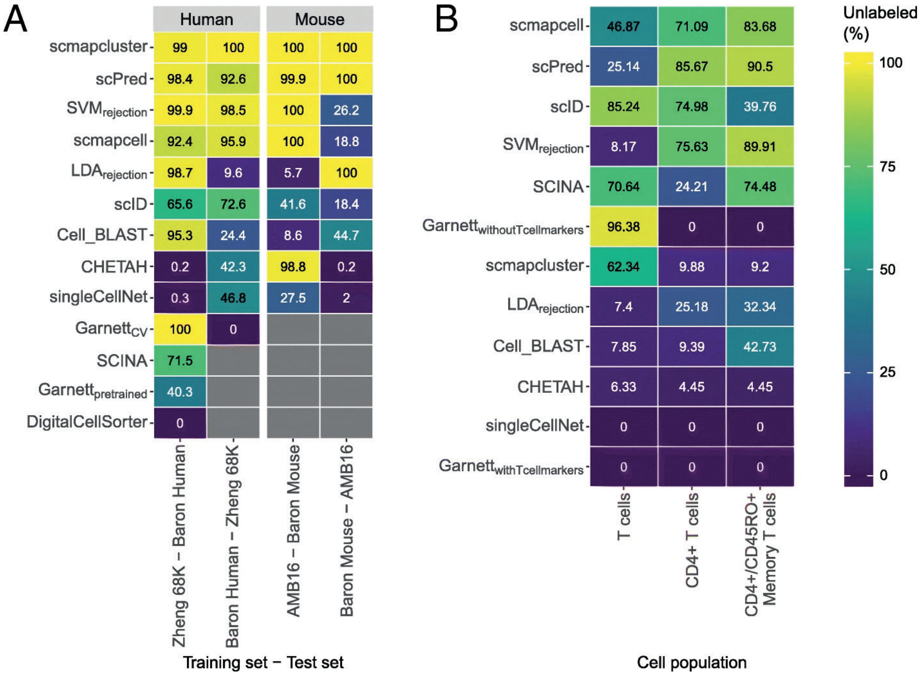


Figure 6. Performance of the classifiers during the rejection experiments. A) Percentage of unlabeled cells during the negative control experiment for all the classifiers with a rejection option. The prior-knowledge classifiers could not be tested on all datasets, this is indicated with a grey box. The species of the dataset is indicated in the grey box on top. Column labels indicate which datasets are used for training and testing respectively. **B)** Percentage of unlabeled cells for all classifiers with a rejection option when a cell population was removed from the training set. Column labels indicate which cell population was removed. This cell population was used as a test set. In both **A)** and **B)** the classifiers are sorted based on their mean performance across all experiments.

(Table S3). First all the T-cells were removed from the training set. Next, only the CD4+ T cells were removed. Finally, only CD4+/CD45RO+ Memory T cells, a subpopulation of the CD4+ T cells, were removed. The top performing methods for this task are: *scmapcell, scPred, scID,* $SVM_{rejection}$, and *SCINA* (Figure 6B). We expected that rejecting T cells would be a relatively easy task as they are quite distinct from all other cell populations in the dataset. It should thus be comparable to the negative control experiment. Rejecting CD4+/CD45RO+ Memory T cells, on the other hand, would be more difficult as they could easily be confused with all other subpopulations of CD4+ T cells. Surprisingly, almost all classifiers, except for *scID* and *scmapcluster,* show the opposite.

To better understand this unexpected performance we analyzed the labels assigned by $SVM_{rejection}$. In the first task (T cells removed from the training set), $SVM_{rejection}$ labels almost all T cells as B cells. This can be explained by the fact that $SVM_{rejection}$, and most classifiers for that matter, rely on classification posterior probabilities to assign labels but ignores the actual similarity between each cell and the assigned population. In task two (CD4+ T cells were removed), there were two subpopulations of CD8+ T cells in the training set. In that case, two cell populations are equally similar to the cells in the test set, resulting in low posterior probabilities for both classes and thus the cells in the test set remain unlabeled. If one of these CD8+ T cell populations was removed from the training set, only 10.53% instead of 75.57% of the CD4+ T cells were assigned as unlabeled by $SVM_{rejection}$. All together, our results indicate that despite the importance of incorporating a rejection option in cell identity classifiers, the implementation of this rejection option remains challenging.

2.2.9 Performance sensitivity to the input features

During the intra-datasets cross-validation experiment described earlier, we used all features (genes) as input to the classifiers. However, some classifiers suffer from overtraining when too many features are used. Therefore, we tested the effect of feature selection on the performance of the classifiers. While different strategies for feature selection in scRNA-seq classification experiments exist, selecting genes with a higher number of dropouts compared to the expected number of dropouts has been shown to outperform other methods [22,43]. We selected subsets of features from the TM dataset using the dropout method. In the experiments, we used the top: 100, 200, 500, 1000, 2000, 5000, and 19791 (all) genes. Some classifiers include a built-in feature selection method which is used by default. To ensure that all methods use the same set of features, the built-in feature selection was turned off during these experiments.

Some methods are clearly overtrained when the number of features increases (Figure 7A). For example, *scmapcell* shows the highest median F1-score when using less features and the performance drops when the number of features increases. On the other hand, the performance of other classifiers, such as *SVM*, keeps improving when the number of features increases. These results indicate that the optimal number of features is different for each classifier.

Looking at the median F1-score, there are several methods with a high maximal performance. *Cell-BLAST, ACTINN, scmapcell*, *scPred*, $SVM_{rejection}$ and *SVM* all have a median F1-score higher than 0.97 for one or more of the feature sets. Some of these well-performing methods, however, leave many cells unlabeled. *scmapcell* and *scPred*, for instance, yield a maximum median F1-score of 0.976 and 0.982 respectively, but 10.7% and 15.1% of the cells are assigned as unlabeled (Figure 7B). On the other hand, $SVM_{rejection}$ has the highest median F1-score (0.991) overall with only 2.9% unlabeled. Of the top performing classifiers only *ACTINN* and *SVM* label all the cells. Overall *SVM* shows the third highest performance with a score of 0.979.

2.2.10 Scalability: performance sensitivity to the number of cells

scRNA-seq datasets vary significantly across studies in terms of the number of cells analyzed. To test the influence of the size of the dataset on the performance of the classifier, we downsampled the TM dataset in a stratified way (i.e. preserving population frequencies) to 1, 5, 10, 20, 50, and 100% of the original number of 45,469 cells (see Methods) and compared the performance of the methods (Figure 7C, D). Using less than 500 cells in the dataset, most classifiers have a relatively high performance. Only *scID, LAmbDA, CaSTLe,* and *Cell-BLAST,*

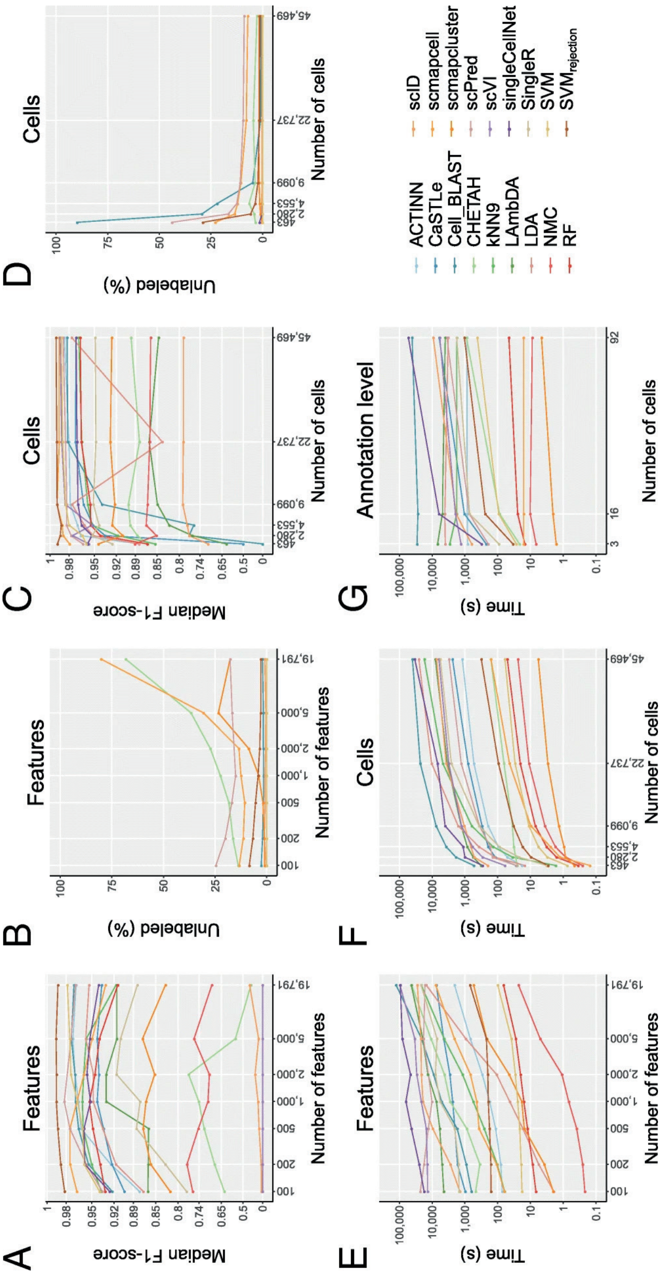


Figure 7. Classification performance and computation time evaluation across different numbers of features, cells, and annotation levels. Line plots show **A)** the median F1-score, **B)** percentage of unlabeled cells, and **E)** computation time of each classifier applied to the TM dataset with the top 100, 200, 500, 1000, 2000, 5000, and 19791 (all) genes as input feature sets. Genes were ranked based on dropout-based feature selection. **C)** The median F1-score, **D)** percentage of unlabeled cells, and **F)** computation time of each classifier applied to the downsampled TM datasets containing 463, 2,280, 4,553, 9,099, 22,737, and 45,469 (all) cells. **G)** The computation time of each classifier is plotted against the number of cell populations. Note that the y-axis is 100^x scaled in **A,C** and log-scaled in **E-G**. The x-axis is log-scaled in **A-F**

have a median F1-score below 0.85. Surprisingly, $SVM_{rejection}$ has almost the same median F1-score when using 1% of the data as when using all data (0.993 and 0.994 respectively). It must be noted here, however, that the percentage of unlabeled cells decreases significantly (from 28.9% to 1.3%). Overall, the performance of all classifiers stabilized when tested on ≥ 20% (9,099 cells) of the original data.

2.2.11 Running time evaluation

To compare the runtimes of the methods and see how they scale when the number of cells increases, we compared the number of cells in each dataset with the computation time of the classifiers (Figure S13). Overall, big differences in the computation time can be observed when comparing the different methods. *SingleR* showed the highest computation time overall. Running *SingleR* on the Zheng 68K dataset took more than 39 hours, while *scmapcluster* was finished within 10 seconds on this dataset. Some of the methods have a high runtime for the small datasets. On the smallest dataset, Xin, all classifiers have a computation time <5 minutes, with most classifiers finishing within 60 seconds. *Cell-BLAST*, however, takes more than 75 minutes. In general, all methods show an increase in computation time when the number of cells increase. However, when comparing the second largest, TM, and largest, Zheng 68K, dataset, not all methods show an increase in computation time. Despite the increase in the number of cells between the two datasets, *CaSTLe*, *CHETAH*, and *SingleR*, have a decreasing computation time. A possible explanation could be that the runtime of these methods also depends on the number of genes or the number of cell populations in the dataset. To evaluate the run time of the methods properly, we therefore investigated the effect of the number of cells, features, and cell populations separately (Figure 7E-G).

To assess the effect of the number of genes on the computation time, we compared the computation time of the methods during the feature selection experiment (Figure 7E). Most methods scale linearly with the number of genes. However, *LDA* does not scale very well when the number of genes increases. If the number of features is higher than the number of cells, the complexity of *LDA* is O(g^3), where g is the number of genes [44].

The effect of the number of cells on the timing showed that all methods increase in computation time when the number of cells increases (Figure 7F). The differences in runtime on the largest dataset are larger. *scmapcluster,* for instance, takes five seconds to finish, while *Cell-BLAST* takes more than 11 hours.

Finally, to evaluate the effect of the number of cell populations, the runtime of the methods on the AMB3, AMB16, and AMB92 datasets were compared (Figure 7G). For most methods this shows an increase in runtime when the number of cell populations increases, specially *singleCellNet*. For other methods, such as *ACTINN* and *scmapcell*, the runtime remains constant. Five classifiers, *scmapcell*, *scmapcluster*, *SVM*, *RF*, and *NMC*, have a computation time below six minutes on all the datasets.

2.3 Discussion

In this study, we evaluated the performance of 22 different methods for automatic cell identification using 27 scRNA-seq datasets. We performed several experiments to cover different levels of challenges in the classification task, and to test specific aspects of the classifiers such as the feature selection, scalability and rejection experiments. We summarize our findings across the different experiments (Figure 8) and provide a detailed summary of which dataset was used for each experiment (Table S4). This overview can be used as a user-guide to choose the most appropriate classifier depending on the experimental setup at hand. Overall, several classifiers performed accurately across different datasets and experiments, particularly: *SVM*$_{rejection}$, *SVM*, *singleCellNet*, *scmapcell*, *scPred*, *ACTINN* and *scVI*. We observed relatively lower performance for the inter-dataset setup, likely due to the technical and biological differences between datasets, compared to the intra-dataset setup. *SVM*$_{rejection}$, *SVM* and *singleCellNet* performed well for both setups, while *scPred* and *scmapcell* performed better in the intra-dataset setup, and *scVI* and *ACTINN* had better performance in the inter-dataset setup (Figure 8). Of note, we evaluated all classifiers using the default settings. While adjusting these settings for a specific dataset might improve the performances it increases the risk of overtraining.

Considering all three evaluation metrics (median F1-score, percentage of unlabeled cells and computation time), *SVM*$_{rejection}$ and *SVM* are overall the best performing classifiers for the scRNA-seq datasets used. Although *SVM* has a shorter computation time, the high accuracy of the rejection option of *SVM*$_{rejection}$, which allows flagging new cells and assigning them as unlabeled, results in an improved performance compared to *SVM*. Our results show that *SVM*$_{rejection}$ and *SVM* scale well to large datasets as well as deep annotation levels. In addition, they did not suffer from the large number of features (genes) present in the data, producing the highest performance on the TM dataset using all genes, due to the incorporated L2-regularization. The comparable or higher overall performance of a general-purpose classier such as *SVM* warrants caution when designing scRNA-seq specific classifiers that they do not introduce unnecessary complexity. For example, deep learning methods, such as *ACTINN* and *scVI*, showed overall lower performance compared to *SVM,* supporting recent observations by Köhler *et al.* [45].

scPred (which is based on an SVM with radial kernel), *LDA*, *ACTINN*, and *singleCellNet* performed well on most datasets, yet the computation time is long for large datasets*. singleCellNet* also becomes slower with a large number of cell populations. In addition, in some cases, *scPred* and *scmapcell/cluster* reject higher proportions of cells as unlabeled compared to *SVM*$_{rejection}$, without a substantial improvement in accuracy. In general, incorporating a rejection option with classification is a good practice to allow the detection of potentially novel cell populations (not present in the training data) and improve the performance for the classified cells with high confidence. However, for the datasets used in this study, the performance of classifiers with rejection option, except for *SVM*$_{rejection}$, did not show substantial improvement compared to other classifiers. Furthermore, our results indicate that designing a proper rejection option can be challenging for complex datasets (e.g. PBMC) and that relying on the posterior probabilities alone might not yield optimal results.

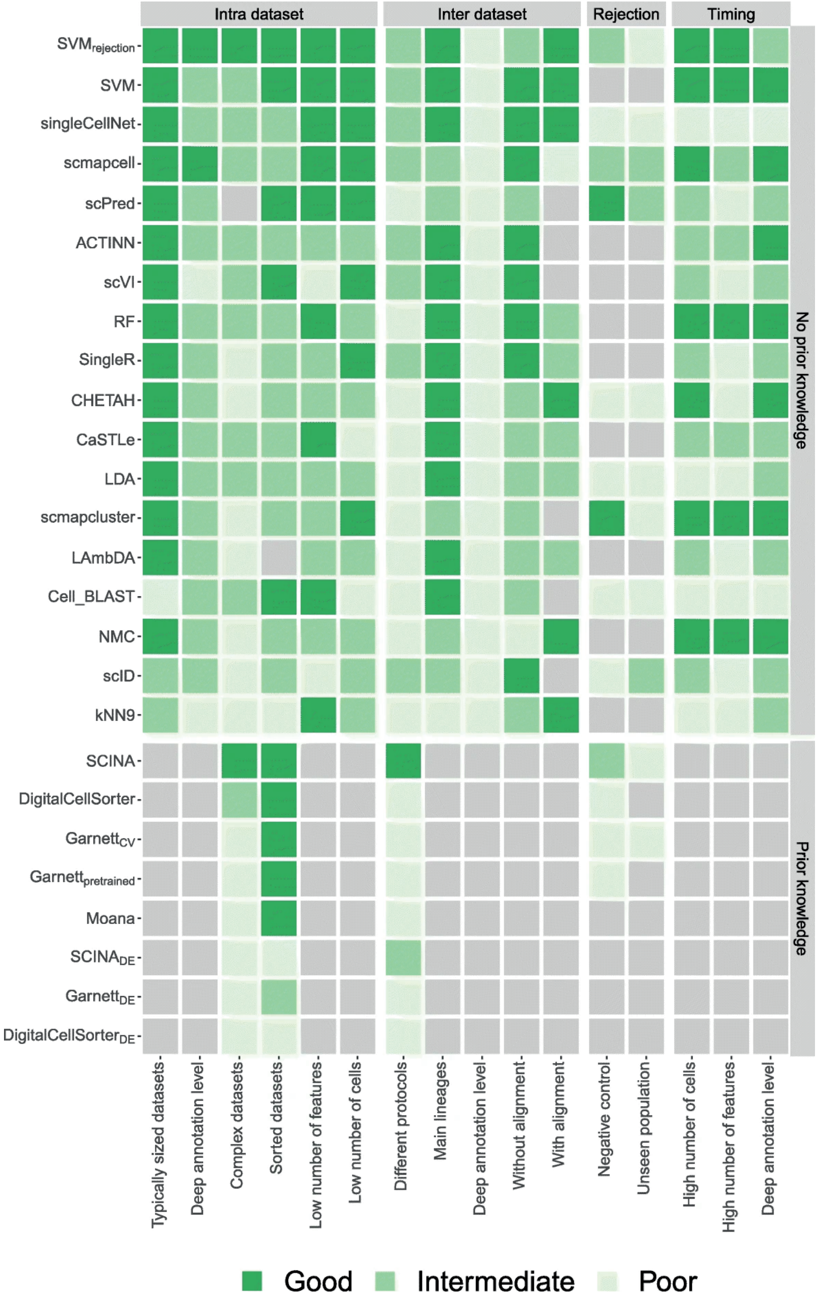


Figure 8. Summary of the performance of all classifiers during different experiments. For each experiment, the heatmap shows whether a classifier performs good, intermediate, or poor. Light-grey indicates that a classifier could not be tested during an experiment. The grey boxes to the right of the heatmap indicate the four different categories of experiments: intra-dataset, inter-dataset, rejection and timing. Experiments itself are indicated using the row labels. Table S4 shows which datasets were used to score the classifiers exactly for each experiment. Grey boxes next to the heatmap indicate the two classifiers categories. Within these two categories, the classifiers are sorted based on their mean performance on the intra and inter dataset experiments.

For datasets with deep levels of annotation (i.e. large number) of cell populations, the classification performance of all classifiers is relatively low, since the classification task is more challenging. *scVI*, in particular, failed to scale with deeply annotated datasets, although it works well for datasets with relatively small number of cell populations. Further, applying the prior-knowledge classifiers becomes infeasible for deeply annotated datasets, as the task of defining the marker-genes becomes even more challenging.

We evaluated the performance of the prior-knowledge methods (marker-based and pre-trained) on PBMC datasets only, due to the limited availability of author-provided marker genes. For all PBMC datasets, the prior-knowledge methods did not improve the classification performance over supervised methods, which do not incorporate such prior knowledge. We extended some prior-knowledge methods such that the marker-genes were defined in a data-driven manner using differential expression which did not improve the performance of these classifiers, except for $SCINA_{DE}$ (with 20 marker-genes) for the PbmcBench datasets. The data-driven selection of markers allows the prediction of more cell populations compared to the number of populations for which marker-genes were originally provided. However, this data-driven selection violates the fundamental assumption in prior-knowledge methods that incorporating expert-defined markers improves classification performance. Further, several supervised classifiers which do not require markers to be defined a priori (e.g. *scPred* and *scID*) already apply a differential expression test to find the best set of genes to use while training the model. The fact that prior-knowledge methods do not outperform other supervised methods and given the challenges associated with explicit marker definition, indicate that incorporating prior knowledge in the form of marker-genes is not beneficial, at least for PBMC data.

In the inter-dataset experiments, we tested the ability of the classifiers to identify populations across different scRNA-seq protocols. Our results show that some protocols are more compatible with one another (e.g. 10Xv2 and 10Xv3), Smart-Seq2 is distinct from the other UMI-based methods, and CEL-Seq2 suffers from low replicability of cell populations across samples. These results can serve as a guide in order to choose the best set of protocols that can be used in studies where more than one protocol is used.

The intra-dataset evaluation included the Zheng sorted dataset, which consists of 10 FACS sorted cell populations based on the expression of surface protein markers. Our results show relatively lower classification performance compared to other datasets, except the Zheng 68K dataset. The poor correlation between the expression levels of these protein markers and their coding genes mRNA levels [46] might explain this low performance.

Overall, we observed that the performance of almost all methods was relatively high on various datasets, while some datasets with overlapping populations (e.g. Zheng 68K dataset) remain challenging. The inter-dataset comparison requires extensive development in order to deal with technical differences between protocols, batches, and labs, as well as proper matching between different cell population annotations. Further, the pancreatic datasets are known to project very well across studies and hence using them to evaluate inter-dataset performance can be misleading. We recommend considering other challenging tissues and cell populations.

2.4 Conclusions

We present a comprehensive evaluation of automatic cell identification methods for single cell RNA-sequencing data. Generally, all classifiers perform well across all datasets, including the general-purpose classifiers. In our experiments, incorporating prior knowledge in the form of marker-genes does not improve the performance (on PBMC data). We observed large differences in the performance between methods in response to changing the input features. Furthermore, the tested methods vary considerably in their computation time which also varies differently across methods based on the number of cells and features.

Taken together, we recommend the use of the general-purpose $SVM_{rejection}$ classifier (with a linear kernel) since it had better performance compared to the other classifiers tested across all datasets. Other high performing classifiers include: *SVM* with a remarkably fast computation time at the expense of losing the rejection option, *singleCellNet, scmapcell*, and *scPred*. To support future extension of this benchmarking work with new classifiers and datasets, we provide a Snakemake workflow to automate the performed benchmarking analyses (https://github.com/tabdelaal/scRNAseq_Benchmark/).

2.5 Methods

2.5.1 Classification methods

We evaluated 22 scRNA-seq classifiers, publicly available as R or Python packages or scripts (Table 1). This set includes 16 methods developed specifically for scRNA-seq data as well as six general-purpose classifiers from the scikit-learn library in Python: linear discriminant analysis (*LDA*), nearest mean classifier (*NMC*), k-nearest neighbor (*kNN*), support vector machine with linear kernel (*SVM*), SVM with rejection option ($SVM_{rejection}$) and random forest (*RF*). The following functions from the scikit-learn library were used respectively: `LinearDiscriminantAnalysis()`, `NearestCentroid()`, `KNeighborsClassifier(n_neighbors=9)`, `LinearSVC()`, `LinearSVC()` with `CalibratedClassifierCV()` wrapper, and `RandomForestClassifier(n_estimators=50)`. For *kNN*, nine neighbors were chosen. After filtering the datasets, only cell populations consisting of ten cells or more remained. Using nine neighbors would thus ensure that this classifier could also predict very small populations. For $SVM_{rejection}$ a threshold of 0.7 was used on the posterior probabilities to assign cells as 'unlabeled'. During the rejection experiments, also an LDA with rejection was implemented. In contrast to the `LinearSVC()`, the `LinearDiscriminantAnalysis()` function can output the posterior probabilities itself, which was also thresholded at 0.7.

scRNA-seq specific methods were excluded from the evaluation if they did not return the predicted labels for each cell. For example, we excluded *MetaNeighbor* [47] because the tool only returns the area under the receiver operator characteristic curve (AUROC). For all methods the latest (May 2019) package was installed or scripts were downloaded from their GitHub. For *scPred* it should be noted that it is only compatible with an older version of

Seurat (v2.0). For *CHETAH* it is important that the R version 3.6 or newer is installed. For *LAmbDA,* instead of the predicted label, the posterior probabilities were returned for each cell population. Here, we assigned the cells to the cell population with the highest posterior probability.

During the benchmark, all methods were run using their default settings and if not available, we used the settings provided in the accompanying examples or vignettes. As input, we provided each method with the raw count data (after cell and gene filtering as described in Section 2.5.3 Data Preprocessing) according to the method documentation. The majority of the methods have a built-in normalization step. For the general-purpose classifiers, we provided log-transformed counts, $\log_2(count+1)$.

Some methods required a marker gene file or pre-trained classifier as an input (e.g. *Garnett*, *Moana*, *SCINA*, *DigitalCellSorter*). In this case, we use the marker gene files of pre-trained classifiers provided by the authors. We did not attempt to include additional marker gene files for all datasets, and hence the evaluation of those methods is restricted to datasets where a marker gene file for cell populations is available.

2.5.2 Datasets

A total of 27 scRNA-seq datasets were used to evaluate and benchmark all classification methods, from which 11 datasets were used for intra-dataset evaluation using a cross-validation scheme, and 22 datasets were used for inter-dataset evaluation, with six datasets overlapping for both tasks as described in Table 2. Datasets vary across species (human and mouse), tissue (brain, pancreas, PBMC and whole mouse), as well as the sequencing protocol used. The brain datasets, including Allen Mouse Brain (AMB), VISp, ALM (GSE115746) and MTG, were downloaded from the Allen Institute Brain Atlas http://celltypes.brain-map.org/rnaseq. All five pancreatic datasets were obtained from: https://hemberg-lab.github.io/scRNA.seq.datasets/ (Baron Mouse: GSE84133, Baron Human: GSE84133, Muraro: GSE85241, Segerstolpe: E-MTAB-5061, Xin: GSE81608). The CellBench 10X dataset was obtained from (GSM3618014), and the CellBench CEL-Seq2 dataset was obtained from 3 datasets (GSM3618022, GSM3618023, GSM3618024) and concatenated into one dataset. The Tabula Muris (TM) dataset was downloaded from https://tabula-muris.ds.czbiohub.org/ (GSE109774). For the Zheng sorted datasets, we downloaded the 10 PBMC sorted populations (CD14+ Monocytes, CD19+ B Cells, CD34+ Cells, CD4+ Helper T Cells, CD4+/CD25+ Regulatory T Cells, CD4+/CD45RA+/CD25- Naive T Cells, CD4+/CD45RO+ Memory T Cells, CD56+ Natural Killer Cells, CD8+ Cytotoxic T cells, CD8+/CD45RA+ Naive Cytotoxic T Cells) from: https://support.10xgenomics.com/single-cell-gene-expression/datasets, next we downsampled each population to 2,000 cells obtaining a dataset of 20,000 cells in total. For the Zheng 68K dataset, we downloaded the gene-cell count matrix for the 'Fresh 68k PBMCs' [36] from: https://support.10xgenomics.com/single-cell-gene-expression/datasets (SRP073767). All 13 PbmcBench datasets, seven different sequencing protocols applied on two PBMC samples, were downloaded from the Broad Institute Single Cell portal https://portals.broadinstitute.org/single_cell/study/SCP424/single-cell-comparison-pbmc-data. The cell population annotation for all datasets was provided with the data, except the Zheng 68K dataset, for which we obtained the cell population annotation from https://github.com/10XGenomics/single-cell-3prime-paper/tree/master/pbmc68k_analysis. These annotations were used as 'ground truth' during the evaluation of the cell population predictions obtained from the classification methods.

2.5.3 Data preprocessing

Based on the manual annotation provided in the datasets, we started by filtering out cells that were labeled as doublets, debris or unlabeled cells. Next, we filtered genes with zero counts across all cells. For cells, we calculated the median number of detected genes per cell, and from that we obtained the median absolute deviation (MAD) across all cells in the log scale. We filtered out cells when the total number of detected genes was below three MAD from the median number of detected genes per cell. The number of cells and genes in Table 2 represent the size of each dataset after this stage of preprocessing.

Moreover, before applying cross validation to evaluate each classifier, we excluded cell populations with less than 10 cells across the entire dataset; Table 2 summarizes the number of cell populations before and after this filtration step for each dataset.

2.5.4 Intra-dataset classification

For the supervised classifiers, we evaluated the performance by applying a 5-fold cross validation across each dataset after filtering genes, cells and small cell populations. The folds were divided in a stratified manner in order to keep equal proportions of each cell population in each fold. The training and testing folds were exactly the same for all classifiers.

The prior-knowledge classifiers, *Garnett*, *Moana*, *DigitalCellSorter* and *SCINA*, were only evaluated on the Zheng 68K and Zheng sorted datasets, for which the marker-genes files or the pre-trained classifiers were available, after filtering genes and cells. Each classifier uses the dataset and the marker-genes file as inputs, and outputs the cell population label corresponding to each cell. No cross validation is applied in this case, except for *Garnett* where we could either use the pretrained version ($Garnett_{pretrained}$) provided from the original study, or train our own classifier using the marker-genes file along with the training data ($Garnett_{CV}$). In this case, we applied 5-fold cross validation using the same train and test sets described earlier. Table S1 shows the mapping of cell populations between the Zheng dataset and each of the prior-knowledge classifiers. For *Moana* a pre-trained classifier was used, this classifier also predicted cells to be Memory CD8+ T cells and CD16+ Monocytes, while these cell populations were not in the Zheng dataset.

2.5.5 Evaluation of marker-genes

The performance and choice of the marker-genes per cell population per classifier were evaluated by comparing the F1-score of each cell population with four different characteristics of the marker-genes across the cells for that particular cell population: 1) the number of marker-genes, 2) the mean expression, 3) the average dropout rate, and 4) the average beta of the marker-genes [37]. Beta is a score developed to measure how specific a marker gene for a certain cell population is based on binary expression.

2.5.6 Selecting marker-genes using differential expression

Using the cross-validation scheme, training data of each fold was used to select sets of 5, 10, 15, and 20 differentially expressed (DE) marker-genes. First, if the data was not already normalized, a CPM read count normalization was applied to the data. Next, the data was log-transformed using $\log_2(count+1)$, and afterwards the DE test could be applied. As recommended in [48], MAST was used to find the DE genes [49]. The implementation of MAST in the FindAllMarkers() function of Seurat v2.3.0 was used to do a one-vs-all differential expression analysis [50]. Genes returned by Seurat were sorted and the top 5, 10, 15, or 20 significant genes with a positive fold change were selected as marker-genes. These marker-genes were then used for population prediction of the test data of the corresponding fold. These marker-genes lists can be used by prior-knowledge classifiers such as *SCINA*, *Garnett*$_{CV}$ and *DigitalCellSorter*, by modifying the cell type marker-genes file required as an input to these classifiers. Such modification cannot be applied to the pre-trained classifiers of *Garnett*$_{pretrained}$ and *Moana*.

2.5.7 Dataset complexity

To describe the complexity of a dataset, the average expression of all genes for each cell population (avg_{c_i}) in the dataset was calculated, representing the prototype of each cell population in the full genes space. Next, the pairwise Pearson correlation between these centroids was calculated $\mathrm{corr}_{\forall i,j}(\mathrm{avg}_{c_i}, \mathrm{avg}_{c_j})$. For each cell population, the highest correlation to another cell population was recorded. Finally, the mean of these per cell population maximum correlations was taken to describe the complexity of a dataset.

$$\mathrm{Complexity} = \mathrm{mean}(\max_{\forall i, i \neq j}(\mathrm{corr}_{\forall i,j}(\mathrm{avg}_{c_i}, \mathrm{avg}_{c_j})))$$

2.5.8 Inter-dataset classification

CellBench. Both CellBench datasets, 10X and CEL-Seq2, were used once as training data and once as test data, to obtain predictions for the five lung cancer cell lines. The common set of detected genes by both datasets was used as features in this experiment.

PbmcBench. Using pbmc1 sample only, we tested all train-test pairwise combinations between all seven protocols, resulting in 42 experiments. Using both pbmc1 and pbmc2 samples, for the same protocol we used pbmc1 as training data and pbmc2 as test data, resulting in six additional experiments (10Xv3 was not applied for pbmc2). As we are now dealing with PBMC data, we evaluated all classifiers, including the prior-knowledge classifiers, as well as the modified versions of *SCINA*, *Garnett*$_{CV}$ and *DigitalCellSorter,* in which the marker-genes are obtained through differential expression from the training data as previously described. Through all these 48 experiments, genes that are not expressed in the training data were excluded from the feature space. Also, as these PbmcBench datasets differ in the number of cell populations (Table 2), only cell populations provided by the training data were used for the test data prediction evaluation.

Brain. We used the three brain datasets, VISp, ALM and MTG with two levels of annotations, 3 and 34 cell populations. We tested all possible train-test combinations, by either using one dataset to train and test on another (6 experiments) or using two concatenated datasets to train and test on the third (3 experiments). A total of nine experiments was applied for each annotation level. We used the common set of detected genes between the datasets involved in each experiment as features.

Pancreas. We selected the four major endocrine pancreatic cell types (alpha, beta, delta and gamma) across all four human pancreatic datasets: Baron Human, Muraro, Segerstolpe and Xin. Table S2 summarizes the number of cells in each cell type across all datasets. To account for batch effects and technical variations between different protocols, datasets were aligned using MNN [41] from the scran R package (version 1.1.2.0). Using both the raw data (un-aligned) and the aligned data, we applied leave-one-dataset-out cross validation where we train on three datasets and test on the left out dataset.

2.5.9 Performance evaluation metrics

The performance of the methods on the datasets is evaluated using three different metrics: 1) For each cell population in the dataset the F1-score is reported. The median of these F1-scores is used as a measure for the performance on the dataset. 2) Some of the methods do not label all the cells. These unassigned cells are not considered in the F1-score calculation. The percentage of unlabeled cells is also used to evaluate the performance. 3) The computation time of the methods is also measured.

2.5.10 Feature selection

Genes are selected as features based on their dropout rate. The method used here, is based on the method described in [22]. During feature selection, a sorted list of the genes is made. Based on this list, the top *n* number of genes can be easily selected during the experiments. First, the data is normalized using $\log_2(count+1)$. Next, for each gene the percentage of dropouts, *d*, and the mean, *m*, of the normalized data are calculated. Genes that have a mean or dropout rate of zero are not considered during the next steps. These genes will be at the bottom of the sorted list. For all other genes, a linear model is fitted to the mean and $\log_2(d)$. Based on their residuals, the genes are sorted in descending order and added to the top of the list.

2.5.11 Scalability

For the scalability experiment we used the TM dataset. To ensure that the dataset could be downsampled without losing cell populations, only the 16 most abundant cell populations were considered during this experiment. We downsampled these cell populations in a stratified way to 1, 5, 10, 20, 50, and 100% of its original size (45,469 cells).

2.5.12 Rejection

Negative control. Two human datasets, Zheng 68K and Baron Human, and two mouse datasets, AMB16 and Baron Mouse, were used. The Zheng 68K dataset was first stratified downsampled to 11% of its original size to reduce computation time. For each species, two different experiments were applied by using one dataset as training set and the other as test set and vice versa.

Unseen cell populations. Zheng 68K dataset was stratified downsampled to 11% of its original size to reduce computation time. Three different experiments were conducted. First, all cell populations that are subpopulation of T cells were considered the test set. Next, the test set consisted of all subpopulations of CD4+ T cells. Last, only the CD4+/CD45RO+ Memory T cells were in the test set. Each time, all cell populations that were not in the test set, were part of the training set. Table S3 gives an exact overview of the populations per training and test set.

2.5.13 Benchmarking pipeline

In order to ensure reproducibility and support future extension of this benchmarking work with new classification methods and benchmarking datasets, a Snakemake [51] workflow for automating the performed benchmarking analyses was developed with an MIT license (https://github.com/tabdelaal/scRNAseq_Benchmark/). Each tool (license permitting) is packaged in a Docker container (https://hub.docker.com/u/scrnaseqbenchmark) alongside the wrapper scripts and their dependencies. These images will be used through snakemake's singularity integration to allow the workflow to be run without the requirement to install specific methods and to ensure reproducibility. Documentation is also provided to execute and extend this benchmarking workflow to help researchers to further evaluate interested methods.

2.6 Availability of data and material

The filtered datasets analyzed during the current study can be downloaded from Zenodo (https://doi.org/10.5281/zenodo.3357167). The source code is available in th e GitHub repository, at https://github.com/tabdelaal/scRNAseq_Benchmark [52], and in the Zenodo repository, at https://doi.org/10.5281/zenodo.3369158 [53]. The source code is released under MIT license. Datasets accession numbers: AMB, VISp, and ALM [35] (GSE115746), MTG [31] (phs001790), Baron Mouse [30] (GSE84133), Baron Human [30] (GSE84133), Muraro

[31] (GSE85241), Segerstolpe [32] (E-MTAB-5061), Xin [33] (GSE81608), CellBench 10X [34] (GSM3618014), CellBench CEL-Seq2 [34] (GSM3618022, GSM3618023, GSM3618024), TM [6] (GSE109774), and Zheng sorted and Zheng 68K [36] (SRP073767). The PbmcBench datasets [38] are not yet uploaded to any data repository.

Bibliography

1. Svensson V, Vento-Tormo R, Teichmann SA. Exponential scaling of single-cell RNA-seq in the past decade. Nat Protoc. 2018;13: 599–604. doi:10.1038/nprot.2017.149
2. Plass M, Solana J, Wolf FA, Ayoub S, Misios A, Glažar P, et al. Cell type atlas and lineage tree of a whole complex animal by single-cell transcriptomics. Science. 2018;360. doi:10.1126/science.aaq1723
3. Cao J, Packer JS, Ramani V, Cusanovich DA, Huynh C, Daza R, et al. Comprehensive single-cell transcriptional profiling of a multicellular organism. Science. 2017;357: 661–667. doi:10.1126/science.aam8940
4. Fincher CT, Wurtzel O, de Hoog T, Kravarik KM, Reddien PW. Cell type transcriptome atlas for the planarian. Science. 2018;360. doi:10.1126/science.aaq1736
5. Han X, Wang R, Zhou Y, Fei L, Sun H, Lai S, et al. Mapping the Mouse Cell Atlas by Microwell-Seq. Cell. 2018;173: 1307. doi:10.1016/j.cell.2018.05.012
6. Schaum N, Karkanias J, Neff NF, May AP, Quake SR, Wyss-Coray T, et al. Single-cell transcriptomics of 20 mouse organs creates a Tabula Muris. Nature. 2018;562: 367–372. doi:10.1038/s41586-018-0590-4
7. Cao J, Spielmann M, Qiu X, Huang X, Ibrahim DM, Hill AJ, et al. The single-cell transcriptional landscape of mammalian organogenesis. Nature. 2019;566: 496–502. doi:10.1038/s41586-019-0969-x
8. Henry VJ, Bandrowski AE, Pepin A-S, Gonzalez BJ, Desfeux A. OMICtools: an informative directory for multi-omic data analysis. Database . 2014;2014. doi:10.1093/database/bau069
9. Zappia L, Phipson B, Oshlack A. Exploring the single-cell RNA-seq analysis landscape with the scRNA-tools database. PLoS Comput Biol. 2018;14: e1006245. doi:10.1371/journal.pcbi.1006245
10. Saelens W, Cannoodt R, Todorov H, Saeys Y. A comparison of single-cell trajectory inference methods. Nat Biotechnol. 2019;37: 547–554. doi:10.1038/s41587-019-0071-9
11. Duò A, Robinson MD, Soneson C. A systematic performance evaluation of clustering methods for single-cell RNA-seq data. F1000Res. 2018;7: 1141. doi:10.12688/f1000research.15666.2
12. Soneson C, Robinson MD. Bias, robustness and scalability in single-cell differential expression analysis. Nat Methods. 2018;15: 255–261. doi:10.1038/nmeth.4612
13. Diaz-Mejia JJ, Javier Diaz-Mejia J, Meng EC, Pico AR, MacParland SA, Ketela T, et al. Evaluation of methods to assign cell type labels to cell clusters from single-cell RNA-sequencing data. 2019. doi:10.1101/562082
14. Pliner HA, Shendure J, Trapnell C. Supervised classification enables rapid annotation of cell atlases. bioRxiv. 2019. p. 538652. doi:10.1101/538652
15. Wagner F, Yanai I. Moana: A robust and scalable cell type classification framework for single-cell RNA-Seq data. bioRxiv. 2018; 456129. doi:10.1101/456129
16. Domanskyi S, Szedlak A, Hawkins NT, Wang J, Paternostro G, Piermarocchi C. Polled Digital Cell Sorter (p-DCS): Automatic identification of hematological cell types from single cell RNA-sequencing clusters. bioRxiv. 2019; 539833. doi:10.1101/539833
17. Zhang Z, Luo D, Zhong X, Choi JH, Ma Y, Mahrt E, et al. SCINA: Semi-Supervised Analysis of Single Cells in silico. bioRxiv. 2019; 559872. doi:10.1101/559872
18. Lopez R, Regier J, Cole MB, Jordan MI, Yosef N. Deep generative modeling for single-cell transcriptomics. Nat Methods. 2018;15: 1053–1058. doi:10.1038/s41592-018-0229-2
19. Cao Z-J, Wei L, Lu S, Yang D-C, Gao G. Cell BLAST: Searching large-scale scRNA-seq databases via unbiased cell embedding. bioRxiv. 2019. p. 587360. doi:10.1101/587360
20. Ma F, Pellegrini M. Automated identification of Cell Types in Single Cell RNA Sequencing. bioRxiv. 2019; 532093. doi:10.1101/532093
21. Johnson TS, Wang T, Huang Z, Yu CY, Wu Y, Han Y, et al. LAmbDA: Label Ambiguous Domain Adaptation Dataset Integration Reduces Batch Effects and Improves Subtype Detection. Bioinformatics. 2019. doi:10.1093/bioinformatics/btz295
22. Kiselev VY, Yiu A, Hemberg M. scmap: projection of single-cell RNA-seq data across data sets. Nat Methods. 2018;15: 359. Available: https://doi.org/10.1038/nmeth.4644
23. Alquicira-Hernandez J, Nguyen Q, Powell JE. scPred: scPred: Cell type prediction at single-cell resolution. bioRxiv. 2018; 369538. doi:10.1101/369538
24. Kanter JK de, Lijnzaad P, Candelli T, Margaritis T, Holstege F. CHETAH: a selective, hierarchical cell type identification method for single-cell RNA sequencing. bioRxiv. 2019; 558908. doi:10.1101/558908
25. Lieberman Y, Rokach L, Shay T. CaSTLe – Classification of single cells by transfer learning: Harnessing the power of publicly available single cell RNA sequencing experiments to annotate new experiments. Kaderali L, editor. PLoS One. 2018;13: e0205499. doi:10.1371/journal.pone.0205499
26. Aran D, Looney AP, Liu L, Wu E, Fong V, Hsu A, et al. Reference-based analysis of lung single-cell sequencing reveals a transitional profibrotic macrophage. Nat Immunol. 2019;20: 163–172. doi:10.1038/s41590-018-0276-y

27. Boufea K, Seth S, Batada NN. scID: Identification of equivalent transcriptional cell populations across single cell RNA-seq data using discriminant analysis. doi:10.1101/470203

28. Tan Y, Cahan P. SingleCellNet: a computational tool to classify single cell RNA-Seq data across platforms and across species. bioRxiv. 2018. p. 508085. doi:10.1101/508085

29. Pedregosa F, Varoquaux G, Gramfort A, Michel V, Thirion B, Grisel O, et al. Scikit-learn: Machine Learning in Python. 2011 pp. 2825–2830. Available: http://scikit-learn.sourceforge.net.

30. Baron M, Veres A, Wolock SL, Faust AL, Gaujoux R, Vetere A, et al. A Single-Cell Transcriptomic Map of the Human and Mouse Pancreas Reveals Inter- and Intra-cell Population Structure. Cell Syst. 2016;3: 346–360.e4. doi:10.1016/j.cels.2016.08.011

31. Muraro MJ, Dharmadhikari G, Grün D, Groen N, Dielen T, Jansen E, et al. A Single-Cell Transcriptome Atlas of the Human Pancreas. Cell Syst. 2016;3: 385–394.e3. doi:10.1016/j.cels.2016.09.002

32. Segerstolpe Å, Palasantza A, Eliasson P, Andersson E-M, Andréasson A-C, Sun X, et al. Single-Cell Transcriptome Profiling of Human Pancreatic Islets in Health and Type 2 Diabetes. Cell Metab. 2016;24: 593–607. doi:10.1016/j.cmet.2016.08.020

33. Xin Y, Kim J, Okamoto H, Ni M, Wei Y, Adler C, et al. RNA Sequencing of Single Human Islet Cells Reveals Type 2 Diabetes Genes. Cell Metab. 2016;24: 608–615. doi:10.1016/j.cmet.2016.08.018

34. Tian L, Dong X, Freytag S, Lê Cao K-A, Su S, JalalAbadi A, et al. Benchmarking single cell RNA-sequencing analysis pipelines using mixture control experiments. Nat Methods. 2019;16: 479–487. doi:10.1038/s41592-019-0425-8

35. Tasic B, Yao Z, Graybuck LT, Smith KA, Nguyen TN, Bertagnolli D, et al. Shared and distinct transcriptomic cell types across neocortical areas. Nature. 2018;563: 72–78. doi:10.1038/s41586-018-0654-5

36. Zheng GXY, Terry JM, Belgrader P, Ryvkin P, Bent ZW, Wilson R, et al. Massively parallel digital transcriptional profiling of single cells. Nat Commun. 2017;8: 14049. doi:10.1038/ncomms14049

37. Hodge RD, Bakken TE, Miller JA, Smith KA, Barkan ER, Graybuck LT, et al. Conserved cell types with divergent features between human and mouse cortex. bioRxiv. 2018; 384826. doi:10.1101/384826

38. Ding J, Adiconis X, Simmons SK, Kowalczyk MS, Hession CC, Marjanovic ND, et al. Systematic comparative analysis of single cell RNA-sequencing methods. bioRxiv. 2019. p. 632216. doi:10.1101/632216

39. Franzén O, Gan L-M, Björkegren JLM. PanglaoDB: a web server for exploration of mouse and human single-cell RNA sequencing data. Database . 2019;2019. doi:10.1093/database/baz046

40. Zhang X, Lan Y, Xu J, Quan F, Zhao E, Deng C, et al. CellMarker: a manually curated resource of cell markers in human and mouse. Nucleic Acids Res. 2019;47: D721–D728. doi:10.1093/nar/gky900

41. Haghverdi L, Lun ATL, Morgan MD, Marioni JC. Batch effects in single-cell RNA-sequencing data are corrected by matching mutual nearest neighbors. Nat Biotechnol. 2018;36: 421–427. doi:10.1038/nbt.4091

42. McInnes L, Healy J, Melville J. UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction. arXiv [stat.ML]. 2018. Available: http://arxiv.org/abs/1802.03426

43. Andrews TS, Hemberg M. M3Drop: dropout-based feature selection for scRNASeq. Birol I, editor. Bioinformatics. 2018. doi:10.1093/bioinformatics/bty1044

44. D. Cai, X. He, J. Han. Training Linear Discriminant Analysis in Linear Time. 2008. doi:10.1109/ICDE.2008.4497429

45. Köhler ND, Büttner M, Theis FJ. Deep learning does not outperform classical machine learning for cell-type annotation. bioRxiv. 2019. p. 653907. doi:10.1101/653907

46. van den Berg PR, Budnik B, Slavov N, Semrau S. Dynamic post-transcriptional regulation during embryonic stem cell differentiation. bioRxiv. 2017. p. 123497. doi:10.1101/123497

47. Crow M, Paul A, Ballouz S, Huang ZJ, Gillis J. Characterizing the replicability of cell types defined by single cell RNA-sequencing data using MetaNeighbor. Nat Commun. 2018;9: 884. doi:10.1038/s41467-018-03282-0

48. Luecken MD, Theis FJ. Current best practices in single-cell RNA-seq analysis: a tutorial. Mol Syst Biol. 2019;15: e8746. doi:10.15252/msb.20188746

49. Finak G, McDavid A, Yajima M, Deng J, Gersuk V, Shalek AK, et al. MAST: a flexible statistical framework for assessing transcriptional changes and characterizing heterogeneity in single-cell RNA sequencing data. Genome Biol. 2015;16: 278. doi:10.1186/s13059-015-0844-5

50. Butler A, Hoffman P, Smibert P, Papalexi E, Satija R. Integrating single-cell transcriptomic data across different conditions, technologies, and species. Nat Biotechnol. 2018;36: 411–420. doi:10.1038/nbt.4096

51. Köster J, Rahmann S. Snakemake—a scalable bioinformatics workflow engine. Bioinformatics. 2018. pp. 3600–3600. doi:10.1093/bioinformatics/bty350

chapter 3

Hierarchical progressive learning of cell identities in single-cell data

Lieke Michielsen, Marcel J.T. Reinders, Ahmed Mahfouz

This chapter is published in: *Nature Communications* (2021) 12: 2799, doi: 10.1038/s41467-021-23196-8.
Supplementary material is available online at:
https://www.nature.com/articles/s41467-021-23196-8#Sec27

Supervised methods are increasingly used to identify cell populations in single-cell data. Yet, current methods are limited in their ability to learn from multiple datasets simultaneously, are hampered by the annotation of datasets at different resolutions, and do not preserve annotations when retrained on new datasets. The latter point is especially important as researchers cannot rely on downstream analysis performed using earlier versions of the dataset. Here, we present scHPL, a hierarchical progressive learning method which allows continuous learning from single-cell data by leveraging the different resolutions of annotations across multiple datasets to learn and continuously update a classification tree. We evaluate the classification and tree learning performance using simulated as well as real datasets and show that scHPL can successfully learn known cellular hierarchies from multiple datasets while preserving the original annotations. scHPL is available at https://github.com/lcmmichielsen/scHPL.

3.1 Introduction

Cell identification is an essential step in single-cell studies with profound effects on downstream analysis. For example, in order to compare cell-population-specific eQTL findings across studies, cell identities should be consistent [1]. Currently, cells in single-cell RNA-sequencing (scRNA-seq) datasets are primarily annotated using clustering and visual exploration techniques, i.e. cells are first clustered into populations which are subsequently named based on the expression of marker genes. This is not only time-consuming, but also subjective [2]. The number of cell populations identified in a dataset, for example, is strongly correlated with the number of cells analyzed, which results in inconsistency across datasets [3–5].

Recently, many supervised methods have been developed to replace unsupervised techniques. The underlying principles of these methods vary greatly. Some methods, for instance, rely on prior knowledge and assume that for each cell population marker genes can be defined (e.g. SCINA [6] and Garnett [7]), while others search for similar cells in a reference database (e.g. scmap [8] and Cell-BLAST [9]), or train a classifier using a reference atlas or a labeled dataset (e.g. scPred [10] and CHETAH [11]).

Supervised methods rely either on a reference atlas or labeled dataset. Ideally, we would use a reference atlas containing all possible cell populations to train a classifier. Such an atlas, however, does not exist yet and might never be fully complete. In particular, aberrant cell populations might be missing as a huge number of diseases exist and mutations could result in new cell populations. To overcome these limitations, some methods (e.g. OnClass) rely on the Cell Ontology to identify cell populations that are missing from the training data but do exist in the Cell Ontology database [12]. These Cell Ontologies, however, were not developed for scRNA-seq data specifically. As a consequence, many newly identified (sub) populations are missing and relationships between cell populations might be inaccurate. A striking example of this inadequacy are neuronal cell populations. Recent single-cell studies have identified hundreds of populations [4,13,14], including seven subtypes and 92 cell populations in one study only [5]. In contrast, the Cell Ontology currently includes only one glutamatergic neuronal cell population without any subtypes.

Since no complete reference atlas is available, a classifier should ideally be able to combine the information of multiple annotated datasets and continue learning. Each time a new cell population is found in a dataset, it should be added to the knowledge of the classifier. We advocate that this can be realized with progressive learning, a learning strategy inspired by humans. Human learning is a continuous process that never ends [15]. Using progressive learning, the task complexity is gradually increased, for instance, by adding more classes, but it is essential that the knowledge of the previous classes is preserved [16,17]. This strategy allows combining information of multiple existing datasets and retaining the possibility to add more datasets afterwards. However, it cannot be simply applied to scRNA-seq datasets as a constant terminology to describe cell populations is missing, which eliminates straightforward identification of new cell populations based on their names. For example, the recently discovered neuronal populations are typically identified using clustering and named based on the expression of marker genes. A standardized nomenclature for these clusters is missing [18], so the relationship between cell populations defined in different datasets is often unknown.

Moreover, the level of detail (resolution) at which datasets are annotated highly depends on the number of cells analyzed [19]. For instance, if a dataset is annotated at a low resolution, it might contain T-cells, while a dataset at a higher resolution can include subpopulations of T-cells, such as CD4+ and CD8+ T-cells. We need to consider this hierarchy of cell populations in our representation, which can be done with a hierarchical classifier. This has the advantage that cell population definitions of multiple datasets can be combined, ensuring consistency. A hierarchical classifier has additional advantages in comparison to a classifier that does not exploit this hierarchy between classes (here denoted as ‘flat classifier’). One of these advantages is that the classification problem is divided into smaller sub-problems, while a flat classifier needs to distinguish between many classes simultaneously. Another advantage is that if we are not sure about the annotation of an unlabeled cell at the highest resolution, we can always label it as an intermediate cell population (i.e. at a lower resolution).

Currently, some classifiers, such as Garnett, CHETAH, and Moana, already exploit this hierarchy between classes [7,11,20]. Garnett and Moana both depend on prior knowledge in the form of marker genes for the different classes. Especially for deeper annotated datasets it can be difficult to define marker genes for each cell population that are robust across scRNA-seq datasets [21,22]. Moreover, we have previously shown that adding prior knowledge is not beneficial [23]. CHETAH, on the contrary, constructs a classification tree based on one dataset by hierarchically clustering the reference profiles of the cell populations and classifies new cells based on the similarity to the reference profile of that cell population. However, simple flat classifiers outperform CHETAH [23], indicating that a successful strategy to exploit this hierarchy is still missing. Furthermore, these hierarchical classifiers cannot exploit the different resolutions of multiple datasets as this requires adaptation of the hierarchical representation.

Even if multiple datasets are combined into a hierarchy, there might be unseen populations in an unlabeled dataset. Identifying these cells as a new population is a challenging problem. Although some classifiers have implemented an option to reject cells, they usually fail when being tested in a realistic scenario [23]. In most cases, the rejection option is implemented by setting a threshold on the posterior probability [7,10,23,24]. If the highest posterior probability does not exceed a threshold, the cell is rejected. By looking at the posterior, the actual similarity between a cell and the cell population is ignored.

In this work, we propose a hierarchical progressive learning approach to overcome these challenges. To summarize our contributions: (i) we exploit the hierarchical relationships between cell populations to be able to classify data sets at different resolutions, (ii) we propose a progressive learning approach that updates the hierarchical relationships dynamically and consistently, and (iii) we adopt an advanced rejection procedure including a one-class classifier to be able to discover new cell (sub)populations.

3.2 Results

3.2.1 Hierarchical progressive learning of cell identities

We developed scHPL, a hierarchical progressive learning approach to learn a classification tree using multiple labeled datasets (Figure 1A) and use this tree to predict the labels of a new, unlabeled dataset (Figure 1B). The tree is learned using multiple iterations (Methods). First, we match the labels of two datasets by training a flat classifier for each dataset and predicting the labels of the other dataset. Based on these predictions we create a matching matrix (*X*)

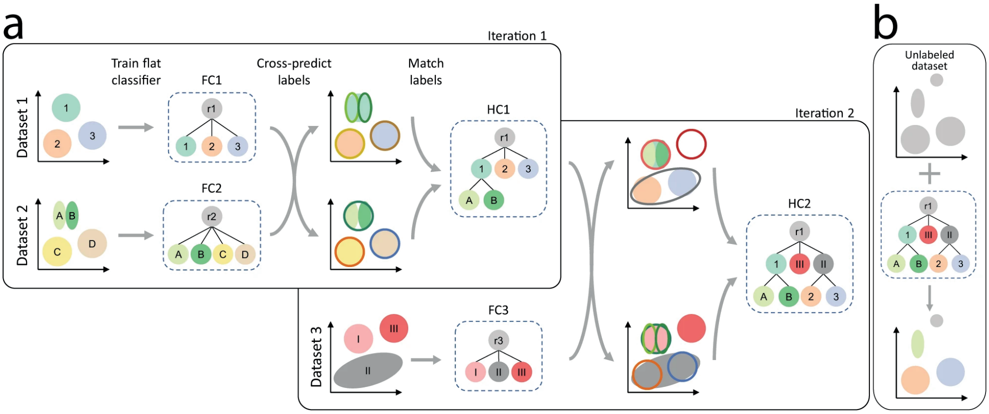


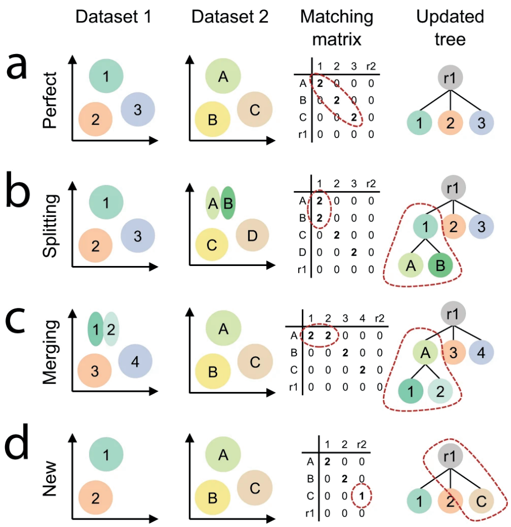
Figure 1. Schematic overview of scHPL. A) Overview of the training phase. In the first iteration, we start with two labeled datasets. The colored areas represent the different cell populations. For both datasets a flat classifier (FC1 & FC2) is constructed. Using this tree and the corresponding dataset, a classifier is trained for each node in the tree except for the root. We use the trained classification tree of one dataset to predict the labels of the other. The decision boundaries of the classifiers are indicated with the contour lines. We compare the predicted labels to the cluster labels to find matches between the labels of the two datasets. The tree belonging to the first dataset is updated according to these matches, which results in a hierarchical classifier (HC1). In dataset 2, for example, subpopulations of population ‘1’ of dataset 1 are found. Therefore, these cell populations, ‘A’ and ‘B’, are added as children to the ‘1’ population. In iteration 2, a new labeled dataset is added. Again a flat classifier (FC3) is trained for this dataset and HC1 is trained on dataset 1 and 2, combined. After cross-prediction and matching the labels, we update the tree which is then trained on all datasets 1-3 (HC2). **B)** The final classifier can be used to annotate a new unlabeled dataset. If this dataset contains unknown cell populations, these will be rejected.

and match the cell populations of the two datasets. In the matching process, we separate different biological scenarios, such as a perfect match, merging or splitting cell populations, as well as creating a new population (Figure 2, Table S1). In the following iterations, we add one labeled dataset at a time by training a flat classifier on this new dataset and training the previously learned tree on all pre-existing datasets. Similar to the previous iteration, the tree is updated after cross-prediction and matching of the labels. It could happen that datasets are inconsistently labeled and the labels cannot be matched (Supplementary Note 1). In this case, one of the populations might be missing from the tree.

Either during tree learning or prediction, there can be unseen populations. Therefore, an efficient rejection option is needed, which we do in two steps. First, we reject cells by thresholding the reconstruction error of a cell when applying a PCA-based dimension reduction: a new, unknown, population is not used to learn the PCA transformation, and consequently will not be properly represented by the selected PCs, leading to a high reconstruction error (Methods). Second, to accommodate rejections when the new population is within the selected PCA domain, scHPL adopts two alternatives to classify cells: a linear and a one-class support vector machine (SVM). The linear SVM has shown a high performance in a benchmark of scRNA-seq classifiers [23], but has a limited rejection option. Whereas, the one-class SVM solves this as only positive training samples are used to fit a tight decision boundary around [25].

3.2.2 Linear SVM has a higher classification accuracy than one-class SVM

We tested our hierarchical classification scheme by measuring the classification performance of the one-class SVM and linear SVM on simulated, PBMC (PBMC-FACS) and brain (Allen Mouse Brain) data using 10-, 10-, and 5-fold cross-validation respectively (Methods). The



	1	2	3	r2
A	**2**	0	0	0
B	0	**2**	0	0
C	0	0	**2**	0
r1	0	0	0	0

	1	2	3	r2
A	**2**	0	0	0
B	**2**	0	0	0
C	0	**2**	0	0
D	0	0	**2**	0
r1	0	0	0	0

	1	2	3	4	r2
A	**2**	**2**	0	0	0
B	0	0	**2**	0	0
C	0	0	0	**2**	0
r1	0	0	0	0	0

	1	2	r2
A	**2**	0	0
B	0	**2**	0
C	0	0	**1**
r1	0	0	0

Figure 2. Schematic examples of different matching scenarios. A) Perfect match, **B)** splitting, **C)** merging, **D)** new population. The first two columns represent a schematic representation of two datasets. After cross-predictions, the matching matrix *(X)* is constructed using the confusion matrices (Methods). We update the tree based on *X*.

simulated dataset was constructed using Splatter [26] and consists of 8,839 cells, 9,000 genes and 6 different cell populations (Figure S1). PBMC-FACS is the downsampled FACS-sorted PBMC dataset from Zheng et al. [27] and consists of 20,000 cells and 10 cell populations. The Allen Mouse Brain (AMB) dataset is challenging as it has deep annotation levels [5], containing 92 different cell populations ranging in size from 11 to 1,348 cells. In these experiments, the classifiers were trained on predefined trees (Figure S1-3).

On all datasets, the linear SVM performs better than the one-class SVM (Figure 3A-D). The simulated dataset is relatively easy since the cell populations are widely separated (Figure S1C). The linear SVM shows an almost perfect performance: only 0.9% of the cells are rejected (based on the reconstruction error only), which is in line with the adopted threshold resulting in 1% false negatives. The one-class SVM labels 92.9% of the cells correctly, the rest is labeled as an internal node (2.3%) or rejected (4.8%), which results in a median Hierarchical F1-score (HF1-score) of 0.973, where HF1 is an F1-score that considers class importance across the hierarchy (Methods).

As expected, the performance of the classifiers on real data drops, but the HF1-scores remain higher than 0.9. On both the PBMC-FACS and AMB dataset, the performance of the linear

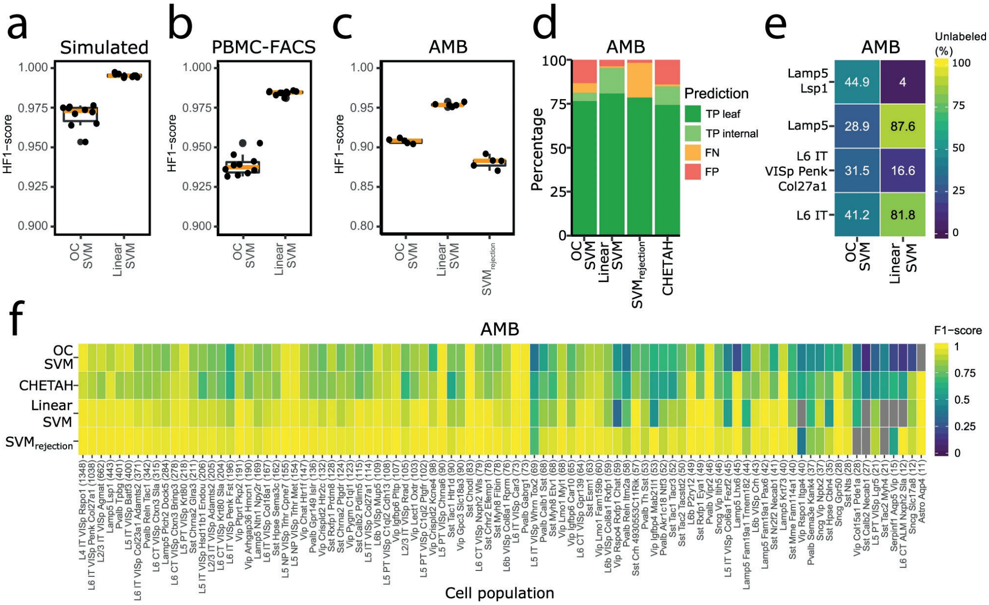


Figure 3. Classification performance. A-C) Boxplots showing the HF1-score of the one-class and linear SVM during *n*-fold cross-validation on the **A)** simulated (*n* = 10), **B)** PBMC-FACS (*n* = 10), and **C)** AMB (*n* = 5) dataset. In the boxplots, the middle (orange) line represents the median, the lower and upper hinge represent the first and third quartiles, and the lower and upper whisker represent the values no further than 1.5 inter-quartile range away from the lower and upper hinge respectively. **D)** Barplot showing the percentage of true positives (TP), false negatives (FN), and false positives (FP) per classifier on the AMB dataset. For the TPs a distinction is made between correctly predicted leaf nodes and internal nodes. **E)** Heatmap showing the percentage of unlabeled cells per classifier during the different rejection experiments. **F)** Heatmap showing the F1-score per classifier per cell population on the AMB dataset. Grey indicates that a classifier never predicted a cell to be of that population.

SVM is higher than the one-class SVM (Figure 3B-D). For the AMB dataset, we used the same cross-validation folds as in Abdelaal et al. [23], which enables us to compare our results. When comparing to CHETAH, which allows hierarchical classification, we notice that the linear SVM performs better based on the median F1-score (0.94 vs 0.83). The one-class SVM has a slightly lower median F1-score (0.80 vs 0.83), but it has more correctly predicted cells and less wrongly predicted cells (Figure 3D).

The linear (hierarchical) SVM also shows a better performance compared to $\text{SVM}_{\text{rejection}}$, which is a flat linear SVM with rejection option based on the posterior probability and was the best classifier for this data [23]. $\text{SVM}_{\text{rejection}}$, however, has a slightly higher median F1-score (0.98 vs 0.94). This is mainly because it makes almost no mistakes, only 1.7% of the cells are wrongly labeled (Figure 3D). The number of rejected cells, on the other hand, is not considered when calculating the median F1-score. This number is relatively high for $\text{SVM}_{\text{rejection}}$ (19.8%). The linear SVM, on the contrary, has almost no rejected cells, which is also reflected in a higher HF1-score (Figure 3C). Because of this large amount of rejections of $\text{SVM}_{\text{rejection}}$, the one-class SVM also has a higher HF1-score.

On the AMB dataset, we observe that the performance of all classifiers decreases when the number of cells per cell population becomes smaller. The performance of the one-class SVM is affected more than the others (Figure 3F). The one-class SVM, for instance, never predicts the 'Astro Aqp4' cells correctly, while this population is relatively different from the rest as it is the only non-neuronal population. This cell population, however, only consists of eleven cells.

In the previous experiments, we used all genes to train the classifiers. Since the selection of highly variable genes (HVGs) is common in scRNA-seq analysis pipelines, we tested the effect of selecting HVGs on the classification performance of the PBMC-FACS dataset. We noted that using all genes results in the highest HF1-score for both the linear and one-class SVM (Figure S4).

3.2.3 One-class SVM detects new cells better than linear SVM

Besides a high accuracy, the classifiers should be able to reject unseen cell populations. First, we evaluated the rejection option on the simulated data. In this dataset, the cell populations are distinct, so we expect that this is a relatively easy task. We removed one cell population, 'Group 3', from the training set and used this population as a test set. The one-class SVM outperforms the linear SVM as it correctly rejects all these cells, while the linear SVM rejects only 38.9% of them.

Next, we tested the rejection option on the AMB dataset. Here, we did four experiments and each time removed a node, including all its subpopulations, from the predefined tree (Figure S3). We removed the 'L6 IT' and 'Lamp5' cell populations from the second layer of the tree, and the 'L6 IT VISp Penk Col27a1' and 'Lamp5 Lsp1' from the third layer. When a node is removed from the second layer of the tree, the linear SVM clearly rejects these cells better than the one-class SVM (Figure 3E). On the contrary, the one-class SVM rejects leaf node cells better.

3.2.4 scHPL accurately learns cellular hierarchies

Next, we tested if we could learn the classification trees for the simulated and PBMC-FACS data using scHPL. In both experiments, we performed a 10-fold cross-validation and splitted the training set in three different batches, Batch 1, Batch 2, and Batch 3, to simulate the idea of different datasets. To obtain different annotation levels in these batches, multiple cell populations were merged into one population in some batches, or cell populations were removed from certain batches (Tables S2-3). Batch 1 contains the lowest resolution and Batch 3 the highest. When learning the trees, we try all (six) different orders of the batches to see whether this affects the tree learning. Combining this with the 10-fold cross-validation, 60 trees were learned in total by each classifier. To summarize the results, we constructed a final tree in which the thickness of an edge indicates how often it appeared in the 60 learned trees.

The linear and one-class SVM showed stable results during both experiments; all 60 trees- except for two trees learned by the one-class SVM on the PBMC data- look identical (Figure 4A-D). The final tree for the simulated data looks as expected, but the tree for the PBMC data looks slightly different from the predefined hematopoietic tree (Figure S2A). In the learned trees, CD4+ memory T-cells are a subpopulation of CD8+ instead of CD4+ T-cells. The correlation between the centroids of CD4+ memory T-cell and CD8+ T-cells (r = 0.985±0.003) is also slightly higher than the correlation to CD4+ T-cells (r = 0.975±0.002) (Figure S5). Using the learned tree instead of the predefined hematopoietic tree improves the classification performance of the linear SVM slightly (HF1-score = 0.990 vs 0.985). Moreover, when relying

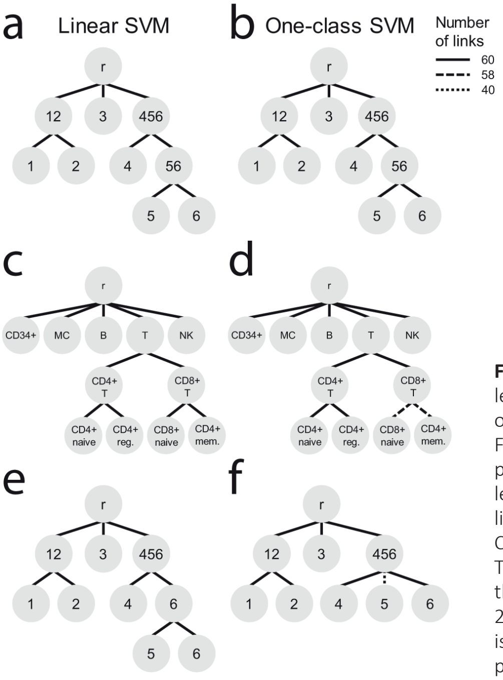


Figure 4. Tree learning evaluation. Classification trees learned when using a **A, C, E)** linear SVM or **B, D, F)** one-class SVM during the **A, B)** simulated, **C, D)** PBMC-FACS, and **E, F)** simulated rejection experiment. The line pattern of the links indicates how often that link was learned during the 60 training runs. **D)** In 2/60 trees, the link between the CD8+ T-cells and the CD8+ naive and CD4+ memory T-cells is missing. In those trees, the CD8+ T-cells and CD8+ naive T-cells have a perfect match and the CD4+ memory T-cells are missing from the tree. **F)** In 20/60 trees, the link between 'Group456' and 'Group5' is missing. In those trees, these two populations are a perfect match.

on the predefined hematopoietic tree, CD4+ memory T-cells, CD8+ T-cells, and CD8+ naive T-cells were also often confused, further highlighting the difficulty in distinguishing these populations based on their transcriptomic profiles alone (Tables S4-5).

Next, we tested the effect of the matching threshold (default = 0.25) on the tree construction by varying this to 0.1 and 0.5. For the linear SVM, changing the threshold had no effect. For the one-class SVM, we noticed a small difference when changing the threshold to 0.1. The two trees that were different using the default threshold (Figure 4D), were now constructed as the other 58 trees. In general, scHPL is robust to settings of the matching threshold due to its reliance on reciprocal classification.

3.2.5 Missing populations affect tree construction with linear SVM

We tested whether new or missing cell populations in the training set could influence tree learning. We mimicked this scenario using the simulated dataset and the same batches as in the previous tree learning experiment. In the previous experiment, ‘Group5’ and ‘Group6’ were merged into ‘Group56’ in Batch 2, but now we removed ‘Group5’ completely from this batch (Table S6). In this setup, the linear SVM misconstructs all trees (Figure 4E). If ‘Group5’ is present in one batch and absent in another, the ‘Group5’ cells are not rejected, but labeled as ‘Group6’. Consequently, ‘Group6’ is added as a parent node to ‘Group5’ and ‘Group6’. On the other hand, the one-class SVM suffers less than the linear SVM from these missing populations and correctly learns the expected tree in 2/3 of the cases (Figure 4F). In the remaining third (20 trees), ‘Group5’ matched perfectly with ‘Group456’ and was thus not added to the tree. This occurs only if we have the following order: Batch 1- Batch 3- Batch 2 or Batch 3- Batch 1- Batch 2. Adding batches in increasing or decreasing resolution consequently resulted in the correct tree.

3.2.6 Linear SVM can learn the classification tree during an inter-dataset experiment

Finally, we tested scHPL in a realistic scenario by using three PBMC datasets (PBMC-eQTL, PBMC-Bench10Xv2, and PBMC-FACS) to learn a classification tree and using this tree to predict the labels of a fourth PBMC dataset (PBMC-Bench10Xv3) (Table 1). Before applying scHPL, we aligned the datasets using Seurat [28]. We constructed an expected classification tree based on the names of the cell populations in the datasets (Figure 5A). Note that matching based on names might result in an erroneous tree since every dataset was labeled using different clustering techniques, marker genes, and their own naming conventions.

When comparing the tree learned using the linear SVM to the expected tree, we notice five differences (Figure 5A-B). Some of these differences are minor, such as the matching of monocytes from the Bench10Xv2 dataset to myeloid dendritic cells (mDC), CD14+ monocytes, and the CD16+ monocytes. Monocytes can differentiate into mDC which can explain their transcriptomic similarity [29]. Other differences between the reconstructed and the expected trees are likely the result of (partly) mislabeled cell populations in the

Table 1. Number of cells per cell population in the different training datasets (batches) and test dataset. Subpopulations are indicated using an indent.

Cell population	Batch 1 eQTL	Batch 2 Bench 10Xv2	Batch 3 FACS	Test dataset Bench 10Xv3
CD19+ B	812	676	2,000	346
CD34+			2,000	
Monocytes (MC)		1,194		
CD14+	2,081		2,000	354
CD16+	274			98
CD4+ T	13,523	1,458		960
Reg.			2,000	
Naive			2,000	
Memory			2,000	
CD8+ T	4,195	2,128		962
Naive			2,000	
Megakaryocyte (MK)	142	433		270
NK cell		429	2,000	194
CD56+ bright	355			
CD56+ dim	2,415			
Dendritic				35
Plasmacytoid (pDC)	101			
Myeloid (mDC)	455			

original datasets (Figure S6-15). (*i*) According to the expression of *FCER1A* (a marker for mDC) and *FCGR3A* (CD16+ monocytes), the labels of the mDC and the CD16+ monocytes in the eQTL dataset are reversed (Figure S6-8). (*ii*) Part of the CD14+ monocytes in the FACS dataset express *FCER1A*, which is a marker for mDC (Figure S6, S8-9). The CD14+ monocytes in the FACS dataset are thus partly mDCs, which explains the match with the mDC from the eQTL dataset. (*iii*) Part of the CD4+ T-cells from the eQTL and Bench10Xv2 dataset should be relabeled as CD8+ T-cells (Figure S6, S10-13). In these datasets, the cells labeled as the CD8+ T-cells only contain cytotoxic CD8+ T-cells, while naive CD8+ T-cells are all labeled as CD4+ T-cells. This mislabeling explains why the CD8+ naive T-cells are a subpopulation of the CD4+ T-cells. (*iv*) Part of the CD34+ cells in the FACS dataset should be relabeled as pDC (Figure S6, S14-15), which explains why the pDC are a subpopulation of the CD34+ cells. In the FACS dataset, the labels were obtained using sorting, which would indicate that these labels are correct. The purity of the CD34+ cells, however, was significantly low (45%) compared to other cell populations (92-100%) [27]. There is only one difference , however, that cannot be explained by mislabeling. The NK cells from the FACS dataset do not only match the NK cells from the eQTL dataset, but also the CD8+ T-cells.

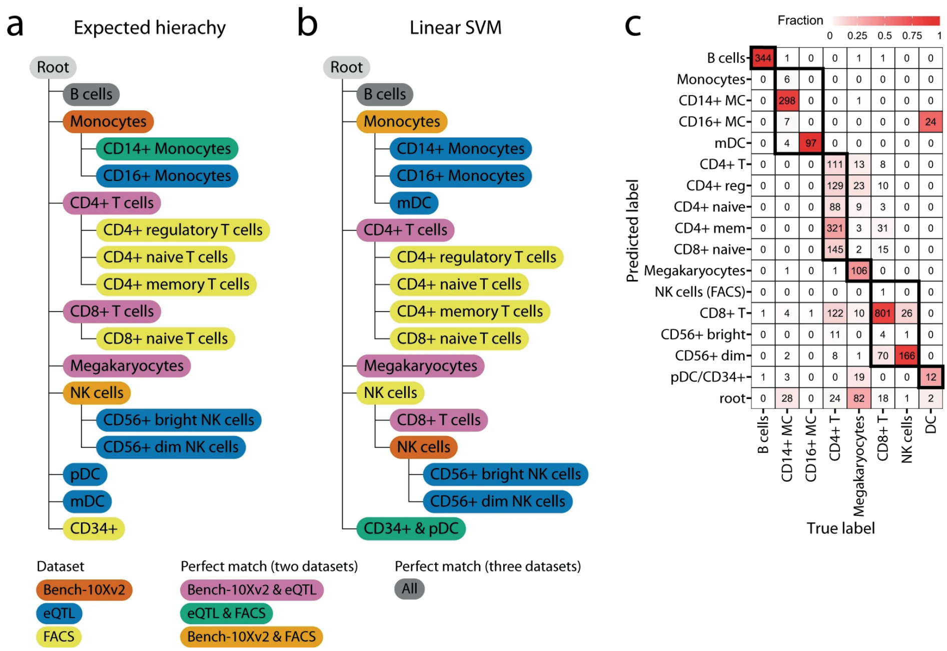


Figure 5. PBMC inter-dataset evaluation. A) Expected and **B) l**earned classification tree when using a linear SVM on the PBMC datasets. The color of a node represents the agreement between dataset(s) regarding that cell population. **C)** Confusion matrix when using the learned classification tree to predict the labels of PBMC-Bench10Xv3. The dark boundaries indicate the hierarchy of the constructed classification tree.

Most cells of the Bench10Xv3 dataset can be correctly annotated using the learned classification tree (Figure 5E). Interestingly, we notice that the CD16+ monocytes are predicted to be mDCs and vice versa, which could be explained by the fact that the labels of the mDCs and the CD16+ monocytes were flipped in the eQTL dataset. Furthermore, part of the CD4+ T-cells are predicted to be CD8+ naïve T-cells. In the Bench10Xv3, we noticed the same mislabeling of part of the CD4+ T-cells as in the eQTL and Bench10Xv2 datasets, which supports our predictions (Figure S6, S10-13).

The tree constructed using the one-class SVM differs slightly compared to the linear SVM (Figure S16A). Here, the monocytes from the Bench10Xv2 match the CD14+ monocytes and mDC (which are actually CD16+ monocytes) as we would expect. Next, the CD14+ monocytes from the FACS dataset merge the CD16+ monocytes (which are actually mDC) and the monocytes. Using the one-class SVM the CD8+ T-cells and NK cells from the Bench10Xv2 dataset are missing since there was a complex scenario. The NK cells are a relatively small population in this dataset which made it difficult to train a classifier for this population.

In the previous experiments, we used the default setting of Seurat to align the datasets (using 2000 genes). We tested whether changing the number of genes to 1000 and 5000 affects the performance. When using the one-class SVM, the number of genes does not affect tree construction. For the linear SVM, we notice one small difference when using 1000 genes: the CD8+ T-cells from the Bench10Xv2 dataset are a subpopulation of the CD8+ T-cells from the eQTL dataset instead of a perfect match.

The predicted labels of the Bench10Xv3 dataset change slightly when using a different number of genes (Figure S17). Whether more genes improves the prediction, differs per cell population. The labels of the megakaryocytes, for instance, are better predicted when more genes are used, but for the dendritic cells we observe the reverse pattern.

3.2.7 Mapping brain cell populations using scHPL

Next, we applied scHPL to construct a tree which maps the relationships between brain cell populations. This is a considerably more challenging task compared to PBMCs given the large number of cell populations as well as the fact that brain cell types are not consistently annotated. First, we combined two datasets from the primary visual cortex of the mouse brain, AMB2016 and AMB2018 [4,5]. AMB2018 contains more cells (12,771 vs. 1,298) and is clustered at a higher resolution (92 cell populations vs. 41) compared to AMB2016. Before applying scHPL, we aligned the datasets using Seurat [28]. Using scHPL with a linear SVM results in an almost perfect tree (Figure 6). We verified these results by comparing our constructed tree to cluster correspondences in Extended Data Fig. 6 from Tasic et al. [5]. Since AMB2018 is clustered at a higher resolution, most populations are subpopulations of AMB2016, which are all correctly identified by scHPL. Conversely, three L4 populations from AMB2016 were merged into one population (L4 IT VISp Rspo1) from AMB2018 [5], forming a continuous spectrum. This relation was also automatically identified using scHPL (Figure 6). Compared to the results from Tasic et al. [5], one cell population from AMB2018 is attached to a different parent node. scHPL assigned ‘L6b VISp Col8a1 Rprm’ as a subpopulation of ‘L6a Sla’ instead of ‘L6b Rgs12’. This population, however, does not express *Rgs12*, but does express *Sla* (Figure S18), supporting the matching identified by scHPL. Three cell populations could not be added to the tree due to complex scenarios. According to Extended Data Fig. 6 from Tasic et al. [5], these AMB2018 populations are a subpopulation of multiple AMB2016 subpopulations.

The AMB2016 and AMB2018 datasets were generated and analyzed by the same group and hence the cluster matching is certainly easier than a real-life scenario. Therefore, we tested scHPL also on a complicated scenario with brain datasets that are sequenced using different protocols and by different labs (Table S7, Figure S19). We used three datasets (Zeisel, Tabula Muris, and Saunders) to construct the tree (Figure 7A-D) [2,30,31]. The Zeisel dataset is annotated at two resolutions. Before applying scHPL, we aligned the datasets using Seurat [28]. First, we constructed a tree using a linear SVM based on the low resolution of Zeisel. We started with the Saunders dataset and added Zeisel (Figure 7E). This is a clear illustration of the possible scenarios scHPL can manage. Some populations are a perfect match between the two datasets (e.g. neurons), some populations from Saunders are splitted (e.g. astrocytes), some are merged (e.g. macrophages and microglia), and some populations from Zeisel have no match (e.g. Ttr). Next, we updated the tree by adding the Tabula Muris dataset (Figure 7F). Here, we found matches that would not have been possible to identify by relying on the assigned cell type labels to map cell types. For example, mural cells from Saunders are a perfect match with the brain pericytes from the Tabula Muris. The results of scHPL with the one-class SVM were almost identical to the linear SVM (Figure S20A).

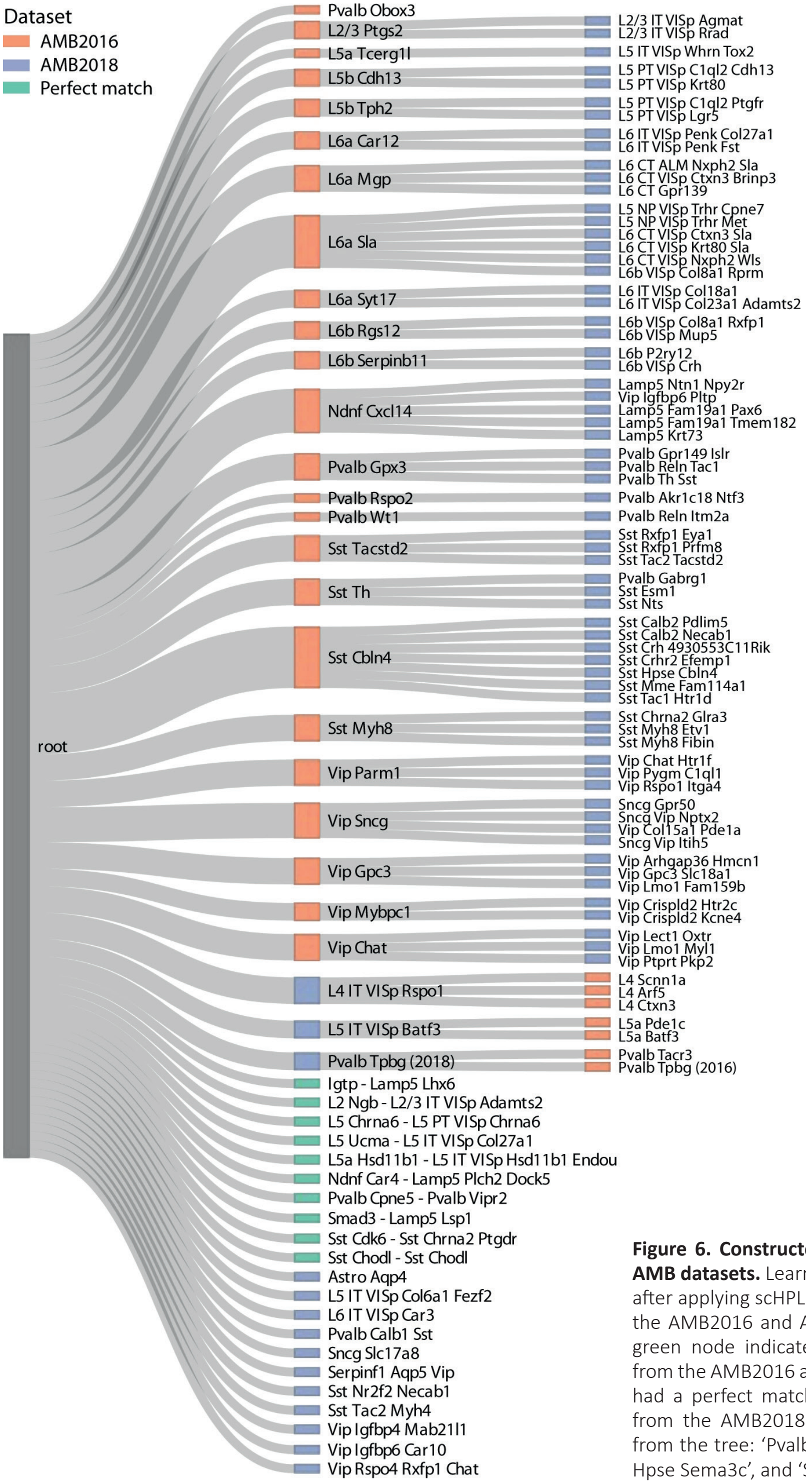


Figure 6. Constructed hierarchy for the AMB datasets. Learned classification tree after applying scHPL with a linear SVM on the AMB2016 and AMB2018 datasets. A green node indicates that a population from the AMB2016 and AMB2018 dataset had a perfect match. Three populations from the AMB2018 dataset are missing from the tree: 'Pvalb Sema3e Kank4', 'Sst Hpse Sema3c', and 'Sst Tac1 Tacr3'.

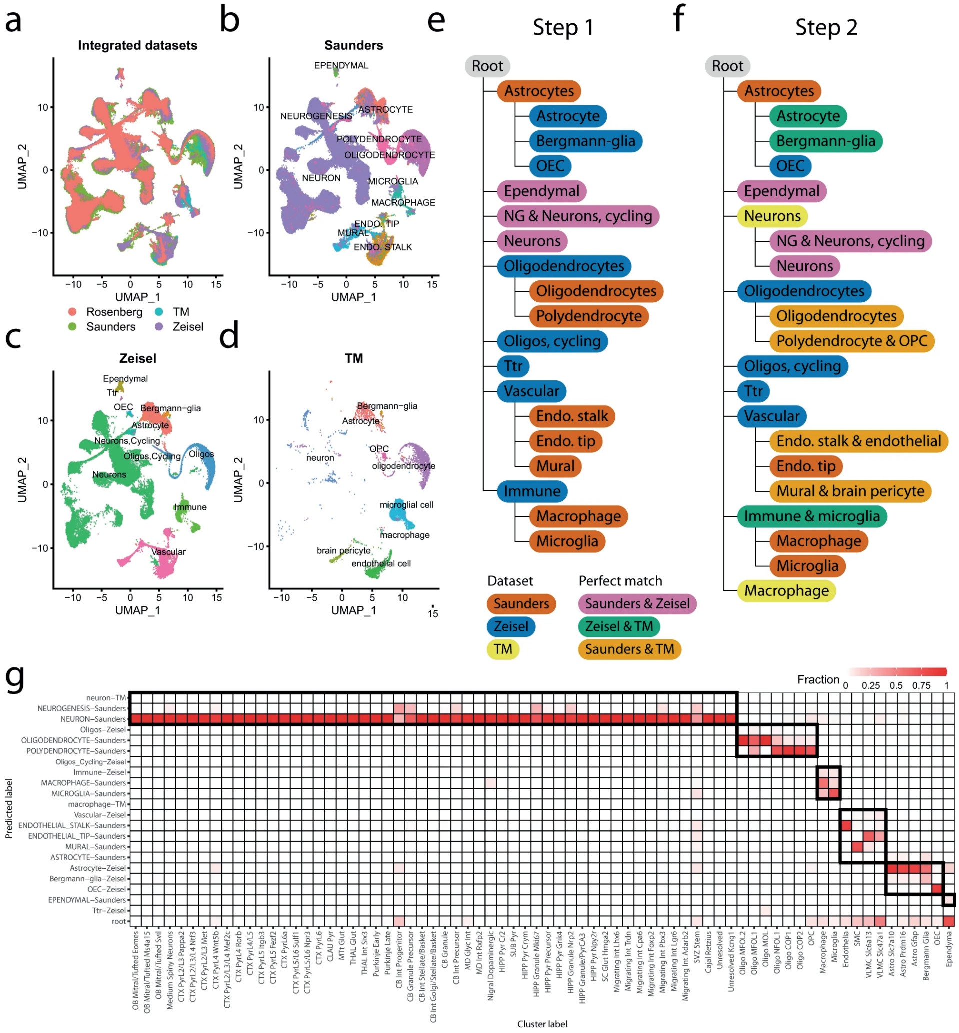


Figure 7. Brain inter-dataset evaluation. A-D) UMAP embeddings of the datasets after alignment using Seurat v3. **E)** Learned hierarchy when starting with the Saunders dataset and adding Zeisel with linear SVM. **F)** Updated tree when the Tabula Muris dataset is added. **G)** Confusion matrix when using the learned classification tree to predict the labels of Rosenberg. The dark boundaries indicate the hierarchy of the classification tree.

Next, we used the resulting tree to predict the labels of a fourth independent dataset (Rosenberg) [32]. The predictions from the linear and the one-class SVM are very similar (Figure 7G, S20B). The only difference is that the linear SVM correctly predicts some progenitor or precursor neuronal populations from Rosenberg to be 'neurogenesis' while the one-class SVM rejects these populations.

To assess the effect of the annotation resolution, we repeated the analysis using the higher resolution annotation from the Zeisel dataset (Figure S21-23). Here, we noticed that the 'brain pericytes (TM)' and 'pericytes (Zeisel)'- two populations one would easily match based on the names only- are not in the same subtree. 'Brain pericyte (TM)' forms a perfect match with ‘mural (Saunders)’ and ‘vascular smooth muscle cells (Zeisel)’, while ‘pericytes (Zeisel)’ is a subpopulation of ‘endothelial stalk (Saunders)’ and ‘endothelial cell (TM)’ (Figure S22-23). In the UMAP embedding of the integrated datasets, the ‘pericytes’ and ‘brain pericyte’ are at a different location, but they do overlap with the cell populations they were matched with (Figure S21). This highlights the power of scHPL matching rather than name-based matching.

3.3 Discussion

In this study, we showed that scHPL can learn cell identities progressively from multiple reference datasets. We showed that using our approach the labels of two AMB datasets can successfully be matched to create a hierarchy containing mainly neuronal cell populations and that we can combine three other brain datasets to create a hierarchy containing mainly non-neuronal cell populations. In both experiments, we discovered new relationships between cell populations, such as the mapping of ‘L6b VISp Col8a1 Rprm’ as a subpopulation of ‘L6b Sla’ instead of ‘L6b Rgs12’. This observation would not be possible to make by manually matching populations based on the assigned labels, highlighting the power of automatically constructing cellular hierarchies. In this case, the Cell Ontology database could also not be used to verify this relationship since many brain cell populations are missing. Most of these populations have recently been annotated using scRNA-seq and there is a wide lack of consistency in population annotation and matching between studies [18]. scHPL can potentially be used to map these relations, irrespective of the assigned labels, and improve the Cell Ontology database.

When combining multiple datasets to construct a tree, we expect that cell populations are annotated correctly. However, in the PBMC inter-dataset experiment, this was not the case. At first sight, the constructed tree looked erroneous, but the expression of marker genes revealed that (parts of) several cell populations were mislabeled. Here, we could use the constructed tree as a warning that there was something wrong with the original annotations.

In general, scHPL is robust to sampling differences between datasets or varying parameters such as the matching threshold or the number of genes used. The brain datasets used to construct the tree, for instance, varied greatly in population sizes, which did not cause any difficulties. This is mainly because we rely on reciprocal classification. A match between cell populations that is missed when training a classifier on one dataset to predict labels of the other, can still be captured by the classifier trained on the other dataset.

Since batch effects are inevitable when combining datasets, we require datasets to be aligned before running scHPL. In all inter-dataset experiments in this manuscript, we used Seurat V3 [28] for the alignment, but we would like to emphasize that scHPL is not dependent on Seurat and can be combined with any batch correction tool, such as more computationally efficient methods like Harmony [33]. A current limitation of these tools is that when a new dataset is added, the alignment- and consequently also scHPL- has to be rerun. An interesting alternative would be to project the new dataset to a latent space learned using reference dataset(s), using scArches [34] for example. In that case, scHPL does not have to be rerun but can be progressively updated.

The batch effects between the datasets make it more difficult to troubleshoot errors. Generally, it will be hard to resolve whether mistakes in the constructed tree are caused by the erroneous alignment of datasets or by mismatches created by scHPL.

We would like to note though that there are inherent limitations to the assumption that cell populations have hierarchical relationships. While this assumption is widely adopted in single cell studies as well as the Cell Ontology, there are indeed situations in which a tree is not adequate. For instance, situations in which cells dedifferentiate into other cell types, such as beta to alpha cell conversions in type2 diabetes [35,36].

Considering the classification performance, we showed that using a hierarchical approach outperforms flat classification. On the AMB dataset, the linear SVM outperformed $\text{SVM}_{\text{rejection}}$, which was the best performing classifier on this dataset [23]. In contrast to $\text{SVM}_{\text{rejection}}$, the linear SVM did not reject any of the cells but labeled them as an intermediate cell population. During this experiment, there were no cells of unknown populations. Correct intermediate predictions instead of rejection are therefore beneficial since it provides the user with at least some information. When comparing the linear SVM and one-class SVM, we noticed that the accuracy of the linear SVM is equal to or higher than the one-class SVM on all datasets. For both classifiers, we saw a decrease in performance on populations with a small number of cells, but for the one-class SVM this effect was more apparent.

Since the one-class SVM has a low performance on small cell populations, it also cannot be used to combine datasets which consist of small populations. If the classification performance is low, it will also not be possible to construct the correct tree. On the other hand, the performance of the linear SVM seems to be robust to small populations throughout our experiments. This classifier can thus better be used when combining multiple datasets with small populations.

When testing the rejection option, the one-class SVM clearly outperforms the linear SVM by showing a perfect performance on the simulated dataset. Moreover, when cell populations are missing from the simulated data, the linear SVM cannot learn the correct tree anymore, in contrast to the one-class SVM. This suggests that the one-class SVM is preferred when cell populations are missing, although on the AMB dataset, the rejection option of both classifiers was not perfect.

In summary, we present a hierarchical progressive learning approach to automatically identify cell identities based on multiple datasets with various levels of subpopulations. We show that we can accurately learn cell identities and learn hierarchical relations between cell populations. Our results indicate that choosing between a one-class and a linear SVM is a trade-off between achieving a higher accuracy and the ability to discover new cell populations. Our approach can be beneficial in single-cell studies where a comprehensive reference atlas is not present, for instance, to annotate datasets consistently during a cohort study. The first available annotated datasets can be used to build the hierarchical tree, which could subsequently can be used to annotate cells in the other datasets.

3.4 Methods

3.4.1 Hierarchical progressive learning

Within scHPL, we use a hierarchical classifier instead of a flat classifier. A flat classifier is a classifier that doesn't consider a hierarchy and distinguishes between all cell populations simultaneously. For the AMB dataset, a flat classifier will have to learn the decision boundaries between all 92 cell populations in one go. Alternatively, a hierarchical classifier divides the problem into smaller subproblems. First it learns the difference between the 3 broad classes: GABAergic neurons, glutamatergic neurons, and non-neuronal cells. Next, it learns the decision boundaries between the six subtypes of GABAergic neurons and the eight subtypes of glutamatergic neurons, separately. Finally, it will learn the decision boundaries between the different cell populations within each subtype separately.

3.4.2 Training the hierarchical classifier

The training procedure of the hierarchical classifier is the same for every tree: we train a local classifier for each node except the root. This local classifier is either a one-class SVM or a linear SVM. We used the one-class SVM (`svm.OneClassSVM(nu = 0.05)`) from the scikit-learn library in Python [37]. A one-class classifier only uses positive training samples. Positive training samples include cells from the node itself and all its child nodes. To avoid overfitting, we select the first 100 principal components (PCs) of the training data. Next, we select informative PCs for each node separately using a two-sided two-sample t-test between the positive and negative samples of a node ($\alpha < 0.05$, Bonferroni corrected). Negative samples are selected using the siblings policy [38], i.e. sibling nodes include all nodes that have the same ancestor, excluding the ancestor itself. If a node has no siblings, cells labeled as the parent node, but not the node itself are considered negative samples. In some rare cases, the Bonferroni correction was too strict and no PCs were selected. In those cases, the five PCs with the smallest p-values were selected. For the linear SVM, we used the `svm.LinearSVC()` function from the scikit-learn library. This classifier is trained using positive and negative samples. The linear SVM applies L2-regularization by default, so no extra measures to prevent overtraining were necessary.

3.4.3 The reconstruction error

The reconstruction error is used to reject unknown cell populations. We use the training data to learn a suitable threshold which can be used to reject cells by doing a nested 5 fold cross-validation. A PCA ($n_{components} = 100$) is learned on the training data. The test data is then reconstructed by first mapping the data to the selected PCA domain, and then mapping the data back to the original space using the inverse transformation (hence the data lies within the plane spanned by the selected PCs). The reconstruction error is the difference between the original data and the reconstructed data (in other words, the distance of the original data to the PC plane). The median of the q^{th} (default $q = 0.99$) percentile of the errors across the test data is used as threshold. By increasing or decreasing this parameter, the number of false negatives can be controlled. Finally, we apply a PCA ($n_{components} = 100$) to the whole dataset to learn the transformation that can be applied to new unlabeled data later.

3.4.4 Predicting the labels

First, we look at the reconstruction error of a new cell to determine whether it should be rejected. If the reconstruction error is higher than the threshold determined on the training data, the cell is rejected. If not, we continue with predicting its label. We start at the root node, which we denote as parent node and use the local classifiers of its children to predict the label of the cell using the `predict()` function, and score it using the `decision_function()`, both from the scikit-learn package. These scores represent the signed distance of a cell to the decision boundary. When comparing the results of the local classifiers, we distinguish three scenarios:

1. All child nodes label the cell negative. If the parent node is the root, the new cell is rejected. Otherwise we have an internal node prediction and the new cell is labeled with the name of the parent node.
2. One child node labels the cell positive. If this child node is a leaf node, the sample is labeled with the name of this node. Otherwise, this node becomes the new parent and we continue with its children.
3. Multiple child nodes label the cell positive. We only consider the child node with the highest score and continue as in scenario two.

3.4.5 Reciprocal matching labels and updating the tree

Starting with two datasets, *D1* and *D2*, and the two corresponding classification trees (which can be either hierarchical or flat), we would like to match the labels of the datasets and merge the classification trees accordingly into a new classification tree while being consistent with both input classification trees (Figure 1). We do this in two steps: first matching the labels between the two dataset and then updating the tree.

Reciprocal matching labels. We first cross-predict the labels of the datasets: we use the classifier trained on *D1* to predict the labels of *D2* and vice versa. We construct confusion matrices, *C1* and *C2*, for *D1* and *D2*, respectively. Here, $C1_{ij}$ indicates how many cells of population *i* of *D1* are predicted to be population *j* of *D2*. This prediction can be either a leaf node, internal node or a rejection. As the values in *C1* and *C2* are highly dependent on the size of a cell population, we normalize the rows such that the sum of every row is one, now indicating the fraction of cells of population *i* in *D1* that have been assigned to population *j* in *D2*:

$$NC1_{ij} = \frac{C1_{ij}}{\sum_{\forall j} C1_{ij}}$$

Clearly, a high fraction is indicative of matching population *i* in *D1* with population *j* in *D2*. Due to splitting, merging, or new populations between both datasets, multiple relatively high fractions can occur (e.g. if a population *i* is split in two populations j_1 and j_2 due to *D2* being of a higher resolution, both fractions NC_{ij1} and NC_{ij2} will be approximately 0.5). To accommodate for these operations, we allow multiple matches per population.

To convert these fractions into matches, *NC1* and *NC2* are converted into binary confusion matrices, *BC1* and *BC2*, where a 1 indicates a match between a population in *D1* with a population in *D2*, and vice versa. To determine a match, we take the value of the fraction as well as the difference with the other fractions into account. This is done for each row (population) of *NC1* and *NC2* separately. When considering row *i* from *NC1,* we first rank all fractions, then the highest fraction will be set to 1 in *BC1*, after which all fractions for which the difference with the preceding (higher) fraction is less than a predefined threshold (default = 0.25) will also be set to 1 in *BC1*.

To arrive at reciprocal matching between *D1* and *D2,* we combine *BC1* and *BC2* into matching matrix *X* (Figure 2):

$$X = BC1^T + BC2$$

The columns in *X* represent the cell populations of *D1* and the rows represent the cell populations of *D2*. If $X_{ij} = 2$, this indicates a reciprocal match between cell population *i* from *D2* and cell populations *j* from *D1*. $X_{ij} = 1$ indicates a one-sided match, and $X_{ij} = 0$ represents no match.

Tree updating. Using the reciprocal matches between *D1* and *D2* represented in *X*, we update the hierarchical tree belonging to *D1* to incorporate the labels and tree structure of *D2*. We do that by handling the correspondences in *X* elementwise. For a non-zero value in *X*, we check whether there are other non-zero values in the corresponding row and column to identify which tree operation we need to take (such as split/merge/create). As an example, if we encounter a split for population *i* in *D1* into j_1 and j_2, we will create new nodes for j_1 and j_2 as child nodes of node *i* in the hierarchical tree of *D1*. Figure 2 and Table S1 explain the four most common scenarios: a perfect match, splitting nodes, merging nodes, and a new population. All other scenarios are explained in Supplementary Note 1. After an update, the corresponding values in *X* are set to zero and we continue with the next non-zero element of *X*. If the matching is impossible, the corresponding values in *X* are thus not set to zero. If we have evaluated all elements of *X*, and there are still non-zero values, we will change *X* into a strict matrix, i.e. we further only consider reciprocal matches, so all '1's are turned into a '0' with some exceptions (Supplementary Note 2). We then again evaluate *X* element wise once more.

3.4.6 Evaluation

Hierarchical F1-score. We use the hierarchical F1-score (HF1-score) to evaluate the performance of the classifiers [39]. We first calculate the hierarchical precision (*hP*) and recall (*hR*):

$$hP = \frac{\sum_i P_i \cap T_i}{\sum_i P_i} \qquad hR = \frac{\sum_i P_i \cap T_i}{\sum_i T_i}$$

Here, P_i is a set that contains the predicted cell population for a cell *i* and all the ancestors of that node, T_i contains the true cell population and all its ancestors, and $P_i \cap T_i$ is the overlap between these two sets. The HF1-score is the harmonic mean of *hP* and *hR*:

$$\mathrm{HF1} = \frac{2hP*2hR}{hP+hR}$$

Median F1-score. We use the median F1-score to compare the classification performance to other methods. The F1-score is calculated for each cell population in the dataset and afterwards the median of these scores is taken. Rejected cells and internal predictions are not considered when calculating this score.

3.4.7 Datasets

Simulated data. We used the R-package Splatter (V1.6.1) to simulate a hierarchical scRNA-seq dataset that consists of 8,839 cells and 9,000 genes and represents the tree shown in Figure S1A (Supplementary Note 3) [26]. We chose this low number of genes to speed up the computation time. In total there are six different cell populations of approximately 1,500 cells each. As a preprocessing step, we log-transformed the count matrix ($\log_2(count+1)$). A UMAP embedding of the simulated dataset shows it indeed represents the desired hierarchy (Figure S1C).

Peripheral Blood Mononuclear Cells (PBMC) scRNA-seq datasets. We used four different PBMC datasets: PBMC-FACS, PBMC-Bench10Xv2, PBMC-Bench10Xv3, and PBMC-eQTL. The PBMC-FACS dataset is the downsampled FACS-sorted PBMC dataset from Zheng et al. [27]. Cells were first FACS-sorted into ten different cell populations (CD14+ monocytes, CD19+ B cells, CD34+ cells, CD4+ helper T-cells, CD4+/CD25+ regulatory T-cells, CD4+/CD45RA+/CD25− naive T-cells, CD4+/CD45RO+ memory T-cells, CD56+ natural killer cells, CD8+ cytotoxic T-cells, CD8+/CD45RA+ naive cytotoxic T-cells) and sequenced using 10X Chromium [27]. Each cell population consists of 2,000 cells. The total dataset consists of 20,000 cells and 21,952 genes. During the cross-validation on the PBMC-FACS dataset, we tested the effect of selecting HVG. We used the 'seurat_v3' flavor of scanpy to select 500, 1000, 2000, and 5000 HVG on the training set [28,40]. The PBMC-Bench10Xv2 and PBMC-Bench10Xv3 datasets are the PbmcBench pbmc1.10Xv2 and pbmc1.10Xv3 datasets from Ding et al. [41]. These datasets consist of 6,444 and 3,222 cells respectively, 22,280 genes, and nine different cell populations. Originally the PBMC-Bench10Xv2 dataset contained CD14+ and CD16+ monocytes. We merged these into one population called monocytes to introduce a different annotation level compared to the other PBMC datasets.The PBMC-eQTL dataset was sequenced using 10X Chromium and consists of 24,439 cells, 22,229 genes, and eleven different cell populations [42].

Brain scRNA-seq datasets. We used two datasets from the mouse brain, AMB2016 and AMB2018, to look at different resolutions of cell populations in the primary mouse visual cortex. The AMB2016 dataset was sequenced using SMARTer [4], downloaded from https://portal.brain-map.org/atlases-and-data/rnaseq/data-files-2018. AMB2016 consists of 1,298 cells and 21,413 genes. The AMB2018 dataset, which was sequenced using SMART-Seq V4 [5], downloaded from https://portal.brain-map.org/atlases-and-data/rnaseq/mouse-v1-

and-alm-smart-seq, consists of 12,771 cells and 42,625 genes. Additionally, we used four other brain datasets: Zeisel [2], Tabula Muris [30], Rosenberg [32], and Saunders [31]. These were downloaded from the scArches 'data' Google Drive ('mouse_brain_regions.h5ad' from https://drive.google.com/drive/folders/1QQXDuUjKG8CTnwWW_u83MDtdrBXr8Kpq) [34]. We downsampled each dataset such that at the highest resolution each cell population consisted of up to 5,000 cells to reduce the computational time for the alignment (Table S7).

Preprocessing scRNA-seq datasets. All datasets were preprocessed as described in Abdelaal et al. [23]. Briefly, we removed cells labeled in the original studies as doublets, debris or unlabeled cells, cells from cell populations with less than 10 cells, and genes that were not expressed. Next, we calculated the median number of detected genes per cell, and from that, we obtained the median absolute deviation (MAD) across all cells in the log scale. We removed cells when the total number of detected genes was below three MAD from the median number of detected genes per cell. During the intra-dataset experiments, we log-transformed the count matrices ($\log_2(count+1)$).

Aligning scRNA-seq datasets. During the inter-dataset experiments, we aligned the datasets using Seurat V3 [28] based on the joint set of genes expressed in all datasets. In the PBMC, AMB, and brain inter-dataset experiment respectively 17,573, 19,197, and 14,858 genes remained. For the PBMC inter-dataset experiment, we also removed cell populations that consisted of less than 100 cells from the datasets used for constructing and training the classification tree (PBMC-eQTL, FACS, Bench10Xv2). To test the effect of the number of genes on scHPL, we integrated this data using 1000, 2000 (default), and 5000 HVGs.

3.5 Code and data availability

The filtered PBMC-FACS and AMB2018 dataset can be downloaded from Zenodo (https://doi.org/10.5281/zenodo.3357167). The simulated dataset and the aligned datasets used during the inter-dataset experiment can be downloaded from Zenodo (http://doi.org/10.5281/zenodo.3736493). Accession numbers or links to the raw data: AMB2016 [4] (GSE71585, https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE71585), AMB2018 [5] (GSE115746, https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE115746), PBMC-FACS [27] (SRP073767, https://support.10xgenomics.com/single-cell-gene-expression/datasets), PBMC-eQTL [42] (EGAS00001002560, https://ega-archive.org/studies/EGAS00001002560), PBMC-Bench10Xv2 and PBMC-Bench10Xv3 [41] (GSE132044, https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE132044), Rosenberg [32] (GSE110823, https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE110823), Zeisel [2] (http://mousebrain.org, file name L5_all.loom, downloaded on 9/9/2019), Saunders [31] (http://dropviz.org, DGE by Region section, downloaded on 30/8/2019), Tabula Muris [30] (https://figshare.com/projects/Tabula_Muris_Transcriptomic_characterization_of_20_organs_and_tissues_from_Mus_musculus_at_single_cell_resolution/27733, downloaded on 14/2/2019). The source code for scHPL is available as a python package that is installable through the PyPI repository (https://github.com/lcmmichielsen/scHPL) [43].

Bibliography

1. van der Wijst MG, de Vries DH, Groot HE, Trynka G, Hon C-C, Bonder M-J, et al. The single-cell eQTLGen consortium. Elife. 2020;9. doi:10.7554/eLife.52155
2. Zeisel A, Hochgerner H, Lönnerberg P, Johnsson A, Memic F, van der Zwan J, et al. Molecular Architecture of the Mouse Nervous System. Cell. 2018;174: 999–1014.e22. doi:10.1016/j.cell.2018.06.021
3. Svensson V, da Veiga Beltrame E, Pachter L. A curated database reveals trends in single-cell transcriptomics. Database. 2020;2020. doi:10.1093/database/baaa073
4. Tasic B, Menon V, Nguyen TN, Kim TK, Jarsky T, Yao Z, et al. Adult mouse cortical cell taxonomy revealed by single cell transcriptomics. Nat Neurosci. 2016;19: 335–346. doi:10.1038/nn.4216
5. Tasic B, Yao Z, Graybuck LT, Smith KA, Nguyen TN, Bertagnolli D, et al. Shared and distinct transcriptomic cell types across neocortical areas. Nature. 2018;563: 72–78. doi:10.1038/s41586-018-0654-5
6. Zhang Z, Luo D, Zhong X, Choi JH, Ma Y, Wang S, et al. SCINA: Semi-Supervised Analysis of Single Cells in Silico. Genes . 2019;10: 531. doi:10.3390/genes10070531
7. Pliner HA, Shendure J, Trapnell C. Supervised classification enables rapid annotation of cell atlases. Nat Methods. 2019; 1–4. doi:10.1038/s41592-019-0535-3
8. Kiselev VY, Yiu A, Hemberg M. scmap: projection of single-cell RNA-seq data across data sets. Nat Methods. 2018;15: 359. Available: https://doi.org/10.1038/nmeth.4644
9. Cao Z-J, Wei L, Lu S, Yang D-C, Gao G. Searching large-scale scRNA-seq databases via unbiased cell embedding with Cell BLAST. Nat Commun. 2020;11: 3458. doi:10.1038/s41467-020-17281-7
10. Alquicira-Hernandez J, Sathe A, Ji HP, Nguyen Q, Powell JE. ScPred: Accurate supervised method for cell-type classification from single-cell RNA-seq data. Genome Biol. 2019;20: 264. doi:10.1186/s13059-019-1862-5
11. de Kanter JK, Lijnzaad P, Candelli T, Margaritis T, Holstege FCP. CHETAH: a selective, hierarchical cell type identification method for single-cell RNA sequencing. Nucleic Acids Res. 2019;47: e95–e95. doi:10.1093/nar/gkz543
12. Wang S, Pisco AO, McGeever A, Brbic M, Zitnik M, Darmanis S, et al. Unifying single-cell annotations based on the Cell Ontology. bioRxiv. 2019; 810234. doi:10.1101/810234
13. Zeisel A, Muñoz-Manchado AB, Codeluppi S, Lönnerberg P, La Manno G, Juréus A, et al. Brain structure. Cell types in the mouse cortex and hippocampus revealed by single-cell RNA-seq. Science. 2015;347: 1138–1142. doi:10.1126/science.aaa1934
14. Hodge RD, Bakken TE, Miller JA, Smith KA, Barkan ER, Graybuck LT, et al. Conserved cell types with divergent features in human versus mouse cortex. Nature. 2019; 1–8. doi:10.1038/s41586-019-1506-7
15. Jarvis P. Towards a Comprehensive Theory of Human Learning. Taylor & Francis Ltd; 2006.
16. Yang BH, Asada H. Progressive learning and its application to robot impedance learning. IEEE Trans Neural Netw. 1996;7: 941–952. doi:10.1109/72.508937
17. Fayek HM. Continual Deep Learning via Progressive Learning. RMIT University. 2019.
18. Yuste R, Hawrylycz M, Aalling N, Aguilar-Valles A, Arendt D, Arnedillo RA, et al. A community-based transcriptomics classification and nomenclature of neocortical cell types. Nature Neuroscience. Nature Research; 2020. doi:10.1038/s41593-020-0685-8
19. Svensson V, Beltrame E da V. A curated database reveals trends in single cell transcriptomics. bioRxiv. 2019; 742304. doi:10.1101/742304
20. Wagner F, Yanai I. Moana: A robust and scalable cell type classification framework for single-cell RNA-Seq data. bioRxiv. 2018; 456129. doi:10.1101/456129
21. Bakken TE, Hodge RD, Miller JA, Yao Z, Nguyen TN, Aevermann B, et al. Single-nucleus and single-cell transcriptomes compared in matched cortical cell types. Soriano E, editor. PLoS One. 2018;13: e0209648. doi:10.1371/journal.pone.0209648
22. Aevermann BD, Novotny M, Bakken T, Miller JA, Diehl AD, Osumi-Sutherland D, et al. Cell type discovery using single-cell transcriptomics: implications for ontological representation. Hum Mol Genet. 2018;27: R40–R47. doi:10.1093/hmg/ddy100
23. Abdelaal T, Michielsen L, Cats D, Hoogduin D, Mei H, Reinders MJT, et al. A comparison of automatic cell identification methods for single-cell RNA sequencing data. Genome Biol. 2019;20: 194. doi:10.1186/s13059-019-1795-z
24. Boufea K, Seth S, Batada NN. scID Uses Discriminant Analysis to Identify Transcriptionally Equivalent Cell Types across Single-Cell RNA-Seq Data with Batch Effect. iScience. 2020;23: 100914. doi:10.1016/j.isci.2020.100914
25. Tax D. One-class classification Concept-learning in the absence of counter-examples. TU Delft. 2001.
26. Zappia L, Phipson B, Oshlack A. Splatter: simulation of single-cell RNA sequencing data. Genome Biol. 2017;18: 174. doi:10.1186/s13059-017-1305-0
27. Zheng GXY, Terry JM, Belgrader P, Ryvkin P, Bent ZW, Wilson R, et al. Massively parallel digital transcriptional profiling of single cells. Nat Commun. 2017;8: 14049. doi:10.1038/ncomms14049
28. Stuart T, Butler A, Hoffman P, Hafemeister C, Papalexi E, Mauck WM, et al. Comprehensive Integration of Single-Cell Data. Cell. 2019;177: 1888–1902.e21. doi:10.1016/j.cell.2019.05.031

29. León B, López-Bravo M, Ardavín C. Monocyte-derived dendritic cells. Semin Immunol. 2005;17: 313–318. doi:10.1016/j.smim.2005.05.013

30. Schaum N, Karkanias J, Neff NF, May AP, Quake SR, Wyss-Coray T, et al. Single-cell transcriptomics of 20 mouse organs creates a Tabula Muris. Nature. 2018;562: 367–372. doi:10.1038/s41586-018-0590-4

31. Saunders A, Macosko EZ, Wysoker A, Goldman M, Krienen FM, de Rivera H, et al. Molecular Diversity and Specializations among the Cells of the Adult Mouse Brain. Cell. 2018;174: 1015–1030.e16. doi:10.1016/j.cell.2018.07.028

32. Rosenberg AB, Roco CM, Muscat RA, Kuchina A, Sample P, Yao Z, et al. Single-cell profiling of the developing mouse brain and spinal cord with split-pool barcoding. Science. 2018;360: 176–182. doi:10.1126/science.aam8999

33. Korsunsky I, Millard N, Fan J, Slowikowski K, Zhang F, Wei K, et al. Fast, sensitive and accurate integration of single-cell data with Harmony. Nat Methods. 2019;16: 1289–1296. doi:10.1038/s41592-019-0619-0

34. Lotfollahi M, Naghipourfar M, Luecken M, Khajavi M, Büttner M, Avsec Z, et al. Query to reference single-cell integration with transfer learning. bioRxiv. 2020; 2020.07.16.205997. doi:10.1101/2020.07.16.205997

35. Cinti F, Bouchi R, Kim-Muller JY, Ohmura Y, Sandoval PR, Masini M, et al. Evidence of β-Cell Dedifferentiation in Human Type 2 Diabetes. J Clin Endocrinol Metab. 2016;101: 1044–1054. doi:10.1210/jc.2015-2860

36. Hunter CS, Stein RW. Evidence for Loss in Identity, De-Differentiation, and Trans-Differentiation of Islet β-Cells in Type 2 Diabetes. Front Genet. 2017;8: 35. doi:10.3389/fgene.2017.00035

37. Pedregosa F, Varoquaux G, Gramfort A, Michel V, Thirion B, Grisel O, et al. Scikit-learn: Machine Learning in Python. 2011 pp. 2825–2830. Available: http://scikit-learn.sourceforge.net.

38. Fagni T, Sebastiani F. On the Selection of Negative Examples for Hierarchical Text Categorization. Proceedings of the 3rd language technology conference. 2007; 24–28.

39. Kiritchenko S, Famili F. Functional Annotation of Genes Using Hierarchical Text Categorization. Proceedings of BioLink SIG, ISMB. 2005.

40. Wolf FA, Angerer P, Theis FJ. SCANPY: Large-scale single-cell gene expression data analysis. Genome Biol. 2018;19: 15. doi:10.1186/s13059-017-1382-0

41. Ding J, Adiconis X, Simmons SK, Kowalczyk MS, Hession CC, Marjanovic ND, et al. Systematic comparison of single-cell and single-nucleus RNA-sequencing methods. Nat Biotechnol. 2020;38: 737–746. doi:10.1038/s41587-020-0465-8

42. Van Der Wijst MGP, Brugge H, De Vries DH, Deelen P, Swertz MA, Franke L. Single-cell RNA sequencing identifies celltype-specific cis-eQTLs and co-expression QTLs. Nat Genet. 2018;50: 493–497. doi:10.1038/s41588-018-0089-9

43. L.C.M. Michielsen, M.J.T. Reinders, A. Mahfouz. Hierarchical progressive learning of cell identities in single-cell data. 2021. doi:10.5281/zenodo.4644285

Supplementary Materials

Supplementary Note 1

When matching the cell populations from two datasets, we distinguish five options: simple, multiple columns, multiple rows, complex, and impossible. When describing the different scenarios within these options, we sometimes make a distinction between leaf nodes and internal nodes. Here, it is important to remember that only *T1* can have internal nodes since this is the tree that is updated. *T2* is always a flat classification tree, so only consists of the root node and leaf nodes.

Simple. In this scenario, we find a unique match between a cell population, P_i, from dataset 1 and a cell population, P_j, from dataset 2. As as consequence, $X_{j,I}$ will be 1 or 2 and the rest of row *j* and column *i* in *X* are zero. Within this scenario, there are three different options:

1. Both cell populations are leaf or internal nodes. This indicates a perfect match. The tree is not updated, but the labels of P_j are renamed to P_i (Figure S24A). This is the same scenario as the 'perfect match' scenario described in the main text.
2. P_i is a leaf or internal node, but P_j is the root node of *T2*. This indicates that P_i is missing in dataset 2. The node, however, is already in the tree, so it is not updated (Figure S24B).
3. P_i is the root of *T1*, but the P_j is a leaf node. This indicates that P_j is missing in dataset 1. The cell population is thus also not in the tree yet, so we will add it as a child to the root (Figure S24C). This is the same scenario as the 'new population' scenario described in the main text.

Multiple rows. In this scenario, a cell population, P_i, from dataset 1 matches multiple populations from dataset 2. In *X* there will be multiple non-zero values in column *i.* Here, we distinguish two different scenarios:

1. P_i matches only cell populations from dataset 2 that are leaf node. We consider the cell populations from dataset 2 subpopulations of P_i, so we add them as descendants to P_i (Figure S25A). This is the same scenario as the 'splitting nodes' scenario described in the main text.
2. The root node of *T2* is also involved. We simple ignore this node and for the rest do the same as above (Figure S25B-C).

Multiple columns. This scenario is quite similar to the multiple rows scenario. Here, however, multiples populations from dataset 1 match one cell population, P_j, of dataset 2. In *X* there will be multiple non-zero values in row *j.* This scenario is a little more complex since the populations from dataset 1 does not have to be leaf nodes or the root node, but there can also be internal nodes in this tree. Here, we distinguish three different scenarios:

1. The root node of *T1* and *T2* are not involved, so multiple cell populations, which can be leaf or internal nodes, from dataset 1 match P_j. We consider the cell populations from dataset 1 subpopulations of P_j, so we need to add P_j as a parent node to these cell populations (Figure S26A). This is same scenario as the 'merging nodes' scenario described in the main text. It could be, however, that this node already exists in this tree

(Figure S26B). If this is the case, we have a perfect match between a node from tree 1 and tree 2, so we do not have to update the tree, but we only have to update the labels of P_j.

2. Besides leaf or internal nodes, the root of *T1* is involved. This indicates that P_j is 'bigger' than the cell populations from dataset 1 as part of it is unlabeled. Therefore, we add P_j as a descendant to the root of *T1*. Next, we rewire the involved cell populations from dataset 1 such that they become descendants of P_j (Figure S26C).
3. The root node of *T2* is involved. This indicates that multiple cell populations from dataset 1 are missing in dataset 2. These nodes, however, are already in the tree, so the tree can remain the same (Figure S26D).

Complex. The scenarios described above were all relatively easy. A cell population from one dataset matches either one or multiple cell populations from another. It could also happen, however, that multiple cell populations from dataset 1 match multiple cell populations from dataset 2 (Figure S27). As a consequence, there will a certain place $X_{j,I}$ which is either 1 or 2 and there are two or more non-zero values in the corresponding row *j* and column *i*. Here, we distinguish three different scenarios:

1. The root node of *T1* is involved. We just assume that the boundary should be adjusted and this is automatically done, so we remove this `1' from the table (Figure S27A). If the situation is still complex after the one is removed, we continue to scenario 2 or 3. If not, we treat it as a multiple rows problem as explained above.
2. The root node of *T2* is involved. Again, we just assume that the boundary should be adjusted, so we remove this `1' from the table (Figure S27B). If the situation is still complex after the one is removed, we continue to scenario 3. If not, we treat it as a multiple columns problem as explained above.
3. Multiple leaf/internal nodes of dataset 1 are involved and multiple leaf nodes of dataset 2. We can only solve this if the 'complex' cell population, P_i, of dataset 1 is not a leaf node. Otherwise we are dealing with an impossible scenario which is described below. If the complex node is an internal node, we attach the involved cell populations of dataset 2 as descendants to the complex node (splitting scenario) and attach the involved cell populations of dataset 1, except for P_i, to P_j (Figure S27C).

Impossible. Sometimes, it could be impossible to match the labels from two datasets. Something could have gone wrong during the clustering, e.g. a population 1 and 2 from dataset 1 match population A from dataset 2, but population 2 also matches population C from dataset 2 (Figure S28A). Here, population A and C should be merged into population 2, but population A should also be split into population 1 and 2. Population 2, however, cannot be added to the tree twice. It could also be that dataset 2 contains labels at a different resolution, e.g. that population B is a subpopulation of population A (Figure S28B). This is not what we assumed and thus impossible to match. Both scenarios occur when a leaf node from dataset 1 is at a crossing of multiple rows and multiple columns (i.e. a complex situation). An extra difficulty is that there are thus multiple situations that could explain this. All of these situation are not what we desired and thus we call it impossible and do nothing.

Supplementary Note 2
If there is a complex scenario that cannot be solved immediately, matrix *X* will be changed into a strict matrix. In the strict matrix, only reciprocal matches are considered, so all ‘1’s’ are turned into ‘0’. There are some exceptions to this rule.

- A population can never have a reciprocal match with the root, so these ‘1’s’ are never removed.
- If a population from a dataset has only one match, it is also never removed. Consider the following example: If population P1 of Dataset 1 is only predicted to be Population Q of Dataset 2, we know that P1 should be a match with Q as it cannot be matched with any other population or with the root. It could be that this match is not reciprocal if population Q has many different subpopulations (e.g. P1, P2, P3, P4). Imagine that population P2 is really big. Almost all cells of population Q will be predicted to be P2 and so the matches with P1 (and P3 and P4) are missed because of the matching threshold. In case there is a complex scenario caused by any other population (maybe P2 or P3 or P4), we still know that P1 is a subpopulation of Q, since that was super clear and didn’t cause any complexity.

Supplementary Note 3
Current scRNA-seq data simulators cannot simulate hierarchical data, so we simulated this dataset step by step (Figure S1B).
First, we simulated the expression of 3,000 genes for 9,000 cells. For this simulation, the cells were divided into three groups. The 3,000 simulated genes represent genes that are differentially expressed between the cell populations at a low resolution, so for example B cells vs. T cells. Next, we simulated *another* 3,000 genes for the *same* 9,000 cells. Now, the cells were divided into five groups. Here, the differentially expressed genes represent genes that distinguish cell populations at a slightly higher resolution, so for example CD4+ T cells vs. CD8+ T cells. We repeated this step for another set of 3,000 genes, but now there were six populations. The third dataset represents the highest resolution, so for instance CD4+ memory T cells vs. CD4+ naïve T cells.
Together this resulted in a dataset of 9,000 cells and 9,000 genes. The cells were labeled at three resolutions. There was some inconsistency between the labels at the different resolutions (e.g. some cells were labeled as ‘Group12’, ‘Group3’, ‘Group3’). We removed these cells from the dataset, which resulted in a final dataset of 8,839 cells and 9,000 genes.

chapter 4

Single-cell reference mapping to construct and extend cell-type hierarchies

Lieke Michielsen*, Mohammad Lotfollahi*, Daniel Strobl, Lisa Sikkema, Marcel J.T. Reinders, Fabian J. Theis†, Ahmed Mahfouz†

This chapter is published in: *NAR Genomics and Bioinformatics* (2023) 5: 3, doi: 10.1093/nargab/lqad070.
Supplementary material is available online at: https://academic.oup.com/nargab/article/5/3/lqad070/7231336#413129034

*Equal contribution, †Equal contribution

Single-cell genomics is now producing an ever-increasing amount of datasets that, when integrated, could provide large-scale reference atlases of tissue in health and disease. Such large-scale atlases increase the scale and generalizability of analyses and enable combining knowledge generated by individual studies. Specifically, individual studies often differ regarding cell annotation terminology and depth, with different groups specializing in different cell type compartments, often using distinct terminology. Understanding how these distinct sets of annotations are related and complement each other would mark a major step towards a consensus-based cell-type annotation reflecting the latest knowledge in the field. Whereas recent computational techniques, referred to as "reference mapping" methods, facilitate the usage and expansion of existing reference atlases by mapping new datasets (i.e., queries) onto an atlas; a systematic approach towards harmonizing dataset-specific cell-type terminology and annotation depth is still lacking. Here, we present "*treeArches*", a framework to automatically build and extend reference atlases while enriching them with an updatable hierarchy of cell-type annotations across different datasets. We demonstrate various use cases for treeArches, from automatically resolving relations between reference and query cell types to identifying unseen cell types absent in the reference, such as disease-associated cell states. We envision *treeArches* enabling data-driven construction of consensus atlas-level cell-type hierarchies and facilitating efficient usage of reference atlases.

4.1 Introduction

Single-cell sequencing technologies have revolutionized our understanding of human health. Hereto, large single-cell datasets - referred to as "reference atlases" - have been built to characterize the cellular heterogeneity of whole organs. An example is all the organ- and body-scale cell atlases constructed within big consortia such as the human cell atlas (HCA) [1–5]. Users can contextualize their datasets within these references to identify novel cell types. This enables the discovery of disease-affected cell types that can be prioritized for treatment design [6–8].

To create a reference atlas, one would ideally leverage information from multiple scRNA-seq datasets and harmonize their cell annotations. This, however, is not as easy as it seems since all datasets are annotated at a different resolution. Furthermore, matching cell types based on their names is difficult. Databases such as 'Cell Ontology' try to overcome this problem, but a complete naming convention is still missing [9]. When constructing the Human Lung Cell Atlas (HLCA), for instance, the cell type labels of 14 datasets had to be manually harmonized, which is a time-consuming process [2]. To accelerate the construction of reference atlases, we developed scHPL: a method to automatically match the cell-type labels of multiple datasets and construct a cell-type hierarchy [10]. In follow-up, Novella-Rausell et al. showed how scHPL simplified the process when building a mouse kidney atlas [11].

The concept of a "reference atlas", however, suggests it should help analyze and interpret new datasets (here denoted as "query"). This is, however, complicated by batch effects between the reference and query, limited computational resources, and data privacy and sharing. Recently, we, along with others, developed computational approaches (known as "reference mapping" methods) to address these challenges [4,12,13]. Such methods could for instance be used to map a query dataset to the reference and annotate the cells. Currently, there is no method available that tackles both challenges simultaneously.

To address these challenges, we present treeArches, a framework that builds upon single-cell architectural surgery (scArches) [12] and single-cell Hierarchical Progressive Learning (scHPL) [10] to progressively build and update a reference atlas and corresponding hierarchical classifier. Our approach allows users to build a reference atlas using existing integration methods supported by scArches (e.g., scVI, scANVI, totalVI, and all others described in [14]). Next, we use scHPL to augment this reference atlas by learning the relations between cell types to construct a cell-type hierarchy. Afterward, query data, which can be either annotated or unannotated, can be mapped to the reference. If the query is annotated, the query cells can expand the newly updated tree by highlighting potential novel cell types and their relationship with other cell types in the reference. Otherwise, the created reference can be used to annotate the query cells and identify new unseen cell types in the query. Unlike existing methods, we show that treeArches can be used to create a reference atlas and corresponding cell-type hierarchy from scratch, update an existing reference atlas and the hierarchy by finding novel relations between cell types, and leverage a reference atlas to transfer labels to a new dataset.

4.2 Methods

4.2.1 Overview

treeArches consists of two main steps: (i) removing the batch effects between datasets and (ii) matching the annotated cell types to construct a cell-type hierarchy (Figure 1). Starting with multiple labeled datasets, hereafter called reference datasets, we first use neural network-based reference-building models (e.g., sc(AN)VI [14] or scGen [15]), which are top performers in recent data benchmarking efforts [16] and compatible with scArches, to construct a latent space. Next, we use scHPL to construct the cell-type hierarchy (Figure 1A). For each dataset, we train a classifier in the learned latent space and cross-predict the labels of the other dataset(s). Using the confusion matrices, we automatically match the cell types to create a hierarchy. This hierarchy also represents a hierarchical classifier where every node represents a cell type in one or more of the datasets. Afterwards, we can map new query datasets to the learned latent space using architectural surgery, a transfer learning approach to map query datasets to references, implemented by scArches (Figure 1B). Architectural surgery brings the advantage that the count matrices of the reference datasets are not needed anymore for querying the model. Instead, we only use the pre-trained neural network architecture. The query datasets can either be labeled or unlabeled. In the case of a labeled dataset, we match the cell types from the query to the reference and again update the hierarchy we had learned on the reference datasets. In the case of an unlabeled query, we annotate the cells using the learned hierarchy.

When matching the cell types or predicting labels of a query dataset, it is important to identify new cell types that are not present in the reference. This is only possible when biological

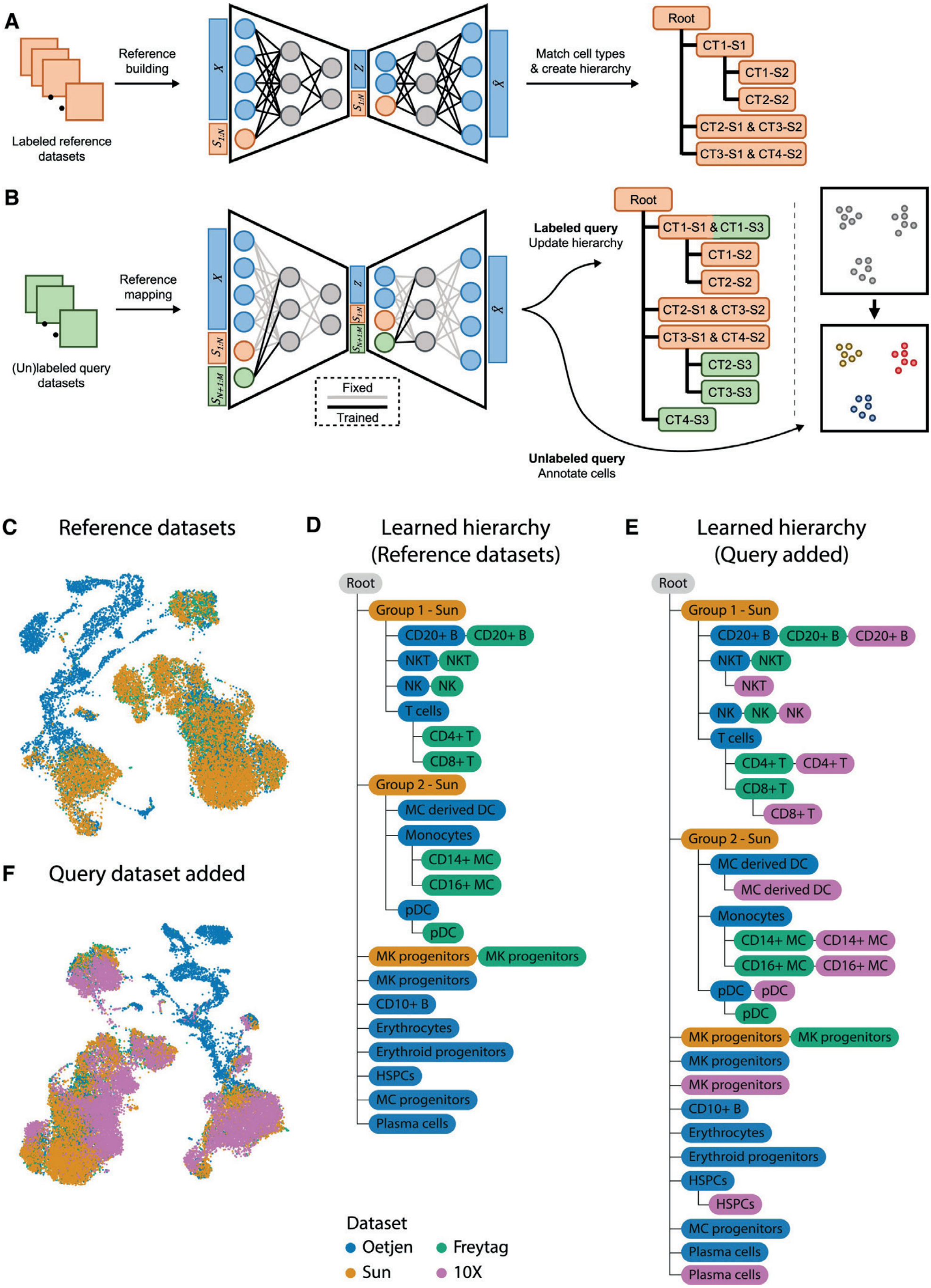


Figure 1. A schematic version of treeArches and an example using PBMC and bone marrow datasets. A) Pre-training of a latent representation using labeled public reference datasets. After integration, a cell-type hierarchy is created by matching the cell types of the different datasets. Here, for instance, cell types (CT) 1 and 2 from study (S) 2 are subtypes of CT1 from S1. **B)** (Un)labeled query datasets can be added to the latent representation by applying

architectural surgery. After integration, the cell-type hierarchy is updated with labeled query datasets. Unlabeled query datasets can be annotated using the learned hierarchy. **C)** UMAP embedding showing the integrated latent space of the three reference datasets. **D)** Cell-type hierarchy learned from the three reference datasets. MC derived DC: monocyte-derived dendritic cells, MC: monocytes, pDC: plasmacytoid dendritic cells, HSPC: hematopoietic stem and progenitor cell. **E)** Updated hierarchy after the 10X dataset was added. **F)** UMAP embedding showing the integrated latent space of the reference and query datasets.

variation is preserved when mapping the datasets to the latent space and when the classifier in scHPL recognizes unseen cells, i.e. cells that are not present in the tree. Therefore scHPL adopts a rejection strategy, which rejects these unseen cells and identifies them as a new cell type. Within scHPL, a cell is rejected if it meets one of the following criteria: 1) if the posterior probability of the classifier is lower than a threshold which means the predicted label is ambiguous, 2) if the distance between a cell and its closest neighbors is too big, and 3) if the reconstruction error (when mapping to a reduced PCA space and back) is above a threshold, which means the query cell is too different from the reference cell types. These three thresholds are automatically set based on the distribution of the data.

treeArches is a framework built around scArches (version 0.5.3) [12] and scHPL (version 1.0.1) [10]. A detailed description of scArches and scHPL can be found in their original papers [10,12]. Here, we only describe changes to the original methods when combined in the treeArches framework. We enhanced the original version of scHPL by adding the option to use a *k*-nearest neighbor (kNN) classifier. The dimensionality of the latent space learned by scArches is relatively low (varying between 10 and 30 dimensions). We noticed that the linear SVM originally implemented doesn't perform well, since the cell types are not linearly separable anymore. Therefore, it is better to use scHPL with the kNN classifier in this case. In contrast to the linear SVM, we train a multiclass classifier for every parent node instead of a binary classifier for every child node [10]. During training, we set the default number of neighbors to 50. However, when there are cell types in the dataset that consist of less than 50 cells, this is not ideal. Therefore, we added an extra option (*dynamic_neighbors*) to automatically decrease *k* to the size of the smallest cell type across the direct child nodes. Since the tree consists of multiple classifiers, it can thus be that they all use a different number of neighbors because of this option. For the kNN classifier itself, we implemented alternatives using either the FAISS library [17] or the scikit-learn library [18]. The FAISS implementation is faster than the scikit-learn library but is only available on Linux.

4.2.2 Detecting new or diseased cell types

We have implemented three methods to detect new or diseased cell types: 1) a threshold on the posterior probability, 2) a threshold on the reconstruction error, and 3) a threshold on the distance between query and reference. The first two options were already implemented in the previous version of scHPL. The default threshold for the first option is 0.5. The threshold for the second rejection option is determined using a nested cross-validation loop. It is the median reconstruction error that gives a certain amount of false negatives on the test folds (default = 0.5%). The third option rejects cells whose distance to the predicted class is too big. The threshold for rejection is determined by calculating the neighbors for all cells in the training set, averaging the distance across the neighbors, and taking the 99th percentile.

4.2.3 Datasets

PBMC datasets. The dataset was obtained from the recent data integration benchmark [16]. The data contains bone marrow samples from Oetjen et al. [19] and also PBMC samples that were obtained from 10x Genomics https://support.10xgenomics.com/single-cell-gene-expression/datasets/3.0.0/pbmc_10k_v3, Freytag et al. and Sun et al. [20,21], the original url and the preprocessing and annotation details can be found in Luecken et al. [16]. Marker genes specific to early erythrocytes and platelets were downloaded from Azimuth [4].

Brain datasets. We used datasets from the primary motor cortex of three species: human, mouse, and marmoset [22]. We downloaded the datasets from the Cytosplore comparison viewer. In these datasets, genes were already matched based on one-to-one homologs. For the analysis, we only kept these one-to-one matches (15,860 genes in total). We selected 2,000 highly variable genes based on the reference datasets (mouse and marmoset) and used those counts as input for treeArches. The datasets are annotated at three different resolutions: Class, Subclass, and RNA_cluster. The class level contains three broad brain cell types: GABAergic neurons, glutamatergic neurons, and non-neuronal cells. At the subclass level, the cells are annotated at a higher resolution (5-10 subclasses per class). The RNA_cluster level contains the highest resolution. Here, we will use the subclass level to match the cell types. Marker genes used for visualization were chosen based on Supplementary Tables 5 and 6 from the original paper [22].

Human Lung Cell Atlas. The human lung cell atlas (HLCA) is a carefully constructed reference atlas for the human respiratory system [2]. Sikkema et al. aligned 14 datasets, harmonized the annotations, and built a cell-type hierarchy consisting of 5 levels. When matching the cell types, we used the latent space generated in their original paper (downloaded from https://zenodo.org/record/6337966#.YqmGIidBx3g). When updating the hierarchy with the IPF data, we removed the cell types smaller than 10 cells. Marker genes were downloaded from the lung reference v2 from Azimuth [2,4]. Marker genes for the Meyer cell populations were obtained from [26]. We annotated the fibrosis-specific cell types in greater detail by sub clustering the cell types of interest (macrophages, epithelial cells, myofibroblasts and identifying the subtypes by marker gene expression. We identified transitioning/basaloid epithelial cells by KRT5/KRT17 expression, inflammatory monocyte-derived macrophages by SPP1 expression, and myofibroblasts by the expression of CTHRC1.

The runtime and memory usage of treeArches on the different datasets can be found in Table S1.

4.2.4 Comparisons

FR-Match. We ran FR-Match (v2.0.0) with default settings on all pairwise combinations of the PBMC reference datasets [23,24]. Before running FR-Match marker genes have to be selected for each cell type. We do this using the method recommended by the authors of FR-Match: NS-Forest [25]. We ran NS-Forest (v3.0) on each dataset separately using the default settings.

MetaNeighbor. We ran MetaNeighbor (v1.13.0) using the default settings on all pairwise combinations of the PBMC datasets [26]. MetaNeighbor returns an AUROC score for all cell-type combinations. As recommended in the MetaNeighbor vignette, we consider two cell types a match when the AUROC is higher than 0.9.

Azimuth. We run Azimuth using Seurat v4.3.0 [4] and follow the 'integration_mapping' vignette.

4.3 Results

4.3.1 treeArches accurately learns PBMC hierarchy

We showcase treeArches with a simulation where we build a cell-type hierarchy using one bone marrow and three PBMC datasets [19–21,27] (Table S2). We consider three datasets as the reference (Freytag, Oetjen, and Sun), and one as the query (10X). The annotations of these datasets have been manually harmonized by Luecken et al. [16], so we relabel some cells to enforce the datasets to be annotated at different resolutions (Table S3, S4). In the Oetjen dataset, for instance, we relabel all the CD4+ and CD8+ T cells as T cells. The challenge here is to correctly match cell types present in multiple datasets and to reconstruct their hierarchy. Some cell types, however, are dataset-specific and these should thus be added as a new node in the tree. Here, it is important to note that these new cell types are not forced to be aligned with other existing cell types during the integration step and that the classifier used by scHPL contains a good rejection option during the matching step. This harmonizing and afterward relabeling of the cells allows us to manually construct a ground truth hierarchy that we can use to evaluate treeArches (Figure S1).

We remove the batch effects from the reference datasets using scVI [14] and match the cell types in the learned latent space (see Methods) (Figure 1C-D, S2). Since both scArches and scHPL are invariant to a different order of the datasets, treeArches will also be invariant [10,12]. For scHPL, however, the datasets still have to be added progressively, which we will do from low to high resolution (Sun - Oetjen - Freytag). The constructed tree by treeArches largely matches the ground truth: seven out of eight Oetjen cell types and all nine Freytag cell types are correctly matched to the Sun cell types (e.g. the CD4+ T cells are a subpopulation of the T cells which are a subpopulation of the Group 1 - Sun cells). The six cell types only found in one dataset are all added as new cell types to the tree (e.g. the CD10+ B cells and erythrocytes).

However, the megakaryocyte (MK) progenitor cells from the Freytag and Sun dataset do not match the cells from Oetjen. The Freytag and Sun datasets are PBMC datasets and the Oetjen dataset is a bone marrow dataset. Looking at the expression of marker genes and the location of the megakaryocyte progenitor cells in the UMAP embedding supports our claim that the cell types from Sun and Freytag should not match Oetjen in the hierarchy (Figure S3). Based on marker gene expression, the MK progenitor cells in the Oetjen dataset should be relabeled as early erythrocytes and the MK progenitor cells in the Freytag and Sun dataset as platelets.

After constructing the reference tree from the three datasets, we align the query dataset to the latent space of the reference datasets using scArches and update the learned hierarchy with the new cell types (Figure 1E-F). For this step, only the trained model and reference latent space are needed. Again, almost all cell types (10 out of 12) are added to the correct node in the tree, while the plasma cells and the MK progenitors are added to the tree as new cell types. These cell types contain 21 and 18 cells, respectively, which makes them difficult to match compared to the other cell types in the query dataset, which contain more than 1000 cells on average.

For some of the cell types, we would expect a perfect match, but the 10X cell type is a subpopulation instead (NKT cells, CD8+ T cells, MC-derived DC, and HSPCs). We tested whether this is indeed a subpopulation and if there are interesting biological differences between the groups. To do so, we used the classifier trained on the 10X dataset and split the cells from these cell types from the reference into two groups: 1) correctly classified, and 2) rejected. Next, we tested whether there are genes differentially expressed between the two groups. Here, we did not look at the HSPCs, since only 6 cells were correctly predicted. For the NKT cells - Freytag, NKT cells - Oetjen, and CD8+ T cells - Freytag, there are (almost) no genes differentially expressed (adjusted p-value < 0.01, log foldchange > 0.5) (Table S5). However, in the monocyte-derived dendritic cells - Oetjen, there are 85 genes upregulated in the rejected cells. According to Enrichr [28–30] 41 of these genes are related to the Cell Cycle R-HSA-1640170 Reactome pathway (adjusted p-value = 3e-40) [31]. The rejected cells are thus probably dividing cells. These results indicate that there could be biological differences between the two groups, but that this is not always the case.

Since there are many dataset-specific cell types in the PBMC datasets, it is important that the rejection option works correctly to ensure that cell types such as erythrocytes from the Oetjen dataset are added to the root node. In treeArches, there are different rejection options: 1) the maximum distance to the training data, 2) the reconstruction error, and 3) the posterior probability. If a cell is rejected based on the first or second option, this indicates that the cell potentially belongs to a new cell type. In the third case, this indicates that the cell's gene expression is similar to two or more cell types and that we thus cannot label it with enough confidence. Using the default settings for these parameters, all dataset-specific cell types are indeed correctly rejected. We tested three options for all thresholds to test the effect related to the different rejection options. This results in minimal differences in the constructed hierarchies (Figure S4). The hierarchies mainly differ in the number of perfect matches. Changing the rejection option causes cell types that were a perfect match to be subpopulations of one another. For example, when using the default settings the CD4+ T cells from the Oetjen and Freytag dataset are a perfect match, but when changing the percentage of false negatives allowed for the reconstruction error to 1%, CD4+ T cells - 10X is a subpopulation of the CD4+ T cells - Freytag. In two cases, however, treeArches cannot resolve where the NKT cells from the 10X dataset should be added to the hierarchy and this cell type is thus missing. In three cases, the megakaryocyte progenitor cells from the Oetjen dataset form a match with the HSPCs from the 10X dataset. When removing all three rejection options, however, the tree looks completely different (Figure S4). Cell types that are dataset-specific are not added to the root node but match another population. For instance, the erythrocytes now are a subpopulation of the Group 1 cells (a combination of T cells, NK cells, NKT cells, and B cells) from the Sun dataset. This shows the importance of the rejection options within treeArches.

Since there is no method with exactly the same functionality as treeArches, we benchmark parts of the algorithm separately. First, we compare our constructed hierarchy for the reference data to the output of two cell-type matching algorithms: FR-Match and MetaNeighbor [23,24,26]. It is important to note that these methods were developed for pairwise comparisons and do not construct a hierarchy. We ran both methods on all combinations of the reference datasets and visualized their matches in a graph (Figure S5). To allow comparisons, we transform the learned hierarchy by treeArches to a graph by adding edges between a parent and all descendants (Figure S5). When comparing the resulting graphs to the ground-truth graph constructed based on the relabeled cell types, treeArches outperforms FR-Match and MetaNeighbor (Table S6). Using treeArches, only two edges are missing and no wrong edges were introduced while using FR-Match and MetaNeighbor there are respectively 11 and 8 wrong edges, and 7 and 11 missing edges.

Next, we compare the cell type classification performance of treeArches to Azimuth [4]. Azimuth allows label transfer by projecting a query dataset onto a reference atlas but assumes that the labels of the reference are already harmonized. Therefore, we compare the performance in two ways: 1) using the datasets annotated at a different resolution, and 2) using the datasets with the manually harmonized labels. We use the Sun, Oetjen, and Freytag datasets as a reference and the 10X dataset as the query. In the first comparison, treeArches outperforms Azimuth (Figure S6), but during the second comparison, Azimuth performs better (Figure S7). During the second comparison, treeArches uses a flat classifier instead of the hierarchical classifier, which might explain why treeArches' performance decreases. Both Azimuth and treeArches rely on a nearest neighbor classifier. Therefore, it's most likely that Azimuth outperforms treeArches because of better data integration. For the data integration, however, Azimuth needs both the reference and query data, while treeArches only uses the trained model and the query data. Purely looking at cell type classification, Azimuth thus outperforms treeArches on this dataset but treeArches offers a broader functionality. Here, we also compare the performance of treeArches using the kNN (default) and a linear SVM which is the best-performing method according to our classification benchmark [32]. Since the latent space is not linearly separable anymore, the kNN outperforms the linear SVM (Figure S7). This motivates the use of a kNN classifier within treeArches.

4.3.2 Increasing the resolution of the human lung cell atlas using treeArches

The human lung cell atlas (HLCA) is a carefully constructed reference atlas for the human respiratory system [2]. Sikkema et al. integrated 14 datasets, re-annotated the cells and constructed a cell-type hierarchy consisting of 5 levels (Figure 2A, S8). Furthermore, they used scArches to project multiple datasets to this reference atlas. Since the cell-type hierarchy for the reference is well-defined, we can omit the reference-building step and leverage treeArches to update the reference hierarchy using one of the labeled query datasets (Meyer) [33]. Using scHPL, we matched the cell types of the Meyer dataset to the cell types from the

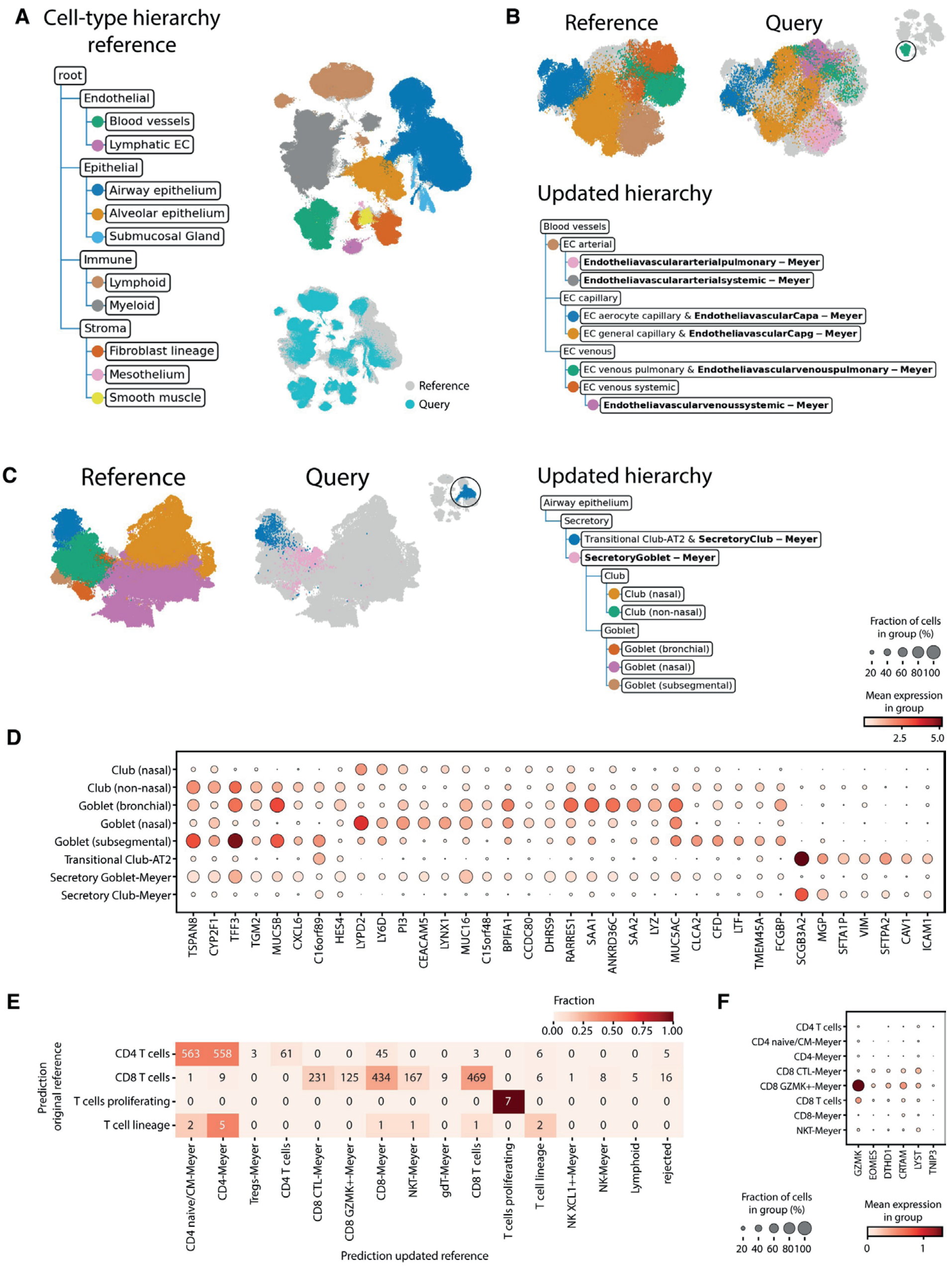


Figure 2. Updated hierarchy when adding Meyer to the reference atlas. A) The cell-type hierarchy corresponding to the reference atlas (only the first two levels are shown). Each node represents a cell type in the reference atlas instead of a cell type in a separate dataset of the reference atlas. The UMAP embedding shows the aligned reference and query dataset. The cells in the reference dataset are colored according to their level 2 annotation. **B, C)** Updated hierarchy zoomed in on the blood vessels and airway epithelium secretory cells respectively. The UMAP embeddings are colored according to their finest resolution. **D)** Expression of marker genes for club and goblet cells in the reference and query cell types. **E)** Comparison of the predictions using the original and updated reference on the T-cells of the Tata dataset. **F)** Expression of marker genes for CD8 + GZMK + cells.

reference (Figure S9). In the updated hierarchy, many cell types from the query dataset match a cell type from the reference as expected based on the cell-type names. Neuroendocrine-Meyer, for instance, is a perfect match to the neuroendocrine cells from the reference. Since no ground truth cell-type matches between the reference datasets and Meyer is known, we cannot assess this quantitatively. For some parts of the hierarchy, we can even increase the resolution. If we zoom in on the blood vessel branch in the tree, for instance, the pulmonary and systemic endothelial vascular arterial cell types from the query both match endothelial cells arterial (EC arterial) from the reference (Figure 2B).

For some parts of the tree, e.g. the airway epithelium secretory cells, the matches are not what we would expect based on the names (Figure 2C). The secretory goblet cells from the query dataset match not only the goblet but also the club cells from the reference and the secretory club cells match the transitional club-alveolar type 2 (AT2) cells. Transitional club-AT2 cells were only recently discovered, which could explain why they are missing from the original Meyer annotations [34–36]. Based on the expression of marker genes, we can conclude that the match between the transitional club-AT2 and secretory club cells is a correct match (Figure 2D). The expression of the marker genes in the other cell types, however, is ambiguous and it is hard to determine what is the correct match. Furthermore, in the HLCA paper, label transfer for these cell types from the reference atlas to the Meyer data did not match well with the original labels either [2].

Furthermore, we see sixteen cell types from the query added to the root node of the tree as a new cell type (Figure S9). Of these cell types, most of them, e.g. chondrocytes, erythrocytes, Schwann cells, and B plasmablasts, are indeed not in the reference atlas. For some, such as some macrophage subtypes that are seen as new, it is more difficult to determine whether they are new or whether they should match one of the macrophage subtypes in the tree. The ‘Macro CHIT1’ cells from the Meyer dataset, for instance, form a relatively big cell type of 1570 cells and are still seen as new. We visualized the expression of *CHIT1*, the gene this cell type was named after, and the marker genes that were used to annotate the cells in the reference data (Figure S10). This shows that the Macro CHIT1 cell type is the only cell type that expresses *CHIT1.* Furthermore, the marker gene profile of the other cell types does not correspond to the profile of the Macro CHIT1 cells, which indicates that this cell type was indeed rejected correctly.

However, twelve out of 77 cell types are missing from the tree, which means that it was impossible to match these Meyer cell types with a cell type from the reference. Due to many-to-many matches between the reference and query cell types, it is sometimes unclear where a cell type should be added to the tree. Especially, when the boundary between cell types is diffuse, it can be quite arbitrary where to put the threshold. If this threshold is different in each dataset or if cells are wrongly annotated in general, this can cause impossible matching scenarios. Here, we notice that this mainly happens with some immune and stromal subtypes. The B cells and plasma cells from the reference and Meyer dataset, for instance, could not be matched automatically, which is caused by the plasma cells in the Meyer dataset that are partially misannotated (Figure S11). Cell types that are missing from the hierarchy thus usually indicate that these cells are wrongly annotated in at least one of the datasets. This information could thus still be used to improve the annotations. Either by using label transfer for these cells using trained hierarchy or manually by visualizing specific marker genes in both datasets.

Next, we annotate a second healthy query dataset (Tata) [35] using the original and updated reference to show that cells in this new query dataset will indeed be mapped to the new Meyer cell types we added to the hierarchy. The majority of the predictions remained unchanged (72.1%, Figure S12). When the predictions differ, cells are often annotated as a Meyer cell type which is a subpopulation of the original annotation (18.4%). A clear example is the T cells: cells previously annotated as CD4+ or CD8+ T cells are now annotated as a subpopulation (Figure 2E). These new annotations are supported by the expression of marker genes (Figure 2F, S13).

4.3.3 treeArches identifies unseen disease-associated cell types in the query data

Next, we show how we can use treeArches to detect previously unseen cell types in idiopathic pulmonary fibrosis (IPF) samples [37]. This dataset was mapped on the HLCA with scArches (Figure 3A-C). Ideally, we would use scHPL to update the hierarchy with the cell types from this query dataset. A downside of the original annotations, however, is that the resolution is very low. Cells are, for instance, only annotated as endothelial cells. Therefore, we used scHPL to predict the labels of the IPF data and compare those predictions to the original annotations (Figure 3D). In the predictions, we see some interesting differences between the IPF and healthy cells.

For the IPF cells, many macrophages and epithelial cells are rejected, while almost none for the healthy cells. Furthermore, most healthy Col1+ cells are predicted to be alveolar fibroblasts, while the diseased Col1+ are mainly SM-activated stress response cells. In all datasets, however, we notice confusion between the B cells and dendritic cells. Based on marker gene expression, the cells originally annotated as B cells and dendritic cells are more likely to be plasma cells and B cells respectively (Figure S14). The cells originally annotated as dendritic cells also overlap in the UMAP with the lymphoid lineage mainly instead of the myeloid lineage (Figure 3A-B).

Next, we annotated the cells at a higher resolution (see Methods) and used these annotations to update the hierarchy (Figure S15). In the updated hierarchy, the healthy and IPF transitioning epithelial cells are not present in the reference atlas and are now correctly added as a new cell type. As expected, we also see some differences in how the healthy and IPF cell types were added to the tree. IPF alveolar macrophage proliferating cells, for instance, are seen as new, while the healthy cells match with the same cell type in the hierarchy. For other IPF macrophage cell types, however, this is not the case even though many cells were rejected previously. Comparing the new annotations with the previously obtained predictions and the matches in the hierarchy, we notice that there are still many macrophages rejected (Figure 3E). For most IPF cell types, however, only a subset of the cells is rejected. For instance, for the IPF monocyte-derived macrophages (Md-M), 486 cells are rejected and 750 are predicted to be Md-M. Therefore, the two cell types are still matched. Comparing the two

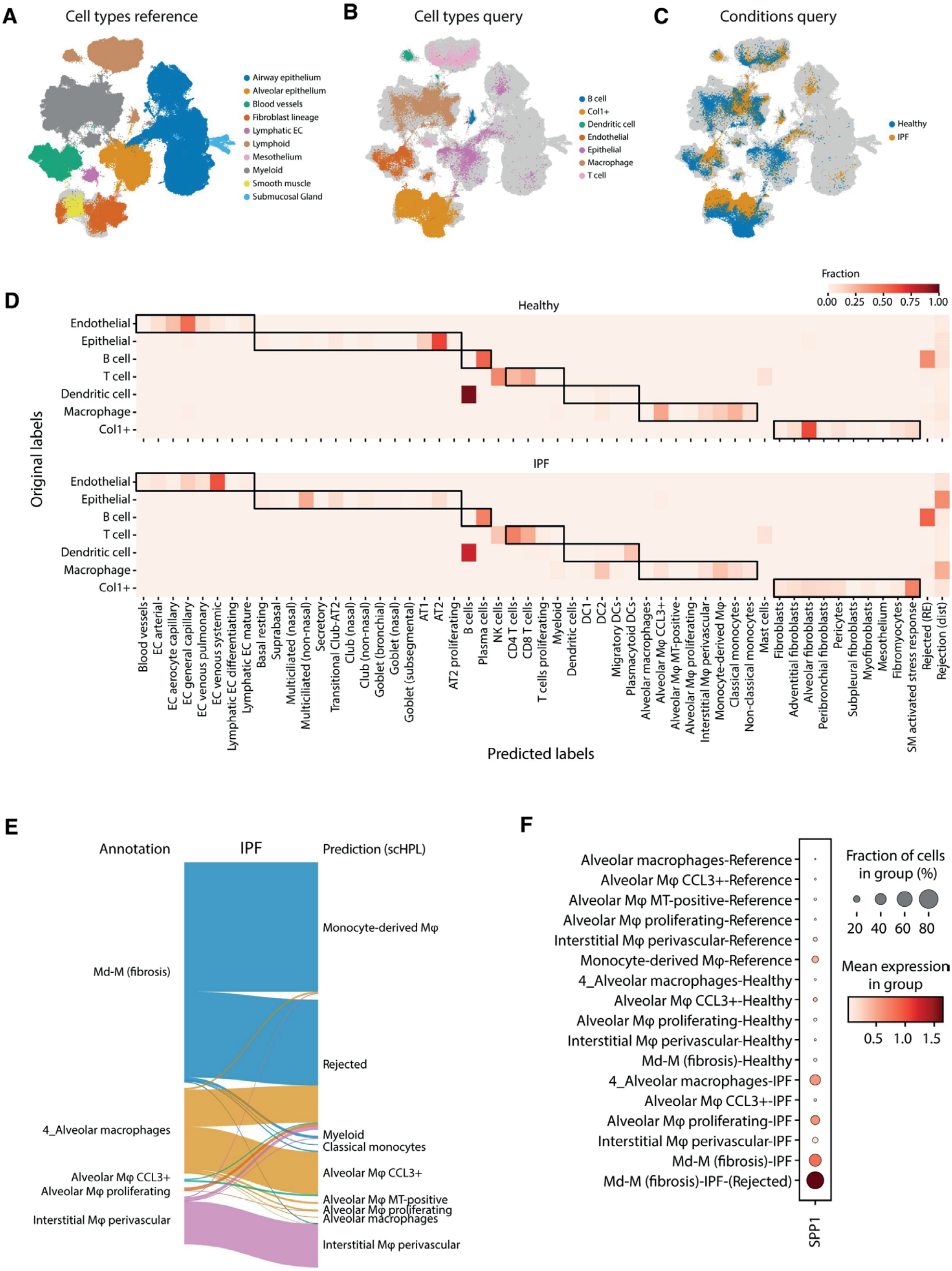


Figure 3. Identifying diseased cells in IPF data. A–C) UMAPs show the HLCA and IPF datasets after alignment. The cells are colored according to their cell type or condition. **D)** Heatmap showing the predicted labels by scHPL and original labels. The dark boundaries indicate the hierarchy of the reference tree. **E)** Sankey diagram showing the new annotations and predictions for the macrophages for the IPF condition. **F)** Expression of *SPP1* in the different cell types of the reference and query datasets.

IPF ‘subtypes’ of Md-M, the top differentially expressed gene is *SPP1* (adjusted p-value = 9.9e-20). Monocytes and macrophages expressing *SPP1* are known to be a hallmark of IPF pathogenesis [38,39]. The rejected Md-M cells are the only group of cells expressing *SPP1* (Figure 3F). For the alveolar and interstitial macrophages, there are 214/493 and 19/276 cells rejected respectively. In these rejected populations, *SPP1* is also upregulated, but only in the alveolar macrophages, it is also differentially expressed (adjusted p-value = 0.0011) (Figure S16). This could indicate that these rejected cells are also a diseased subpopulation. By combining the confusion matrices with the created hierarchy, these diseased subtypes are easily found, either directly as the proliferating cells, or by looking at the rejected cells of a matched cluster.

4.3.4 treeArches can correctly map cell types across species

Next, we show how treeArches can be applied to map the relationship between cell types of different species. We construct a cell-type hierarchy for the motor cortex of the brain using human, mouse, and marmoset data (Table S7) [22]. We integrate the reference datasets, mouse and marmoset, using scVI and construct the cell-type hierarchy using scHPL (Figure 4A-B, S17). Here, we focus on the GABAergic neurons to make the results less cluttered. Almost all cell types (5 out of 7) are a perfect match, except for ‘Meis2’ and ‘Sncg’. In the latent space, the Meis2 cell types from mouse and marmoset also show no overlap, and both cell types were defined using different marker genes (Figure S18A-B). Furthermore, Bakken et al. didn’t find a match between these two either [22]. This could indicate that the Meis2 cells are species-specific and should indeed not match one another. It is unclear why the Sncg cell types (559 and 960 cells in mouse and marmoset respectively) do not match. Even though the cell types are aligned in the UMAP embedding as expected and the marker genes correspond quite well, the cells are rejected based on distance (Figure S18C-D). This means that the cells are still too separated in the latent space. Next, we align the human dataset to the reference using architectural surgery and add the human cell type to the reference hierarchy (Figure 4B-C). Here, the constructed hierarchy looks like what we would expect based on the names of the cell types.

All previous results were obtained using the default parameters (number of neighbors = 50, dynamic number of neighbors = True, see Methods), which turned out to be relatively robust (Figure S19). The main difference is whether a match is found between the Sncg cell types. When increasing the number of neighbors, this match is correctly found.

4.4 Discussion

In this study, we present treeArches, a method to create and extend a reference atlas and the corresponding cell type hierarchy. treeArches builds on scArches, which allows users to easily map new query datasets to the latent space learned from the reference datasets using architectural surgery. Architectural surgery has the advantage that the reference datasets are not needed anymore for the mapping and that the latent space corresponding to the reference datasets does not change. This last point is especially important for scHPL, which

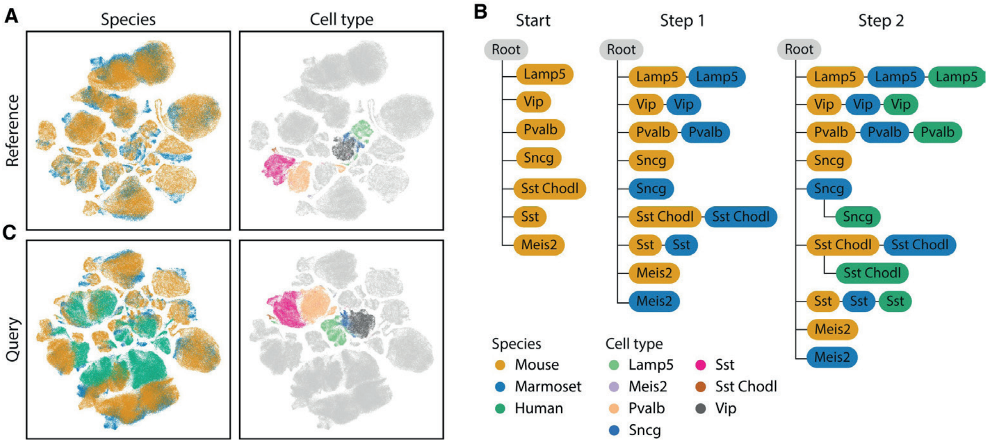


Figure 4. Results motor cortex across species. A) UMAP embedding of the integrated reference datasets. **B)** Learned hierarchy when combining mouse and marmoset (step 1) and after adding human (step 2). The color of each node represents the dataset(s) from which the cell type originates. **C)** UMAP embedding after architectural surgery with the human dataset.

then allows users to match the cell types of multiple labeled datasets to build a cell-type hierarchy. If the latent space of all datasets would be altered when a new dataset is added, we would have to restart the construction of the tree completely.

We have shown three different situations where treeArches can be applied: building a reference atlas from scratch, extending an existing reference atlas to add new cell types or increase the resolution, or using an existing reference atlas to label cells in a new dataset. By using the HLCA data, we show an example of how treeArches can be used to extend a hierarchy or to label cells in a new dataset. The HLCA reference atlas consists of 16 datasets with a well-defined cell-type hierarchy. We show that treeArches can be used to extend this hierarchy. For instance, by increasing the resolution of some branches of the tree, but also by adding new cell types. We could also detect diseased cell types in the IPF datasets.

Whether building or extending a reference atlas or labeling new cells, it is essential that we can detect new cell types, such as disease-specific cell types. To do so, it is important that during the mapping, the cell types are not forced to align; the biological variation should be preserved. Furthermore, during the classification, there should be a correctly working rejection option (i.e. cells are recognized to belong to a new unseen class). Here, we showed that this indeed works in all tested scenarios. A disadvantage of our current approach is that new cell types are usually added to the root node directly instead of to an intermediate node in the hierarchy. However, this is still informative for potential users. It indicates that a certain cell type is different from the known cell types in the tree, and by using prior knowledge or visualizing potential marker genes such cell types could manually be placed at a different, more specific place in the hierarchy.

Due to the extended rejection options, however, it is difficult to match small cell types (less than 50 cells). We modified the kNN classifier from scHPL such that the number of neighbors automatically decreases when there is a small cell type in the training data, but apparently, this is not sufficient in all cases. The number of neighbors is a trade-off between the ability to learn a representation for small cell types and the generalizability of the big cell types.

treeArches relies on the original annotations to extend the cell-type hierarchy. This can be a problem in two different situations. If the annotations are missing or at a too low resolution, it is impossible to extend the atlas. This was the case with the original annotations of the IPF dataset. Alternatively, annotations can have a high resolution, but (partially) incorrect. Especially when there is no clear boundary between cell types, experts might disagree on where to put the boundary (the threshold for the classifier). Inconsistencies like this might result in a hierarchy that looks erroneous at first sight. In those cases, however, treeArches can still be more useful than expected. A cell-type hierarchy that looks different than expected, is usually a sign that the original annotations are inconsistent (e.g. different thresholds are used in different datasets). Certain parts of the dataset, e.g. the cell types that could not be added to the tree or caused confusion, can then be reannotated. Furthermore, the tree can still be adapted afterwards. Examples of this are the goblet and club cells in the HLCA and the megakaryocyte progenitor cells in the PBMC datasets. The learned hierarchy is a good starting point. Based on marker gene expression or expert knowledge, cell types can also be added to the tree, removed from the tree, or rewired. After manually adapting the tree, the classifiers have to be retrained though.

Our proposed method builds upon existing data integration methods. Thus, it naturally inherits both advantages and disadvantages linked to these existing models. As previously reported [12], the choice of the reference building algorithm and reference atlas itself can influence the quality of reference mapping. Therefore, in scenarios where the query dataset is strikingly different from the reference, the integrated query will still contain batch effects leading to inaccurate estimation of hierarchies in treeArches. This erroneous modeling results in weak label transfer results and thus identifies many overlapping cell types between query and reference as a new cell type only present in the query. We advise users to choose a comprehensive reference atlas and extensively benchmark and screen various data integration methods for an optimal reference representation [16].

In summary, we present treeArches, a method that can be used to combine multiple labeled datasets to create or extend a reference atlas and the corresponding cell-type hierarchy. This way we provide users with an easy-to-use pipeline to map new datasets to a current reference atlas, match cell types across multiple labeled datasets, and consistently label cells in new datasets. With the increasing availability of reference atlases, we envision treeArches facilitating the usage of reference atlases allowing users to automatically analyze their datasets from label transfer to the automatic identification of novel cell states in the query data. In conclusion, treeArches will enable a data-driven path towards consensus-based cell type annotation of (human) tissues and will significantly speed up the building and annotation of atlases.

4.5 Code and data availability

treeArches is part of the scArches repository (https://github.com/theislab/scarches). The code for scHPL as a standalone package can be found here: https://github.com/lcmmichielsen/scHPL. All code to reproduce the results and figures can be found at the reproducibility GitHub: https://github.com/lcmmichielsen/treeArches-reproducibility. PBMC count data: https://drive.google.com/uc?id=1Vh6RpYkusbGIZQC8GMFe3OKVDk5PWEpC. Brain count data: https://doi.org/10.5281/zenodo.6786357. PBMC + brain latent space: https://doi.org/10.5281/zenodo.6786357. HLCA latent space: https://zenodo.org/record/6337966#.YqmGIidBx3g

Bibliography

1. Suo C, Dann E, Goh I, Jardine L, Kleshchevnikov V, Park J-E, et al. Mapping the developing human immune system across organs. Science. 2022; eabo0510. doi:10.1126/science.abo0510
2. Sikkema L, Strobl D, Zappia L, Madissoon E, Markov NS, Zaragosi L, et al. An integrated cell atlas of the human lung in health and disease. bioRxiv. 2022. p. 2022.03.10.483747. doi:10.1101/2022.03.10.483747
3. Tabula Sapiens Consortium*, Jones RC, Karkanias J, Krasnow MA, Pisco AO, Quake SR, et al. The Tabula Sapiens: A multiple-organ, single-cell transcriptomic atlas of humans. Science. 2022;376: eabl4896. doi:10.1126/science.abl4896
4. Hao Y, Hao S, Andersen-Nissen E, Mauck WM, Zheng S, Butler A, et al. Integrated analysis of multimodal single-cell data. Cell. 2021;0. doi:10.1016/j.cell.2021.04.048
5. Swamy VS, Fufa TD, Hufnagel RB, McGaughey DM. Building the mega single-cell transcriptome ocular meta-atlas. Gigascience. 2021;10. doi:10.1093/gigascience/giab061
6. Osorio D, McGrail DJ, Sahni N, Stephen Yi S. Drug combination prioritization for cancer treatment using single-cell RNA-seq based transfer learning. bioRxiv. 2022. p. 2022.04.06.487357. doi:10.1101/2022.04.06.487357
7. Bharat A, Querrey M, Markov NS, Kim S, Kurihara C, Garza-Castillon R, et al. Lung transplantation for patients with severe COVID-19. Sci Transl Med. 2020;12. doi:10.1126/scitranslmed.abe4282
8. Wang M, Zadeh S, Pizzolla A, Thia K, Gyorki DE, McArthur GA, et al. Characterization of the treatment-naive immune microenvironment in melanoma with BRAF mutation. J Immunother Cancer. 2022;10. doi:10.1136/jitc-2021-004095
9. Diehl AD, Meehan TF, Bradford YM, Brush MH, Dahdul WM, Dougall DS, et al. The Cell Ontology 2016: enhanced content, modularization, and ontology interoperability. J Biomed Semantics. 2016;7: 44. doi:10.1186/s13326-016-0088-7
10. Michielsen L, Reinders MJT, Mahfouz A. Hierarchical progressive learning of cell identities in single-cell data. Nat Commun. 2021;12: 1–12. doi:10.1038/s41467-021-23196-8
11. Novella-Rausell C, Grudniewska M, Peters DJM, Mahfouz A. A comprehensive mouse kidney atlas enables rare cell population characterization and robust marker discovery. bioRxiv. 2022. p. 2022.07.02.498501. doi:10.1101/2022.07.02.498501
12. Lotfollahi M, Naghipourfar M, Luecken MD, Khajavi M, Büttner M, Wagenstetter M, et al. Mapping single-cell data to reference atlases by transfer learning. Nat Biotechnol. 2022;40: 121–130. doi:10.1038/s41587-021-01001-7
13. Kang JB, Nathan A, Weinand K, Zhang F, Millard N, Rumker L, et al. Efficient and precise single-cell reference atlas mapping with Symphony. Nat Commun. 2021;12: 5890. doi:10.1038/s41467-021-25957-x
14. Gayoso A, Lopez R, Xing G, Boyeau P, Valiollah Pour Amiri V, Hong J, et al. A Python library for probabilistic analysis of single-cell omics data. Nat Biotechnol. 2022;40: 163–166. doi:10.1038/s41587-021-01206-w
15. Lotfollahi M, Wolf FA, Theis FJ. scGen predicts single-cell perturbation responses. Nat Methods. 8/2019;16: 715–721. doi:10.1038/s41592-019-0494-8
16. Luecken MD, Büttner M, Chaichoompu K, Danese A, Interlandi M, Mueller MF, et al. Benchmarking atlas-level data integration in single-cell genomics. Nat Methods. 2022;19: 41–50. doi:10.1038/s41592-021-01336-8
17. Johnson J, Douze M, Jégou H. Billion-Scale Similarity Search with GPUs. IEEE Transactions on Big Data. 2021;7: 535–547. doi:10.1109/TBDATA.2019.2921572
18. Pedregosa F, Varoquaux G, Gramfort A, Michel V, Thirion B, Grisel O, et al. Scikit-learn: Machine Learning in Python. 2011 pp. 2825–2830. Available: http://scikit-learn.sourceforge.net.
19. Oetjen KA, Lindblad KE, Goswami M, Gui G, Dagur PK, Lai C, et al. Human bone marrow assessment by single-cell RNA sequencing, mass cytometry, and flow cytometry. JCI Insight. 2018;3. doi:10.1172/jci.insight.124928
20. Freytag S, Tian L, Lönnstedt I, Ng M, Bahlo M. Comparison of clustering tools in R for medium-sized 10x Genomics single-cell RNA-sequencing data. F1000Res. 2018;7: 1297. doi:10.12688/f1000research.15809.2
21. Sun Z, Chen L, Xin H, Jiang Y, Huang Q, Cillo AR, et al. A Bayesian mixture model for clustering droplet-based single-cell transcriptomic data from population studies. Nat Commun. 2019;10: 1649. doi:10.1038/s41467-019-09639-3
22. Bakken TE, Jorstad NL, Hu Q, Lake BB, Tian W, Kalmbach BE, et al. Comparative cellular analysis of motor cortex in human, marmoset and mouse. Nature. 2021;598: 111–119. doi:10.1038/s41586-021-03465-8
23. Zhang Y, Aevermann B, Gala R, Scheuermann RH. Cell type matching in single-cell RNA-sequencing data using FR-Match. Sci Rep. 2022;12: 9996. doi:10.1038/s41598-022-14192-z
24. Zhang Y, Aevermann BD, Bakken TE, Miller JA, Hodge RD, Lein ES, et al. FR-Match: robust matching of cell type clusters from single cell RNA sequencing data using the Friedman-Rafsky non-parametric test. Brief Bioinform. 2021;22. doi:10.1093/bib/bbaa339
25. Aevermann B, Zhang Y, Novotny M, Keshk M, Bakken T, Miller J, et al. A machine learning method for the discovery of minimum marker gene combinations for cell type identification from single-cell RNA sequencing. Genome Res. 2021;31: 1767–1780. doi:10.1101/gr.275569.121
26. Crow M, Paul A, Ballouz S, Huang ZJ, Gillis J. Characterizing the replicability of cell types defined by single cell RNA-sequencing data using MetaNeighbor. Nat Commun. 2018;9: 884. doi:10.1038/s41467-018-03282-0

27. Genomics 10x. 10x Datasets Single Cell Gene Expression. 2018. Available: https://support.10xgenomics.com/single-cell-gene-expression/datasets/3.0.0/pbmc_10k_v3

28. Chen EY, Tan CM, Kou Y, Duan Q, Wang Z, Meirelles GV, et al. Enrichr: interactive and collaborative HTML5 gene list enrichment analysis tool. BMC Bioinformatics. 2013;14: 128. doi:10.1186/1471-2105-14-128

29. Kuleshov MV, Jones MR, Rouillard AD, Fernandez NF, Duan Q, Wang Z, et al. Enrichr: a comprehensive gene set enrichment analysis web server 2016 update. Nucleic Acids Res. 2016;44: W90–7. doi:10.1093/nar/gkw377

30. Xie Z, Bailey A, Kuleshov MV, Clarke DJB, Evangelista JE, Jenkins SL, et al. Gene Set Knowledge Discovery with Enrichr. Curr Protoc. 2021;1: e90. doi:10.1002/cpz1.90

31. Gillespie M, Jassal B, Stephan R, Milacic M, Rothfels K, Senff-Ribeiro A, et al. The reactome pathway knowledgebase 2022. Nucleic Acids Res. 2022;50: D687–D692. doi:10.1093/nar/gkab1028

32. Abdelaal T, Michielsen L, Cats D, Hoogduin D, Mei H, Reinders MJT, et al. A comparison of automatic cell identification methods for single-cell RNA sequencing data. Genome Biol. 2019;20: 194. doi:10.1186/s13059-019-1795-z

33. Madissoon E, Oliver AJ, Kleshchevnikov V, Wilbrey-Clark A, Polanski K, Orsi AR, et al. A spatial multi-omics atlas of the human lung reveals a novel immune cell survival niche. bioRxiv. 2021. p. 2021.11.26.470108. doi:10.1101/2021.11.26.470108

34. Basil MC, Cardenas-Diaz FL, Kathiriya JJ, Morley MP, Carl J, Brumwell AN, et al. Human distal airways contain a multipotent secretory cell that can regenerate alveoli. Nature. 2022;604: 120–126. doi:10.1038/s41586-022-04552-0

35. Kadur Lakshminarasimha Murthy P, Sontake V, Tata A, Kobayashi Y, Macadlo L, Okuda K, et al. Human distal lung maps and lineage hierarchies reveal a bipotent progenitor. Nature. 2022;604: 111–119. doi:10.1038/s41586-022-04541-3

36. Rustam S, Hu Y, Mahjour SB, Rendeiro AF, Ravichandran H, Urso A, et al. A Unique Cellular Organization of Human Distal Airways and Its Disarray in Chronic Obstructive Pulmonary Disease. Am J Respir Crit Care Med. 2023. doi:10.1164/rccm.202207-1384OC

37. Tsukui T, Sun K-H, Wetter JB, Wilson-Kanamori JR, Hazelwood LA, Henderson NC, et al. Collagen-producing lung cell atlas identifies multiple subsets with distinct localization and relevance to fibrosis. Nat Commun. 2020;11: 1920. doi:10.1038/s41467-020-15647-5

38. Morse C, Tabib T, Sembrat J, Buschur KL, Bittar HT, Valenzi E, et al. Proliferating SPP1/MERTK-expressing macrophages in idiopathic pulmonary fibrosis. Eur Respir J. 2019;54. doi:10.1183/13993003.02441-2018

39. Karman J, Wang J, Bodea C, Cao S, Levesque MC. Lung gene expression and single cell analyses reveal two subsets of idiopathic pulmonary fibrosis (IPF) patients associated with different pathogenic mechanisms. PLoS One. 2021;16: e0248889. doi:10.1371/journal.pone.0248889

chapter 5

Cell type matching across species using protein embeddings and transfer learning

Kirti Biharie, Lieke Michielsen, Marcel J.T. Reinders, Ahmed Mahfouz

---

This chapter is published in: *Bioinformatics* (2023) 39: Issue Supplement_1, doi: 10.1093/bioinformatics/btad248
Supplementary material is available online at:
https://academic.oup.com/bioinformatics/article/39/Supplement_1/i404/7210442#409575728

Knowing the relation between cell types is crucial for translating experimental results from mice to humans. Establishing cell type matches, however, is hindered by the biological differences between the species. A substantial amount of evolutionary information between genes that could be used to align the species is discarded by most of the current methods since they only use one-to-one orthologous genes. Some methods try to retain the information by explicitly including the relation between genes, however, not without caveats. In this work, we present a model to Transfer and Align Cell Types in Cross-Species analysis (TACTiCS). First, TACTiCS uses a natural language processing model to match genes using their protein sequences. Next, TACTiCS employs a neural network to classify cell types within a species. Afterwards, TACTiCS uses transfer learning to propagate cell type labels between species. We applied TACTiCS on scRNA-seq data of the primary motor cortex of human, mouse and marmoset. Our model can accurately match and align cell types on these datasets. Moreover, our model outperforms Seurat and the state-of-the-art method SAMap. Finally, we show that our gene matching method results in better cell type matches than BLAST in our model. TACTiCS is available at https://github.com/kbiharie/TACTiCS.

5.1 Introduction

Model organisms, such as mouse and marmoset, are often used in brain research as a substitute for humans. However, because of differences between species, experiments performed on model organisms do not directly translate to humans. For example, widely-used antidepressants that target serotonin receptors are often tested on mice, while the expression pattern of serotonin receptors is highly divergent between human and mouse, likely leading to differences in cell function between species [1]. Consequently, to facilitate translational research, it is important to better characterize cell type matches between species. This facilitates studying how drugs then alter biological processes within specific cell types between these species.

Traditionally, cell types were characterized solely based on morphology, but using single-cell RNA sequencing (scRNA-seq), the expression pattern across thousands of genes can now be used to describe a cell type. This has resulted in the identification of an increasing number of cell types within specific brain regions [2,3]. Although this improves our understanding of biological processes in the brain, when comparing species, it introduces the need for a method that can match these new cell types accurately between species.

Unfortunately, this is not a trivial task as genes are modified, duplicated and deleted throughout evolution, resulting in complicated many-to-many gene-gene relationships between species. These relationships become even more complicated when evolutionary distances increase.

Current methods that match cell types across species based on scRNA-seq data can be divided into two groups, mainly based on how they solve the gene-matching problem. The first group only uses the one-to-one orthologous genes, which are genes with exactly one match in the other species based on sequence similarity (e.g. using BLAST [4]). Methods such as scANVI [5], MetaNeighbour [6], and LAMbDA [7] belong to this group. While this is a straightforward approach, it ignores genes with a more complex evolutionary history which might have caused divergent functional specification of cell types between species. The second group of methods, including SAMap [8], CAME [9], Kmermaid [10], and C3 [11], overcomes this limitation by considering many-to-many relationships between the genes based on sequence similarity. All these methods rely on the classical assumption that sequence similarity is a good measure of how genes functionally relate to each other. However, sequence similarity often considers one nucleotide/amino acid at a time, which to a large extent ignores sequence contexts important for functional characterization (e.g. secondary structures and sequence motifs). A growing body of evidence suggests that language models are a powerful approach to capture functional similarities between genes [12–15]. Similarly, we hypothesize that using language models to match genes between species can be beneficial for cell type matching.

Once we identified matching relationships between genes across species, the next step is to characterize cell type matches. We and others have posed cell type matching as a classification task where the agreement of predictions from two classifiers, trained on two labeled scRNA-seq datasets, is used to match cell types between the datasets [7,16,17]. Biological differences between species, however, hinder applying such a method directly. A solution could be to learn a common embedding space for the cells before training the classifiers.

Here we introduce a method to Transfer and Align Cell Types in Cross-Species analysis (TACTiCS) that incorporates the two claims that we make: 1) using language models to match genes functionally between species, and 2) training classifiers in a shared embedding space to transfer cell types from one species to the other. We show that TACTiCS correctly matches human, mouse and marmoset brain cell populations from the primary motor (M1) cortex at a detailed cell type level, and does so better than SAMap, the current state-of-the-art method.

5.2 Methods

TACTiCS takes as input two single-cell (sc) or single-nucleus (sn) RNA-seq datasets, with raw expression counts, from two species A and B. TACTiCS consists of four steps (Figure 1): 1) matching genes based on the protein sequences, 2) creating a shared feature space by mapping expression values with the gene matches obtained in step 1, 3) training within-species cell type classifiers, and 4) matching cell types by swapping the classifiers.

5.2.1 Matching genes

First, we created an embedding for every gene using ProtBERT, a transformer-based language model [15]. The protein sequences were retrieved from UniProt [18]. For human and mouse, we selected only the Swiss-prot sequences, but for marmoset we selected all protein sequences. We input the protein sequences to ProtBERT to create an embedding for each protein (Figure 1A). ProtBERT generates a 1024-dimensional embedding for every amino acid in the protein sequence. To allow TACTiCS to work with variable-length proteins, we followed common practice [14] and took the mean embedding over all positions to represent the whole protein sequence (as well as the corresponding gene). Protein sequences longer than

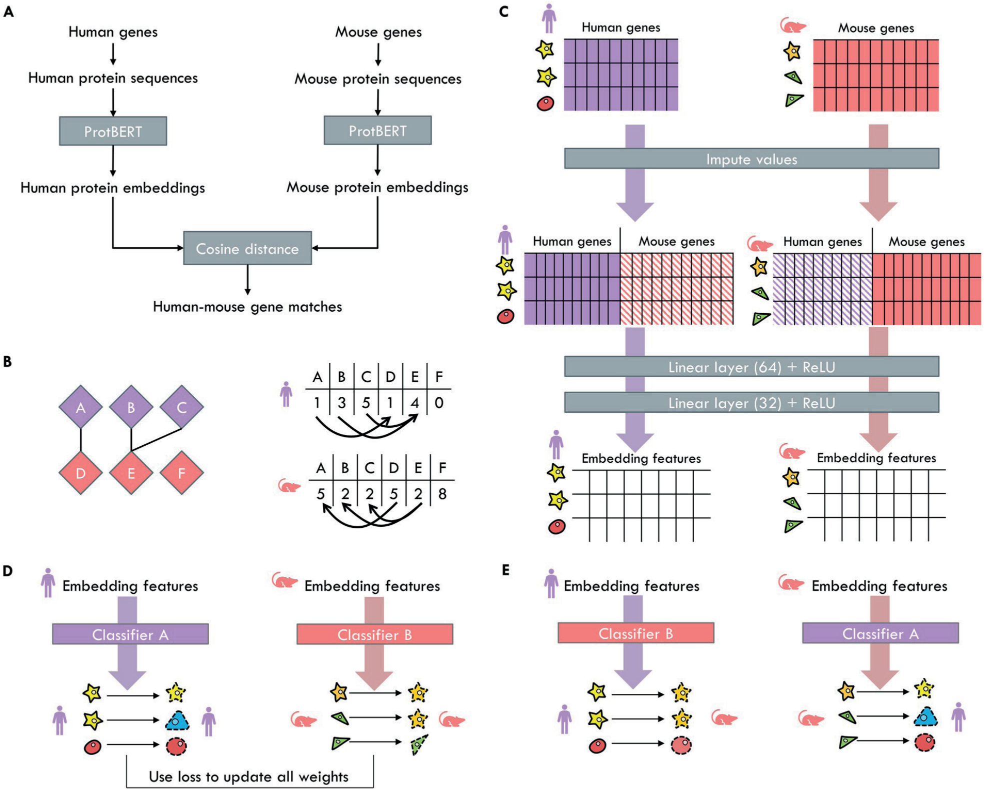


Figure 1. Schematic overview of TACTiCS. We use human and mouse as example, but cell types from any two species can be matched. **A)** Matching genes on protein sequences using ProtBERT. **B)** Bipartite graph of gene matches. Gene expression is imputed by taking the weighted average from connected genes in the bipartite graph. **C)** Creating cell embeddings using linear layers on the shared feature space. The weights of the linear layers are shared. **D)** Classifying within-species cells during training. The classifier consists of a linear layer outputting the cell type probabilities followed by a softmax. **E)** Classifying cross-species cells using transfer learning. The predictions are used to match cell types.

2500 amino-acids (<2% of all sequences) were truncated to the first 2500 to fit into the memory of the GPU.

Next, for every pair of genes from species A and species B, we calculated the cosine distance between the ProtBERT embeddings. The initial set of gene matches were pairs with a cosine distance ≤ 0.005. To ensure that a gene is not connected to too many genes, we kept only the five closest genes, that met the distance threshold, for every gene.

Finally, we filtered the informative gene matches. Hereto, we calculated the top 2000 highly variable genes per species using Scanpy `highly_variable_genes`, and kept only those gene matches where at least one of the two genes is within the set of highly variable genes in their respective species [19]. From these matches, we constructed two sets of genes G_A and G_B, corresponding to species A and B respectively, consisting of genes with a match in the other species.

To obtain sequence similarity-based gene matches, we used BLAST instead of ProtBERT. To obtain the many-to-many BLAST matches we selected matches with an E-value < 1e-6 as the initial set of matches. We used the bitscore as the distance metric. Since BLAST is not symmetrical, one gene match is assigned a separate E-value and bitscore for each direction. If only one direction meets the E-value threshold, we use the corresponding bitscore as the gene distance. If both directions meet the threshold, we use the average of the two bitscores. The list of matches is then filtered similarly as before with the closest-five and highly varying gene filtering. Additionally, we obtained one-to-one BLAST matches by starting with the same set of matches using the E-value threshold. For every gene we kept only the best match, i.e. the gene with the highest bitscore. We discarded gene matches that were not reciprocal and finally also applied the highly varying gene filtering to obtain the one-to-one matches.

5.2.2 Creating a shared feature space by mapping expression values with the gene matches

We normalized the expression levels of genes as follows: 1) the raw expression counts of each dataset are normalized by the number of reads per cell such that the total number of counts in every cell is 10,000, and 2) the natural logarithm of the normalized counts are taken:

$$x_{ij} = \ln\left(\frac{x_{ij}}{\sum_{k \in G} x_{ik}} * 1e5 + 1\right)$$

where x_{ij} is the expression of gene j in cell i. Finally, a Z-score per gene is calculated to form the normalized expression matrices X^A and X^B for genes G_A and G_B, respectively. We created a shared feature space for the two datasets spanning $G_A \cup G_B$ (Figure 1B). The shared feature space is partly equal to the expression matrices X^A and X^B and partly imputed:

$$\overline{X^A_{iu}} = \begin{cases} X^A_{iu} & \text{if } u \in G_A \\ \frac{1}{\sum_{v \in G_A} e_{uv}} \sum_{v \in G_A} e_{uv} X^A_{iv} & \text{if } u \in G_B \end{cases}$$

where $\overline{X^A_{iu}}$ is the normalized expression of cell i from species A for gene u in the shared feature space. The expression of within species genes does not change. For a cross-species gene, we imputed the expression by taking the weighted average of the expression of the within-species genes it is matched to. The weight between gene u and gene v is calculated as:

$$e_{uv} = 1 - \frac{\text{similarity}(h_u^{\text{ProtBERT}}, h_v^{\text{ProtBERT}})}{0.005}$$

where similarity calculates the cosine distance between the ProtBERT embeddings. The weights are scaled to the interval [0, 1] by dividing with the distance threshold. When BLAST is used instead, we used the (average) bitscore between the two genes directly, since the bitscore does not have to be inversed. The edge weight is set to 0 for gene pairs that do not match according to the threshold and filtering criteria. The resulting matrices $\overline{X^A}$ and $\overline{X^B}$ both span the same set of genes, and can thus be compared directly.

5.2.3 Cell embeddings

The shared feature space is put through two linear layers to create the cell embeddings (Figure 1C). Each linear layer is followed by a Rectified Linear Unit (ReLU) activation function. The first layer creates embeddings of length 64. The second layer creates embeddings of length 32. These embeddings are used to visualize the embedding space with a UMAP. The weights to embed the cells are shared across the species.

5.2.4 Training species-specific cell type classifier

We used these embeddings to train a separate classifier per species. We used a neural network consisting of one linear layer followed by a softmax activation function (Figure 1D). Both classifiers take the cell embedding as input and output cell type probabilities, $h^{A,\text{out}}$ or $h^{B,\text{out}}$, only for cell types belonging to its respective species. During training, cells are input only to the classifier of its corresponding species.

The loss to update the embedding and classification weights consists of two parts: 1) the classification loss, and 2) the alignment loss. Both losses are calculated separately per species. For the classification loss, we used the weighted cross-entropy loss between the predictions and targets:

$$L_{\text{cls}_A} = \tfrac{1}{N_A}\sum_{i=1}^{N_A}\sum_{t=1}^{T_A} w_t Y_{it}^{LS}\ \ln(h_{it}^{A,\text{out}})$$

where L_{cls_A} is the classification loss for species A. N_A and T_A are the number of cells and cell types in species A respectively. w_t is the weight for cell type t, explained further below. $h_{it}^{A,\text{out}}$ is the output of classifier A, specifically the probability that cell i belongs to cell type t. The one-hot encoded targets Y are modified with label smoothing to prevent overfitting and improve stability:

$$Y_{it}^{LS} = \begin{cases} 1-\varepsilon & \text{if } Y_i = t \\ \frac{\varepsilon}{T-1} & \text{otherwise} \end{cases}$$

where ε (=0.1) controls the smoothness. The weight of each cell type is updated every epoch based on the accuracy of that cell type:

$$w_t = (1-\text{acc}_t) * \alpha + 1$$

where acc_t is the accuracy of class t in the current epoch. α is a hyperparameter that controls the influence of the accuracy on the weight. We use $\alpha = 9$ such that the weights are in the interval [1,10] which restricts the relative difference in weight between cell types. By updating the weights, a cell type with a lower accuracy in the current epoch will have a higher weight in the next epoch and thus the predictions will shift to that cell type.

The alignment loss aims to integrate the embedding space across the species, such that cross-species cells with a similar gene expression are close in the embedding space:

$$L_{\text{align}_A} = \tfrac{1}{N^A}\sum_{i=1}^{N^A} \text{MSE}\left(\tfrac{1}{|N_i^{cross}|}\sum_{j\in N_i^{cross}} \overline{X_j^B}, \overline{X_i^A}\right)$$

where N^A is the number of cells of species A and N_i^{cross} are the 20 nearest cross-species neighbours for cell i. MSE calculates the mean squared error between the prediction of the shared features of neighbours j and the actual shared features for cell i. If the alignment loss is minimized, neighbours in the embedding space can be used to predict the gene expression. The final loss is a combination of the classifier loss, the alignment loss and a regularization loss:

$$L = L_{\mathrm{cls}_A} + L_{\mathrm{cls}_B} + L_{\mathrm{align}_A} + L_{\mathrm{align}_B} + \gamma \parallel \theta \parallel_2^2$$

where θ consists of all parameters in the model, and is used for the L2 regularization to prevent overfitting. γ is the weight of the L2 norm, which is set to 0.01. The model is trained for 200 epochs. We used the Adam optimizer with a learning rate of 0.001. The full training process takes around 30 minutes.

To efficiently use large scRNA-seq datasets, the neural network is trained in batches. A batch size of 5000 cells per species is used to speed up the training while still having enough cells per cell type. Instead of sequentially iterating over the dataset, each batch is randomly sampled from the full dataset, while accounting for the size of each cell type. More specifically, every cell is assigned a probability N^A / N_t^A or N^B / N_t^B , where N^A is the total number of cells of species A and N_t^A is the number of cells of species A belonging to cell type t. These probabilities are then used to sample a batch of cells per species with a similar number of cells for each cell type.

5.2.5 Transferring cell type predictions across species

After the neural network is trained, the cell types are transferred by using the classifiers on the species they were not trained on (Figure 1E). That is, we calculate $h^{B,\mathrm{out}}$ for cells of species A, and $h^{A,\mathrm{out}}$ for cells of species B. The transferred cell type for a single cell is the cell type with the highest probability. To aggregate the information of the single cells to the cell type, we calculate the fraction of cells that are predicted to match cell types across species, which forms a normalized confusion matrix for both transferring directions. We average the two matrices to create a combined matrix, where high values indicate reciprocal matches. The values in the combined matrix can be used to score a match.

5.2.6 Dataset

We evaluated TACTiCS on snRNA-seq data taken from the primary motor cortex of human, mouse and marmoset [20]. These datasets consist of 76k human cells, 159k mouse cells and 69k marmoset cells, respectively. The cell type distribution varies considerably across species. For instance, non-neuronal cells make up around a third of both mouse and marmoset cells, while only 5% of the human cells are non-neuronal. We use two resolutions of the cell labels assigned by the original authors: 1) a higher resolution, consisting of 45 cell types present in all species; and 2) a lower resolution, consisting of 20 human, 23 mouse and 22 marmoset subclass cell types. At the lower resolution not all cell types occur in all species. SMC is only present in mouse, while Meis2 and Peri are only present in mouse and marmoset. Species-specific cells are labeled with “NA” at the higher resolution.

5.2.7 Evaluation

The combined matrix cannot be evaluated using standard metrics for confusion matrices, such as precision or F1 score, since we cannot distinguish between false positives and false negatives. Instead, we focus on the matching scores from corresponding cell types in the combined matrix, which ideally should be 1. We define the Average Diagonal Score (ADS) as the average score of the diagonal entries, after excluding species-specific cell types. A high ADS indicates that many cell types are correctly and reciprocally matched. However, the ADS does not indicate how many cell types are correctly matched. To this end, we define the recall as the fraction of diagonal entries where the score is highest for both that row and column.

We compared TACTiCS to SAMap [8] and Seurat (version 4) [21]. SAMap is a cell type matching method that iterates between two steps. The first step matches the genes, which is initially done with BLAST on the DNA or protein sequences. Instead of taking the top-1 match, SAMap uses the BLAST bitscore directly in their model which allows for many-to-many matches. The second step uses the gene matches to first impute genes across species and then embed the cells by concatenating the principal components of the original expression and imputed expression. Then, the correlation between genes in the embedding space is used to update the gene matches. The two steps are repeated until the process converges.

Seurat can be used to transfer cell type labels from a reference to a query dataset. Since Seurat cannot use many-to-many matches, we use BLAST one-to-one matches for the data integration and label transfer. Since labels can only be transferred from the reference to the query dataset, we had to integrate the data twice for each pairwise comparison: once using one species as the reference and once using the other species as the reference.

5.2.8 Implementation

TACTiCS is implemented in Python 3.9. Pytorch [22] was used for the model architecture. The scRNA-seq data is stored as Anndata [23] objects, containing both the gene expression and the cell type annotations. The implementation of TACTiCS is available at https://github.com/kbiharie/TACTiCS.

As Tarashanky et al. have noted, the runtime of SAMap increases significantly for larger datasets, and we were unable to run SAMap for the full datasets [8]. Instead, we used SAMap on subsets of 50k cells per species. We subsampled the data to keep the cell type proportions similar while making sure that all cell types are included. During sampling we ensured that at least 50 cells were present in the subset. If a cell type contained less than 50 cells, all cells were included in the subset.

5.3 Results

5.3.1 Matching genes using sequence embeddings is comparable to sequence alignment with notable differences

First, we investigate how similar the gene matches returned by ProtBERT and BLAST are. We retrieved 17,435 human and 14,033 mouse protein sequences, discarding 47% of the human genes and 49% of the mouse genes for which we do not have the protein sequence. We used both ProtBERT and BLAST to generate gene matches.

For 13,935 human genes, we found a one-to-one mouse match using BLAST. For these human genes, we defined the ProtBERT match as the mouse gene with the most similar ProtBERT embedding. For 13,050 out of 13,935 human genes (94%), the BLAST match is identical to the ProtBERT match. Thus, the top-1 match is identical for the vast majority of genes. We ranked the BLAST matches according to the ProtBERT embedding distance to all mouse genes (Figure 2A). Most of the BLAST matches have a rank close to 1 and over 98% of the BLAST matches have a rank below 100. Additionally, 48% of the BLAST matches that differ from the ProtBERT match are in the top-5 and thus considered in the many-to-many matches. Thus, if the BLAST match is not considered to be the best match by ProtBERT, it is still relatively similar based on the embedding distance.

Next, we focus on the human genes for which the ProtBERT and BLAST match differ to investigate which method returns the most functionally similar match. We restrict the comparison to the 818 human genes where the human gene, the BLAST match and the ProtBERT match are expressed in at least one cell. We assess functional similarity here in terms of gene expression similarity across cell types. Therefore, we calculated the Pearson correlation coefficient across cell types in humans and mouse. We considered the harmonized cell types as defined in [20] (Figure 2B). For 568 out of 818 (69%) genes, the BLAST match has a higher gene

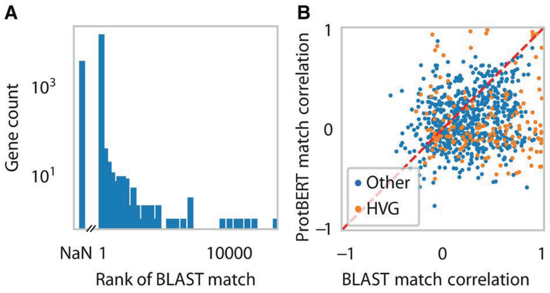
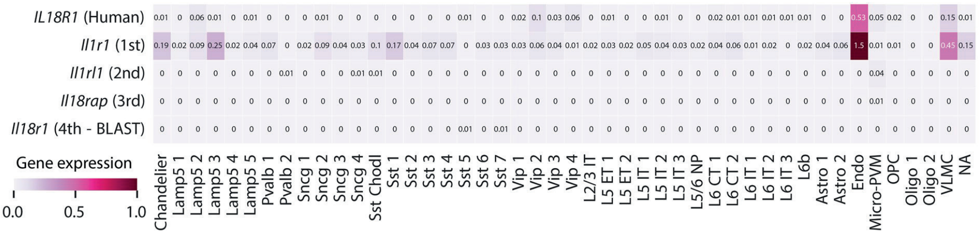


Figure 2. Comparison of ProtBERT and BLAST matches. A) Rank of BLAST match according to ProtBERT embedding distances. Rank 1 indicates that the best ProtBERT match and the best BLAST match are the same. Rank NaN indicates a human gene with a ProtBERT match but no BLAST one-to-one match. **B)** Scatterplot of the correlation of the expression of human and mouse genes when considering the best BLAST match (x-axis) and the best ProtBERT match (y-axis). The expression correlation is calculated as the Pearson correlation across the average expression profiles of the cross-species harmonized cell types. We omitted human genes where the BLAST match and ProtBERT match are the same. Gene matches where either the human gene, ProtBERT match or BLAST match is highly variable, are colored orange.

correlation than the ProtBERT match. This is to be expected since the harmonized cell types were defined using the BLAST matches. However, for some genes, the ProtBERT match has a higher correlation than the BLAST match. For example, human *IL18R1* is matched to mouse *Il1r1* according to ProtBERT with a correlation coefficient of 0.945, while BLAST matches the gene to mouse *Il18r1* with a correlation coefficient of 0.103 (Figure 3). Human *IL18R1* and mouse *Il1r1* both show an increased expression for the endothelial and VLMC cells, while mouse *Il18r1* does not show this pattern, and is lowly expressed in all cell types.

5.3.2 TACTiCS accurately matches cortical cell types across mouse and human

Now that we have seen that ProtBERT matches can be a powerful way to capture gene relationships, we use them in TACTiCS to match cell types in mouse and human cortex data. We use the Allen Brain Data, since the cell types have been carefully matched and harmonized by curators. We train TACTiCS for the human-mouse comparison for both the subclass and cross-species resolution. At the subclass resolution, TACTiCS returns the correct cell type for all 23 cell types that are present in both human and mouse (Figure 4A). The species-specific cell types, mouse Meis2, Peri and SMC, do not have a one-to-one match with a human cell type. Mouse Peri only matches human VLMC with a score of 0.5, but human VLMC matches mouse VLMC with a higher score of 1.0. Cell types present in both species have matching scores of ≥ 0.9 while wrong matches all have matching scores ≤ 0.5.



	Chandelier	Lamp5 1	Lamp5 2	Lamp5 3	Lamp5 4	Lamp5 5	Pvalb 1	Pvalb 2	Sncg 1	Sncg 2	Sncg 3	Sncg 4	Sst Chodl	Sst 1	Sst 2	Sst 3	Sst 4	Sst 5	Sst 6	Sst 7	Vip 1	Vip 2	Vip 3	Vip 4
IL18R1 (Human)	0.01	0	0.06	0.01	0	0.01	0	0	0	0.01	0	0	0	0	0	0	0	0.01	0	0	0.02	0.1	0.03	0.06
Il1r1 (1st)	0.19	0.02	0.09	0.25	0.02	0.04	0.07	0	0.02	0.09	0.04	0.03	0.1	0.17	0.04	0.07	0.07	0	0.03	0.03	0.03	0.06	0.04	0.01
Il1rl1 (2nd)	0	0	0	0	0	0	0	0.01	0	0	0	0.01	0.01	0	0	0	0	0	0	0	0	0	0	0
Il18rap (3rd)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Il18r1 (4th - BLAST)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.01	0	0.01	0	0	0	0

	L2/3 IT	L5 ET 1	L5 ET 2	L5 IT 1	L5 IT 2	L5 IT 3	L5/6 NP	L6 CT 1	L6 CT 2	L6 IT 1	L6 IT 2	L6 IT 3	L6b	Astro 1	Astro 2	Endo	Micro-PVM	OPC	Oligo 1	Oligo 2	VLMC	NA
IL18R1 (Human)	0	0.01	0	0	0.01	0	0	0.02	0.01	0.01	0.01	0.01	0.01	0	0	0.53	0.05	0.02	0	0	0.15	0.01
Il1r1 (1st)	0.02	0.03	0.02	0.05	0.04	0.03	0.02	0.04	0.06	0.01	0.02	0	0.02	0.04	0.06	1.5	0.01	0.01	0	0	0.45	0.15
Il1rl1 (2nd)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.04	0	0	0	0	0
Il18rap (3rd)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.01	0	0	0	0	0
Il18r1 (4th - BLAST)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 3. Average expression of human *IL18R1* and mouse matches across harmonized cell types. The mouse matches are ordered according to the ProtBERT embedding distances. BLAST matches human *IL18R1* to mouse *Il18r1*.

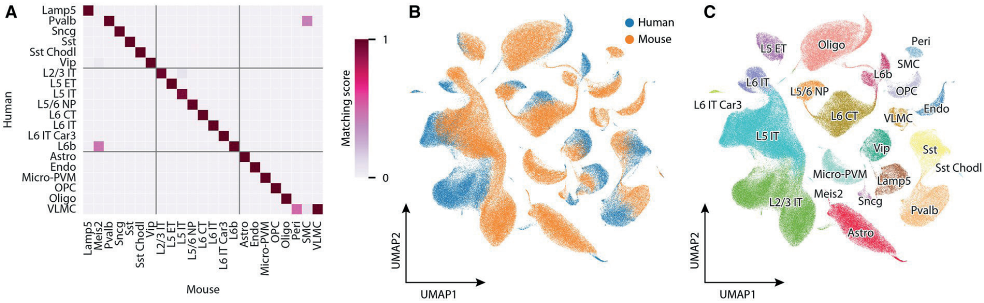


Figure 4. TACTiCS' performance when matching human and mouse cell types at the subclass resolution. A) Average confusion matrix of transferred cell types. **B)** UMAP of cell embeddings, colored by species. **C)** UMAP of cell embeddings, colored by cell type.

To get better insight into TACTiCS performance, we visualized the 32-dimensional cell embeddings using UMAP (Figure 4BC, S3). Individual human and mouse cells do not mix well in the embedding space, but the UMAP does seem to align at the cell type level, i.e. corresponding cell types either overlap partially in the embedding space, or are relatively close. For example, Vip cells form a large cluster with partly human and mouse cells separated, and cells of mixed origin in the middle. The Sncg cells also form a larger cluster, but the separation between the human and mouse cells is more visible. The Oligodendrocytes form two separate clusters, but they are closer to each other than to other cell types. The cell type proportions do seem to have an effect on the alignment in the embedding space. Cell types with a similar number of cells in human and mouse, such as Vip (6% in human and 2% in mouse), are clustered more coherently. Cell types with a large difference of occurrence within human and mouse, such as Astro (1% in human and 11% in mouse), form one small distinct cluster that is close to the larger cluster. The mouse-specific cell types Meis2, Peri, and SMC are (correctly) clustered separately from the human cells. Thus, the embedding space can align the cell types across the species, but not the individual cells. Note that this can be due to unresolved batch effects or actual biological differences between the two species.

At the cross-species resolution, TACTiCS returns correct matches for the majority of cell types, with a recall of 0.96 (Figure 5A, S1). The two cell types that are not properly matched, namely a L5-IT subtype and a Sncg subtype, are still matched with closely related cell types. The L5-IT subtype is matched with another L5-IT subtype and the Sncg subtype is matched to a subtype from the similar Lamp5 subclass.

To evaluate the performance of TACTiCS across species with variable evolutionary distance, we tested TACTiCS on cortical cell types between human-marmoset and mouse-marmoset (Table 1). At the subclass resolution, TACTiCS performs similar on all three comparisons with a recall of 1.0. At the cross-species resolution, TACTiCS performs best for the human-marmoset

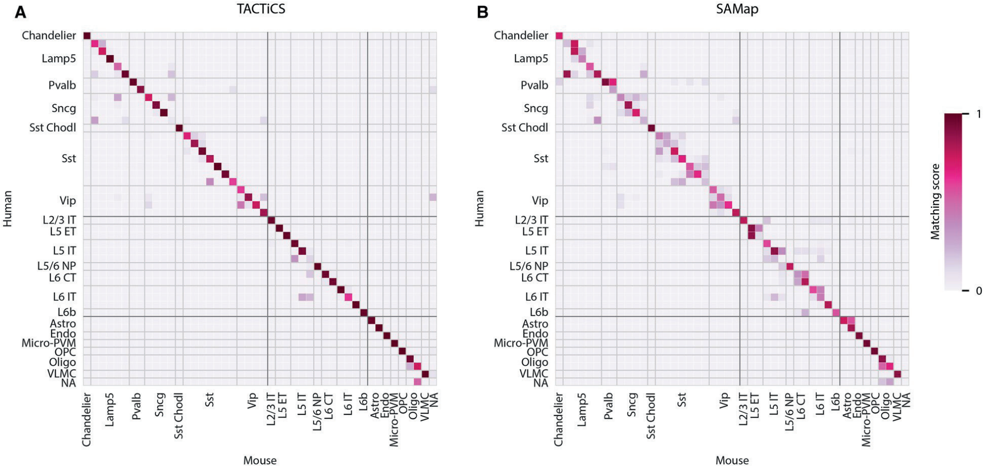


Figure 5. Performance of A) TACTiCS and B) SAMap when matching human and mouse cell types at cross-species resolution. Cross-species cell types are grouped per subclass (indicated with the light-gray lines) and class (indicated with dark-gray lines).

Table 1. ADS and recall for TACTiCS, Seurat, and SAMap on human, mouse, and marmoset.

Comparison	Method	Matching	Subclass		Cross-species	
			ADS	Recall	ADS	Recall
Hu-mo	TACTiCS	P (m:m)	0.991	1.000	0.856	0.956
Hu-mo	TACTiCS	B (m:m)	0.915	0.900	0.509	0.489
Hu-mo	TACTiCS	B (1:1)	0.992	1.000	0.724	0.778
Hu-mo	Seurat	B (1:1)	0.821	0.850	0.435	0.400
Hu-mo (50k)	TACTiCS	P (m:m)	0.894	0.900	0.780	0.822
Hu-mo (50k)	SAMap	P (m:m)	0.814	1.000	0.635	0.733
Hu-mo (50k)	SAMap	B (m:m)	0.827	1.000	0.630	0.800
Hu-ma	TACTiCS	P (m:m)	0.981	1.000	0.920	0.956
Hu-ma	TACTiCS	B (m:m)	0.891	0.900	0.848	0.889
Hu-ma	TACTiCS	B (1:1)	0.983	1.000	0.919	0.956
Hu-ma	Seurat	B (1:1)	0.906	1.000	0.697	0.822
Hu-ma (50k)	TACTiCS	P (m:m)	0.982	1.000	0.949	1.000
Hu-ma (50k)	SAMap	P (m:m)	0.892	1.000	0.816	0.978
Hu-ma (50k)	SAMap	B (m:m)	0.899	1.000	0.816	0.978
Mo-ma	TACTiCS	P (m:m)	0.990	1.000	0.735	0.733
Mo-ma	TACTiCS	B (m:m)	0.844	0.864	0.483	0.467
Mo-ma	TACTiCS	B (1:1)	0.991	1.000	0.770	0.778
Mo-ma	Seurat	B (1:1)	0.819	0.864	0.488	0.489
Mo-ma (50k)	TACTiCS	P (m:m)	0.928	0.909	0.730	0.733
Mo-ma (50k)	SAMap	P (m:m)	0.798	0.955	0.608	0.689
Mo-ma (50k)	SAMap	B (m:m)	0.823	0.955	0.637	0.689

comparison and worst for the mouse-marmoset comparison. These results indicate that the performance of TACTiCS is dependent on the evolutionary distance between the species, since the evolutionary distance to the closest common ancestors from human and marmoset (~40mya) is a lot less than human and mouse (~70mya).

5.3.3 TACTiCS outperforms SAMap and Seurat in matching cortical cell types across mouse, human, and marmoset

To benchmark TACTiCS, we compare its performance to SAMap and Seurat using three pairwise comparisons (human-mouse, human-marmoset, and mouse-marmoset). Across all comparisons, TACTiCS has a higher ADS and recall than SAMap and Seurat at the subclass resolution (Table 1). TACTiCS and SAMap perform well for all comparisons with a recall ≥0.95. Seurat performs well for the human-marmoset comparison, but the performance drops for the other two comparisons with a recall of 0.85 and 0.86 for the human-mouse and mouse-marmoset comparisons respectively. Although the resulting matches of TACTiCS and SAMap are similar, the scores assigned by TACTiCS to those correct matches is higher than SAMap. For instance, SAMap correctly matches human L6b to mouse L6b, but with a very low matching score equal to 0.47, while TACTiCS matches the same cell types with a matching score of 1.0. Interestingly, for the species-specific cell types, TACTiCS suggests matches that have a low score (0.04-0.5), allowing to detect the species-specific cell types. The performance of SAMap and Seurat for the species-specific cell types is not consistent across all cell types and comparisons. For example, SAMap correctly assigns zero scores to mouse Meis2, Peri and SMC in the human-mouse comparison, but incorrectly matches mouse SMC to marmoset Peri with a high matching score. Likewise, Seurat correctly assigns low scores to mouse Meis2 across all three comparisons, but incorrectly assigns higher scores to mouse Peri and SMC.

At the cross-species resolution the performance of all methods drops compared to the subclass level as expected, but the difference between the three methods becomes more apparent (Figure 5, S2). TACTiCS achieved the highest ADS and recall for the human-mouse and mouse-marmoset comparisons. SAMap has a higher recall than TACTiCS for the human-marmoset comparison, but not a better ADS. Seurat performs the worst across all three comparisons and achieves a recall of only 0.4 for the human-mouse comparison. For mismatches between subtypes, TACTiCS usually matches to subtypes within the same subclass, while SAMap regularly maps to cell types from another subclass. While both TACTiCS and SAMap partly match human Sncg to mouse Lamp5, SAMap additionally shows similarity between human Sncg and mouse Vip.

While the human and mouse cells did not overlap much in the UMAP of TACTiCS, Seurat consistently maps the query dataset onto the reference dataset (Figure S3, S4). However, the query dataset is not mapped equally onto the reference dataset, which leaves large regions of the clusters consisting of only one species. For both methods the mixing of species is the highest for the human-marmoset comparison and lowest for the human-mouse comparison.

To account for the differences in dataset size, we compare TACTiCS and SAMap on the same 50k subset. The performance of TACTiCS drops on the subset compared to the full dataset and does not match all common cell types correctly anymore at the subclass resolution. However, TACTiCS still outperforms SAMap at the higher resolution across all three comparisons.

5.3.4 Using ProtBERT matches improves the cell type matching for TACTiCS

Finally, we assessed the importance of using the ProtBERT embeddings to match genes compared to using BLAST on the final cell type matches. To this end, we trained TACTiCS based on the BLAST many-to-many matches and SAMap using the ProtBERT matches on the human-mouse data. For a fair comparison of ProtBERT to BLAST in SAMap, we only apply the embedding distance threshold to the ProtBERT matches, rather than filtering the gene matches thoroughly. Training TACTiCS at the cross-species resolution using the BLAST matches decreased the ADS and recall by a lot across all comparisons (Table 1). For SAMap, the performance remained similar, except for the human-mouse comparison where the recall decreased from 0.8 to 0.73 when ProtBERT matches were used instead of the BLAST matches.

Additionally, we trained TACTiCS on the BLAST one-to-one matches. At the subclass resolution, the ADS and recall remain similar if BLAST one-to-one is used instead of ProtBERT many-to-many. This is not the case for all comparisons at the cross-species resolution. The performance decreases for human-mouse, remains similar for human-marmoset and is increased for mouse-marmoset when BLAST one-to-one is used.

5.4 Discussion

Here, we present TACTiCS, a method to accurately match cell types from scRNA-seq data across species. We applied TACTiCS to match cell types across human, marmoset, and mouse motor cortex, species with different evolutionary distances to each other. Even though TACTiCS matches cell types from all three species with high confidence, we showed that human and marmoset cell types are considerably easier to match which correlates with their closer evolutionary distance. Furthermore, we showed that TACTiCS outperforms the state-of-the-art method SAMap on all comparisons with the biggest difference at a higher resolution in favor of TACTiCS. We should note that our evaluation is limited to using only three datasets from one tissue with a relatively small evolutionary distance, while SAMap was originally developed to match cell types across larger evolutionary distances [8].

Even though TACTiCS outperforms SAMap on the (finer) cross-species resolution, its performance drops as well. We would like to note that the cell types at this resolution were established by Bakken et al. by integrating datasets from the different species and clustering them in an embedding space [20]. This resulted in ambiguous clusters which were resolved manually by the authors to determine which cell types would be in one cross-species group. Since these matches are not perfect, it makes sense that we cannot achieve a perfect performance either.

Furthermore, the ground-truth matches used for evaluation are based on analyses performed using BLAST one-to-one matches, also causing unwanted differences when comparing results. This might explain why the performance of TACTiCS using BLAST one-to-one is comparable to using ProtBERT many-to-many matches. Here, we only see an improvement for species with larger evolutionary distances (i.e. human-mouse comparison).

All the results obtained by TACTiCS were obtained using the same hyperparameters, which have not been tuned. Although, tuning the hyperparameters could potentially improve matches between species, the advantage of the current set of hyperparameters is that they show robust performance across all pairwise-comparisons regardless of species and resolution (i.e. subclass or cross-species).

Gene matching is one of the main components of TACTiCS. We match genes based on the distance between their corresponding protein embeddings, which are generated using ProtBERT instead of the commonly used sequence similarity based on BLAST. Even though the top-1 matches of ProtBERT and BLAST are largely similar, we have shown that using ProtBERT instead of BLAST distances improves the performance of TACTiCS. When aligning sequences using BLAST, every amino acid is considered to be equally important, while we speculate that ProtBERT focuses more on functional domains. During further research, it would be interesting to dive deeper into the ProtBERT embedding space and see how this could be used to learn more about the relationships between cell types and the genes involved. A downside, however, of using ProtBERT distances is that the protein sequence is needed and as a consequence, we can only use coding genes. Using DNA sequence embedding models, e.g. DNABert [24], for non-coding genes, could in the future be used to overcome this limitation.

Some cell types, such as Meis2 and Peri in mice, are species-specific. A limitation of our current approach is that the classifiers we built in TACTiCS are missing a rejection option and therefore we cannot identify these species-specific cells automatically. Although we observed that TACTiCS usually assigns a low matching score to these species-specific cell types, it is, however, important to realize that the matching score represents the average accuracy of the two classifiers and does not represent an absolute measure of cell type similarity. For instance, if two human cell types are very similar, predictions for a mouse cell type may be split over these two human cell types (e.g. both get a score of 0.5). This is, for instance, the case with the Vip cross-species clusters in Figure 5A. This lower score indicates that there are similar human cell types in the data that both look like this mouse cell type. A high score, however, does not guarantee that the two cell types are very similar. It only indicates that these two cell types are most similar to each other and that they are transcriptionally very distinct from the other cell types in the dataset. In other words, the scores are summaries of the classification results, and as such, they are very much dependent on the cell types present in both datasets (i.e. the scores will change if one cell type is missing from one of the 2 species).

When inspecting the cell embeddings in the low dimensional space, we notice that the cells from difference species are not well mixed. Matching cell types, however, are closest to each other and species-specific cell types are more separated from all other cells. There are many data integration methods developed for single-cell data, such as scVI [25], that would achieve a significantly better integration. Since data integration is not the main goal of TACTiCS, we did not add an explicit mixing component to the loss function. The current loss function enforces that neighboring cells from the other species can predict the other cell's gene expression profile. This enforces cells of the same cell type to be the closest, but not to fully overlap. Adding a component to the loss that forces cells to be mixed (e.g. to have neighbors of both species) could greatly improve the integration. Alternatively, if good integration is a user's desire, an option would be to replace the component of TACTiCS that generates the cell embeddings with another data integration method such as scVI. The flexible architecture of TACTiCS allows the individual components (gene matching, cell embedding, and cell classification) to be easily replaced, extended, or integrated with different methods.

With TACTiCS we showed that using protein embeddings to match genes is a viable alternative to BLAST when matching cell types based on their scRNA expression levels across species. TACTiCS can accurately match cell types at different resolutions for large datasets, outperforming Seurat and SAMap. We envision that this fast and accurate cell type matching method, will make comparative analyses across species considerably easier, contributing to, e.g. to the study of cell type evolution or translational research.

Bibliography

1. Hodge RD, Bakken TE, Miller JA, Smith KA, Barkan ER, Graybuck LT, et al. Conserved cell types with divergent features in human versus mouse cortex. Nature. 2019; 1–8. doi:10.1038/s41586-019-1506-7
2. Tasic B, Yao Z, Graybuck LT, Smith KA, Nguyen TN, Bertagnolli D, et al. Shared and distinct transcriptomic cell types across neocortical areas. Nature. 2018;563: 72–78. doi:10.1038/s41586-018-0654-5
3. Siletti K, Hodge R, Mossi Albiach A, Lee KW, Ding S-L, Hu L, et al. Transcriptomic diversity of cell types across the adult human brain. Science. 2023;382: eadd7046. doi:10.1126/science.add7046
4. Altschul SF, Gish W, Miller W, Myers EW, Lipman DJ. Basic local alignment search tool. J Mol Biol. 1990;215: 403–410. doi:10.1016/S0022-2836(05)80360-2
5. Xu C, Lopez R, Mehlman E, Regier J, Jordan MI, Yosef N. Probabilistic harmonization and annotation of single-cell transcriptomics data with deep generative models. Mol Syst Biol. 2021;17: e9620. doi:10.15252/msb.20209620
6. Crow M, Paul A, Ballouz S, Huang ZJ, Gillis J. Characterizing the replicability of cell types defined by single cell RNA-sequencing data using MetaNeighbor. Nat Commun. 2018;9: 884. doi:10.1038/s41467-018-03282-0
7. Johnson TS, Wang T, Huang Z, Yu CY, Wu Y, Han Y, et al. LAmbDA: Label Ambiguous Domain Adaptation Dataset Integration Reduces Batch Effects and Improves Subtype Detection. Bioinformatics. 2019. doi:10.1093/bioinformatics/btz295
8. Tarashansky AJ, Musser JM, Khariton M, Li P, Arendt D, Quake SR, et al. Mapping single-cell atlases throughout Metazoa unravels cell type evolution. Elife. 2021;10. doi:10.7554/eLife.66747
9. Liu X, Shen Q, Zhang S. Cross-species cell-type assignment from single-cell RNA-seq data by a heterogeneous graph neural network. Genome Res. 2023;33: 96–111. doi:10.1101/gr.276868.122
10. Botvinnik OB, Vemuri VNP, Pierce NT, Logan PA, Nafees S, Karanam L, et al. Single-cell transcriptomics for the 99.9% of species without reference genomes. bioRxiv. 2021; 2021.07.09.450799. doi:10.1101/2021.07.09.450799
11. Kabir MH, Djordjevic D, O’Connor MD, Ho JWK. C3: An R package for cross-species compendium-based cell-type identification. Comput Biol Chem. 2018;77: 187–192. doi:10.1016/j.compbiolchem.2018.10.003
12. Villegas-Morcillo A, Makrodimitris S, van Ham RCHJ, Gomez AM, Sanchez V, Reinders MJT. Unsupervised protein embeddings outperform hand-crafted sequence and structure features at predicting molecular function. Bioinformatics. 2020. doi:10.1093/bioinformatics/btaa701
13. Rives A, Meier J, Sercu T, Goyal S, Lin Z, Liu J, et al. Biological structure and function emerge from scaling unsupervised learning to 250 million protein sequences. Proc Natl Acad Sci U S A. 2021;118. doi:10.1073/pnas.2016239118
14. Heinzinger M, Elnaggar A, Wang Y, Dallago C, Nechaev D, Matthes F, et al. Modeling aspects of the language of life through transfer-learning protein sequences. BMC Bioinformatics. 2019;20: 723. doi:10.1186/s12859-019-3220-8
15. Elnaggar A, Heinzinger M, Dallago C, Rehawi G, Wang Y, Jones L, et al. ProtTrans: Toward Understanding the Language of Life Through Self-Supervised Learning. IEEE Trans Pattern Anal Mach Intell. 2022;44: 7112–7127. doi:10.1109/TPAMI.2021.3095381
16. Michielsen L, Reinders MJT, Mahfouz A. Hierarchical progressive learning of cell identities in single-cell data. Nat Commun. 2021;12: 1–12. doi:10.1038/s41467-021-23196-8
17. Yuan M, Chen L, Deng M. scMRA: a robust deep learning method to annotate scRNA-seq data with multiple reference datasets. Bioinformatics. 2022;38: 738–745. doi:10.1093/bioinformatics/btab700
18. UniProt Consortium. UniProt: the Universal Protein Knowledgebase in 2023. Nucleic Acids Res. 2023;51: D523–D531. doi:10.1093/nar/gkac1052
19. Wolf FA, Angerer P, Theis FJ. SCANPY: Large-scale single-cell gene expression data analysis. Genome Biol. 2018;19: 15. doi:10.1186/s13059-017-1382-0
20. Bakken TE, Jorstad NL, Hu Q, Lake BB, Tian W, Kalmbach BE, et al. Comparative cellular analysis of motor cortex in human, marmoset and mouse. Nature. 2021;598: 111–119. doi:10.1038/s41586-021-03465-8
21. Hao Y, Hao S, Andersen-Nissen E, Mauck WM, Zheng S, Butler A, et al. Integrated analysis of multimodal single-cell data. Cell. 2021;0. doi:10.1016/j.cell.2021.04.048
22. Paszke A, Gross S, Massa F, Lerer A, Bradbury J, Chanan G, et al. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In: Wallach H, Larochelle H, Beygelzimer A, d\textquotesingle Alché-Buc F, Fox E, Garnett R, editors. Advances in Neural Information Processing Systems 32. Curran Associates, Inc.; 2019. pp. 8024–8035. Available: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf
23. Virshup I, Rybakov S, Theis FJ, Angerer P, Alexander Wolf F. anndata: Annotated data. bioRxiv. 2021. p. 2021.12.16.473007. doi:10.1101/2021.12.16.473007
24. Ji Y, Zhou Z, Liu H, Davuluri RV. DNABERT: pre-trained Bidirectional Encoder Representations from Transformers model for DNA-language in genome. Kelso DJ, Kelso J, editors. Bioinformatics. 2021. doi:10.1093/bioinformatics/btab083
25. Lopez R, Regier J, Cole MB, Jordan MI, Yosef N. Deep generative modeling for single-cell transcriptomics. Nat Methods. 2018;15: 1053–1058. doi:10.1038/s41592-018-0229-2

chapter 6

Predicting cell population-specific gene expression from genomic sequence

Lieke Michielsen, Marcel J.T. Reinders, Ahmed Mahfouz

This chapter is published in: *Frontiers in Bioinformatics* (2024) 4:1347276, doi: 10.3389/fbinf.2024.1347276
Supplementary material is available online at:
https://www.frontiersin.org/articles/10.3389/fbinf.2024.1347276/full#h12

Most regulatory elements, especially enhancer sequences, are cell population-specific. One could even argue that a distinct set of regulatory elements is what defines a cell population. However, discovering which non-coding regions of the DNA are essential in which context, and as a result, which genes are expressed, is a difficult task. Some computational models tackle this problem by predicting gene expression directly from the genomic sequence. These models are currently limited to predicting bulk measurements and mainly make tissue-specific predictions. Here, we present a model that leverages single-cell RNA-sequencing data to predict gene expression. We show that cell population-specific models outperform tissue-specific models, especially when the expression profile of a cell population and the corresponding tissue are dissimilar. Further, we show that our model can prioritize GWAS variants and learn motifs of transcription factor binding sites. We envision that our model can be useful for delineating cell population-specific regulatory elements.

6.1 Introduction

In multicellular organisms, every cell has the same DNA apart from somatic mutations. Yet its function and the related proteins and genes expressed vary enormously. This is among others caused by transcriptional and epigenetic regulation. Proteins that bind the DNA sequence around the transcription start site (TSS) control whether a gene is transcribed in a cell [1,2]. Which transcription factors, and thus which DNA binding motifs, are essential differ per cell population [1–4]. As such, mutations in regulatory regions might affect specific tissues or cell populations differently. Improving our understanding of these regulatory mechanisms will help us relate genomic functions to a phenotype.

For example, while promoter sequences are identical across the four major human brain cell populations (neurons, oligodendrocytes, astrocytes, and microglia), almost all enhancer sequences, the regions in the DNA where a transcription factor binds, are cell population-specific [3]. These population-specific regulatory elements are discovered by combining single-cell measurements of different data types, including chromatin accessibility, ChIP-seq, and DNA methylation. Bakken et al., for instance, identified differentially methylated and differentially accessible regions across neuronal cell populations in the human brain, albeit with little overlap [5]. This emphasizes the complexity of transcriptional regulation and the need for more measurements to fully resolve these mechanisms at the cell population-specific level.

An alternative approach would be to train a computational model that directly predicts gene expression from the genomic sequence around the TSS. This way, we can learn which regulatory elements are important for transcriptional regulation in different contexts. Several computational methods have been developed for this task [6–12]. These methods have in common that they one-hot encode the DNA sequence and input this to either a convolutional neural network (CNN) or transformer. ExPecto, Xpresso, and ExpResNet predict expression measurements from bulk RNA-sequencing, while Basset, Basenji, BPNet, and the Enformer model predict regulatory signals, such as cap analysis gene expression (CAGE) reads or TF binding from CHIP-nexus.

A promising application of these models is to prioritize variants that have been identified using genome-wide association studies (GWAS) [6,13]. Using GWAS many potential disease-associating variants have been identified [14–16]. Within each locus, however, it is often challenging to pinpoint which variant is causal and which gene is affected by the variant.

These current computational gene prediction models, however, are designed for predicting bulk gene expression data. This means that they are either tissue-specific or could be applied to FACS-sorted cells [13]. Since transcriptional regulation is even more context-specific, the resolution of current methods is not sufficient for heterogeneous tissues where single-cell RNA-sequencing (scRNA-seq) has revealed hundreds of cell populations [5,17,18]. To increase the resolution, the models would ideally be trained on scRNA-seq data.

Here, we present scXpresso, a deep learning model that uses a CNN to learn cell population-specific expression in scRNA-seq data from genomic sequences. Since single-cell and bulk data have different characteristics and distributions, we explored whether this type of model is suitable for single-cell data. We show that (i) cell population-specific models outperform tissue-specific models on several tissues from the Tabula Muris, (ii) increasing the resolution improves the predictions for human brain cell populations, and (iii) *in-silico* saturation mutagenesis of the input sequence can be used to prioritize GWAS variants.

6.2 Materials and methods

6.2.1 Architecture of scXpresso

scXpresso is a one-dimensional convolutional neural network (CNN) adapted from the (bulk gene expression-based) Xpresso model [9] (Figure 1A, S1). The input to the CNN is four channels with the one-hot encoded sequence around the transcription start site (TSS) (7kb upstream and 3.5kb downstream). Every channel represents one of the four nucleotides (A, C, T, G). For some positions, the exact nucleotide is not known (e.g. any nucleic acid (N) or a purine nucleotide (R)). The exact coding scheme for such positions is shown in Table S1. The CNN consists of two convolutional layers. The output of the convolutional layers is flattened and concatenated with the half-life time features. Together, this is subsequently fed into a fully connected (FC) layer(s). The output of the FC layers is the aggregated expression per tissue or for each cell population.

Comparing scXpresso to Xpresso, there are three main differences: 1) we designed scXpresso as a multitask model so that it predicts the expression of multiple cell populations simultaneously. 2) We decreased the number of half-life time features from eight to five; the three features we removed (5’ UTR, ORF, and 3’ UTR GC content) correlated less with half-life time, so we removed them to make the model less complex [9,19,20]. Furthermore, removing these three half-life time features from the original Xpresso model did not lower its performance (Table S2). 3) For the multitask model, there is only one FC layer. For the other models, which we use to make tissue-specific predictions as a comparison, we used two FC layers.

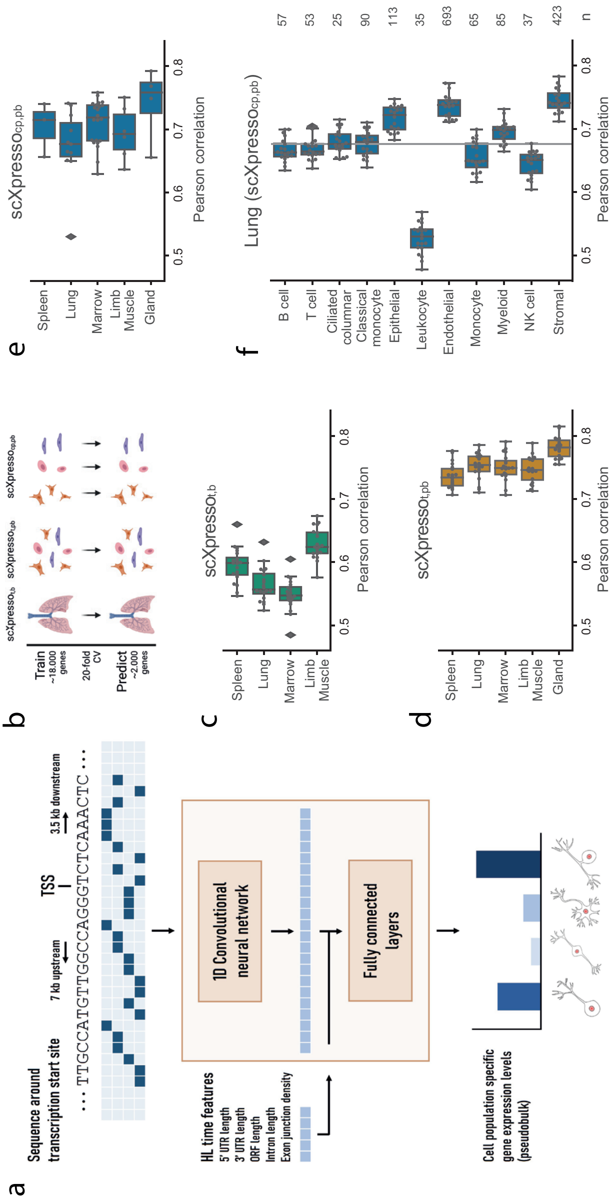


Figure 1. Schematic overview of scXpresso and performance on Tabula Muris datasets. A) We one-hot encode the DNA sequence around the transcription start site (TSS) and input this to a one-dimensional convolutional neural network (CNN). The output of the CNN is flattened and concatenated with the five half-life time features. The fully connected layers output the cell population's specific gene expression levels simultaneously (Figure S1, see Methods). **B)** Schematic overview of the experiment. **C-D)** Performance of scXpresso$_{t,b}$ (tissue-specific (t) model on bulk (b) data) and scXpresso$_{t,pb}$ (tissue-specific model on pseudo-bulk (pb) data), respectively. Every dot is the performance (Pearson correlation) across one fold of the 20-fold CV. **E)** Performance of scXpresso$_{cp,pb}$ (cell population-specific (cp) model on pseudo-bulk data) summarized per tissue. Every dot represents the model's performance on a cell population in that tissue (median Pearson correlation across the 20 folds). **F)** Performance of scXpresso$_{cp,pb}$ on the different lung cell populations. The grey line indicates the median performance across all cell populations. Every dot is the performance across one fold of the 20-fold CV.

6.2.2 Training scXpresso

We split the genes into a train, validation, and test dataset and evaluated using 20-fold cross-validation. These sets are the same across all experiments (i.e. one train, validation, and test set for mouse genes and one for human genes) such that the results of different models can be compared. We update the weights of scXpresso using the Adam optimizer based on the mean square error loss on the training set. The initial learning rate is set to 0.0005 and if the loss on the validation set is not improved from 5 epochs, the learning rate is reduced by a factor of 10. We train the model for 40 epochs and the model with the lowest loss on the validation set is used for evaluation on the test dataset. Since there is always some stochasticity when training a CNN, we always train 5 models and average the predictions. We used the following software packages for training the model: Pytorch (version 1.9.0) [21], CUDA (version 11.1), cuDNN (version 8.0.5.39), and Python (version 3.6.8).

6.2.3 Datasets

Tabula Muris. The single-cell Tabula Muris data [22] for the five different tissues (gland, spleen, lung, limb muscle, and bone marrow) and two different protocols (10X and FACS-based Smart-seq2) were downloaded from: https://figshare.com/projects/Tabula_Muris_Transcriptomic_characterization_of_20_organs_and_tissues_from_Mus_musculus_at_single_cell_resolution/27733. To extract input features, we downloaded the reference genome (MM10-PLUS) that was used during the alignment from: https://s3.console.aws.amazon.com/s3/object/czb-tabula-muris-senis?region=us-west-2&prefix=reference-genome/MM10-PLUS.tgz.

The four bulk datasets (spleen, lung, limb muscle, and bone marrow) from the Tabula Muris were downloaded from https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE132040 [23]. For the bulk data, we used the same reference genome as for the single-cell data.

Human motor cortex data. The human motor cortex data from the Allen Institute [5] was downloaded from the Cytosplore Comparison Viewer. We downloaded the reference genome (version GRCh38.p2) and corresponding GTF file with information about the location of transcription start sites of the genes here: (https://www.gencodegenes.org/human/release_22.html)

6.2.4 Aggregated expression values

First, we normalized the count matrices. For the single-cell datasets, we performed library size normalization in the same way as The Tabula Muris Consortium: i.e. counts per million for the smart-seq2 data and counts per ten thousand for the 10X data [22]. For the bulk Tabula Muris data, we performed TPM normalization. For the single-cell datasets, we used the annotations defined by the authors to aggregate the expression values per tissue or per cell population using $\log_{10}(\text{mean}(x))$ (without pseudocount) into pseudobulk values. The advantage of not adding a pseudocount is that the distribution looks more like a normal distribution, which makes it easier to train the models (Figure S2). A limitation, however, is that we could not calculate the exact value for genes that were not expressed in any of the cells. For these genes, we replaced the pseudobulk values with -4 in the Tabula Muris and -5 in the motor cortex dataset, since this extrapolated well (Figure S2). For the bulk data, we aggregated over the samples instead of the cells. Here, we set the genes that are not expressed in any of the samples to -4. We standardized the expression values before running the model such that the average expression of all genes in each cell population or tissue is zero and the standard deviation is one. Before analyzing the results and comparing the predictions across cell populations, we undid the z-score normalization but kept the log normalization.

6.2.5 Input features

Sequence around the transcription start site. Before extracting the sequences around the transcription start site, we removed genes that are transgenes, ERCC spike-ins, genes without a coding region, and genes on the Y chromosome. This resulted in 20,467 mouse genes and 18,138 human genes. Some genes had multiple transcripts. We downloaded a list with canonical transcripts for each gene from biomart and we used the transcript and transcription start site belonging to the canonical transcript. If the canonical transcript was not defined, we used the transcript that had the longest coding region. After having defined the transcription start site for each gene, we used seqkit [24] to extract sequences from the FASTA file containing the reference genome.

Half-life time features. For every gene, we extracted five half-life time features: 5’ UTR length, 3’ UTR length, ORF length, intron length, and exon junction density ($\frac{\#exons}{\mathrm{length}(ORF)} * 1000$). We obtained these features by first filtering the GTF files for the canonical or longest transcript. The 5’ UTR length is the length of the sequence from the start of the first exon to the start codon. The 3’ UTR length is the length of the sequence from the last coding sequence to the end of the last exon. The ORF length is the sum of the length of the coding sequences. The intron length is the length of the transcript minus the length of the ORF, 5’ UTR, and 3’ UTR. All features are log-normalized using $\log_{10}(x + 0.1)$ and afterwards z-scaled.

6.2.6 Evaluating the predictions

For every gene in the test dataset, we averaged the predictions of the five models we trained. We evaluated the performance for every cell population by calculating the Pearson correlation between the true and predicted expression of the genes in the test set. To evaluate the increase in performance between the tissue-specific (t) pseudobulk (pb) and cell population-specific (cp) pseudobulk (pb) model on the Tabula Muris datasets, we calculate: $\Delta_{cp,t} = \text{median Pearsoncorrelation}(scEP_{cp,pb}) - \text{median Pearsoncorrelation}(scEP_{t,pb})$. On the motor cortex dataset, we also evaluated the performance of each gene by calculating the Pearson correlation between the true and predicted expression per cell population.

6.2.7 In-silico saturation mutagenesis

For *CACNA1I*, we mutated all positions *in-silico*, which means we tested all possible substitutions at every position. We undid the z-score normalization and calculated the difference between the original (wild-type) prediction and the mutated prediction. The prediction models used during these experiments were the models where *CACNA1I* itself was originally in the test set. For every position, we only plotted one predicted difference in expression in Figure 4E. This is the substitution that was predicted to have the largest absolute effect. We downloaded the locations of the candidate *cis*-regulatory elements that fall within the input region for *CACNA1I* from screen registry v3 (release date 2021) [25]. When plotting the difference between two cell populations, we ignored the positions where one is positive and the other predicts a negative effect. This rarely happened and if it was the case, the predicted effect was very small.

For the 2000 highly variable genes, selected using scanpy [26], we applied ISM similar as described for *CACNA1I.* For every position we then calculated the average maximum absolute predicted effect:

$$y_{\max}(i) = \tfrac{1}{2000}\sum_{g \in HVG} \max_{alt \in \{A,C,G,T\}, alt \neq ref} | y_{pred,g,ref}(i) - y_{pred,g,alt}(i) |$$

where i indicates the genomic position, HVG is the list of highly variable genes, ref indicates the reference allele, and alt indicates the alternative allele.

6.2.8 Comparison to other models

Enformer. Enformer uses the DNA sequence to predict reads for 5,313 human tracks which include CAGE, DNAse, CHIP, and ATAC-seq [11]. Here, we only looked at the effect of a variant on the CAGE tracks that are related to the brain (77 tracks in total, see Table S3). Enformer predicts the effect of variants on 128bp bins. When predicting the effect of a variant on the CAGE reads, we looked at the effect on the bin containing the TSS.

ExPecto. ExPecto predicts gene expression for 218 tissues and cell lines [8]. Here, we only focused on 27 outputs that are related to the brain (Table S4). We used the ExPecto web server to predict the effect of the variants (https://hb.flatironinstitute.org/expecto/?tabId=3). ExPecto is trained using Hg19 instead of Hg39. We used the R-package SNPlocs.Hsapiens.dbSNP155.GRCh37 (v 0.99.23) to lift-over the variants. Using ExPecto we could not predict the effect of all variants, since for some variants there was no location in Hg19 found, some were too far away from a TSS, and some were linked to a different gene than we were interested in (see Table S5 for an explanation per variant).

Xpresso. We trained the Xpresso model on bulk RNA-seq data from the precentral gyrus [9]. The data from two individuals were downloaded from the Allen Human Brain Atlas: https://human.brain-map.org/static/download (H0351.2001, H0351.2002). We used the normalized matrices. Labels were created as described in the Xpresso paper: we took the median expression across the 6 precentral gyrus samples, log-normalized the output using $\log_{10}(x + 0.1)$, and z-score normalized the expression. Similar to scXpresso, we trained the model using

20-fold cross-validation. Per fold, we trained 10 runs and used the model with the lowest MSE on the validation data (as described in [9]). Afterwards, we predicted the effect of the variants. We could not predict the effect of all variants, since some genes were not measured in the bulk RNA-seq data and for some genes, there were no Xpresso input features defined (see Table S5 for an explanation per variant).

6.3 Results

6.3.1 Predicting cell population-specific gene expression using scXpresso

Here, we present scXpresso, a multitask convolutional neural network (CNN) to predict cell population-specific gene expression using genomic sequences only (Figure 1A, S1). We developed scXpresso by adapting the Xpresso model [9], which was originally designed for bulk data, to single-cell data. Similar to Xpresso, we use two types of input to the model: (1) the DNA sequence around the transcription start site (TSS) (7kb upstream- 3.5kb downstream) to model transcription, and (2) five half-life time features (5’ UTR length, 3’ UTR length, ORF length, intron length, and exon junction density) to model mRNA degradation. We input the one-hot encoded DNA sequence into a CNN. The output of the CNN is concatenated with the half-life time features and fed to a fully connected network (see Methods). Since our model is a multitask CNN, the desired output of the fully connected network is the gene expression for every cell population. We predict expression per cell population instead of per cell to achieve more stable predictions with less noise as single-cell data is known to be quite sparse. To obtain one expression value per cell population, we aggregate the single-cell expression into pseudobulk measurements (see Methods).

Since single-cell and bulk data have different characteristics, we tested whether scXpresso performs equally well on single-cell and bulk data. We used scRNA-seq data from five different tissues (limb muscle, spleen, gland, marrow, and lung) from the Tabula Muris [22] (Table S6). Here, we used cells isolated via FACS that were sequenced using the Smart-seq2 protocol. Using the annotations defined by the authors, we aggregate the values per cell population and per tissue into pseudobulk values. For four tissues (limb muscle, spleen, marrow, and lung), there are also bulk RNA-sequencing datasets available (Table S7). We compared the pseudobulk to the bulk expression per tissue and noticed that these are indeed correlated ($r_{muscle} = 0.69$, $r_{spleen} = 0.71$, $r_{marrow} = 0.50$, $r_{lung} = 0.67$) (Figure S3).

Next, we trained three different models: 1) a tissue-specific (t) model on the bulk (b) values ($scXpresso_{t,b}$), 2) a tissue-specific model on the pseudobulk (pb) values ($scXpresso_{t,pb}$), 3) a cell population-specific (cp) model on the pseudobulk values ($scXpresso_{cp,pb}$) (Figure 1B). The cell population-specific model is, in contrast to the tissue-specific models, a multitask model that predicts the expression of all cell populations in a tissue simultaneously. We evaluated the performance of the models by calculating the Pearson correlation between the true and predicted expression values. In general, the tissue-specific models trained on pseudobulk reach higher performance than the models trained on bulk (Figure 1C-D).

Even though the bulk and pseudobulk values are correlated, the pseudobulk distributions are bimodal compared to the normally distributed bulk data (Figure S3-4). This turns the problem more into a classification problem (is a gene low or high expressed), which might be easier to learn. On average, predicting cell population-specific expression is more difficult than predicting tissue-specific expression (Figure 1D-E): scXpresso$_{cp,pb}$ performs slightly worse than scXpresso$_{t,pb}$ (median correlation of 0.71 vs 0.75), but still better than scXpresso$_{t,b}$ (0.58).

One of the adaptations to Xpresso is that scXpresso$_{cp,pb}$ is a multitask model. This slightly increases the performance compared to a single-task model (Figure S5) but mainly makes the model computationally more efficient. The marrow-FACS dataset, for instance, contains 22 cell populations. Since the single-task and multitask models need the same training time (approximately 30-60 minutes), this gives a 22x speed up.

The Tabula Muris scRNA-seq datasets were generated using two different protocols: 10X Genomics, a droplet-based method, and FACS-based Smart-seq2, a plate-based method. When comparing scXpresso$_{t,pb}$ and scXpresso$_{cp,pb}$ trained on the two different protocols, e.g. lung-droplet vs. lung-FACS, we conclude that they perform equally well (Figure 1DE, S6-7). Depending on the tissue and cell population, one performs slightly higher than the other, but there are no significant differences. This is as expected since the pseudobulk values of both protocols are highly correlated (Pearson correlation > 0.85) (Figure S8). Hence, the protocol used to create the single-cell dataset does not influence the results.

For scXpresso$_{cp,pb}$, we tested how the two types of input features, DNA sequence and half-life time, influence the performance. We tested different lengths of the input sequence and whether one of the two features was enough to predict expression (Figure S9). A range of different sequence lengths results in the same performance (3.5-3.5, 7-3.5, and 10-5kb upstream-downstream). A longer sequence gives more information but also adds more noise. Since the model also becomes more complex, more parameters have to be learned and it takes more time and memory to train the model. Therefore, we decided to use 7kb upstream and 3.5kb downstream for further experiments. We also observed that adding the half-life time features results in higher performance, suggesting that these features are not easily captured from DNA sequences alone.

For the cell population-specific models, the performance varies considerably across different populations (Figure 1E). Comparing the populations in the lung dataset, for instance, the performance of the endothelial cells is very high compared to leukocytes (Figure 1F, S10). In general, the performance of scXpresso increases when more genes and cells are measured in a population (Figure 1F, S11). The leukocyte population is small (35 cells) and fewer genes are non-zero compared to other cell populations in the lung (8,678 out of 20,467 vs. 12,715 on average). The ciliated cell population, on the other hand, is also small (25 cells), but this model reaches a higher performance. In this cell population, however, more genes were non-zero (11,717) compared to the leukocyte population. Hence, to train the model, we need a good representation of the cell population that includes enough expressed genes.

In all previous experiments, we evaluated scXpresso using 20-fold cross-validation with the genes randomly divided over the folds. The results could be positively biased if genes from the same chromosome are in different folds. Therefore, we also evaluated the models using cross-chromosomal cross-validation. This slightly reduces the models' performance, but the difference is not significant (lowest p-value = 0.11 for myeloid cells, two-sample Wilcoxon rank sum test) (Figure S12).

6.3.2 Cell population-specific models outperform tissue-specific models

Now that we know that all models are well-trained, we predicted cell population-specific expression using the three different models to see whether increasing the resolution of the models increases the performance (Figure 2A). Since scXpresso$_{t,b}$ and scXpresso$_{t,pb}$ were trained using tissue-specific expression values, these models predict the same value for every cell population. On all datasets, scXpresso$_{cp,pb}$ outperforms the tissue-specific models, which shows the benefit of training the models on a higher resolution (Figure 2B, S13A). Especially in more heterogeneous tissues, where the gene expression of cell populations is weakly correlated to the corresponding tissue, we see a large improvement (Figure 2C, S13B). For the lung-FACS dataset, for instance, the performance increases the most for immune cell populations ($\Delta_{cp,t}$ for B cells: 0.11, NK cells: 0.11, T cells: 0.09; see Methods) and the least for lung-specific populations ($\Delta_{cp,t}$ for stromal cells: 0.01, endothelial cells: 0.03, epithelial cells: 0.05). In the B cells in the lung, 4,081 genes are not expressed in any of the cells and thus have a log-normalized expression of-4, but for which the tissue-specific model predicts a positive log-normalized expression value (Figure 2D). In contrast, the model trained on B cells predicts a lower expression for these genes (Figure 2E). Almost all these genes, however, are expressed in the lung (in the non-B cells), the lung-model learned this correctly too (Figure 2F).

Some of the Tabula Muris datasets contain similar cell populations. For instance, B cells, macrophages, and T cells are measured in four, three, and three tissues, respectively. We hypothesized that if our models are cell population-specific, they should accurately predict the expression of a cell population in one tissue with a model trained on the same cell population but from another tissue (even though a cell's tissue will slightly change the expression for (some) genes). To test this, we predicted the expression for common cell populations using three different types of models: 1) scXpresso$_{cp,pb}$ trained on the same cell population, but from a different tissue, 2) scXpresso$_{cp,pb}$ trained on a different cell population, but from the same tissue, 3) scXpresso$_{t,pb}$ trained on the same tissue (Figure 3A). For example, we predict the expression of B-cells in the limb muscle, using 1) a model trained on B-cells in the lung, 2) a model trained on endothelial cells in the limb muscle, and 3) a model trained on the limb muscle. Again, the cell population-specific models outperform the tissue-specific models, even though they predict either a different dataset or a different cell population than they were trained on (Figure 3B, S14-15). This indicates that if you want to train a model for a cell population from a specific tissue where no single-cell data is available, you are better off using a model trained on a similar cell population from a different tissue than relying on a tissue-specific model. Whether a model trained on a different cell population and the same tissue performs better than a model trained on the same cell population but a different tissue, differs per tissue and cell population. For example, when predicting the expression

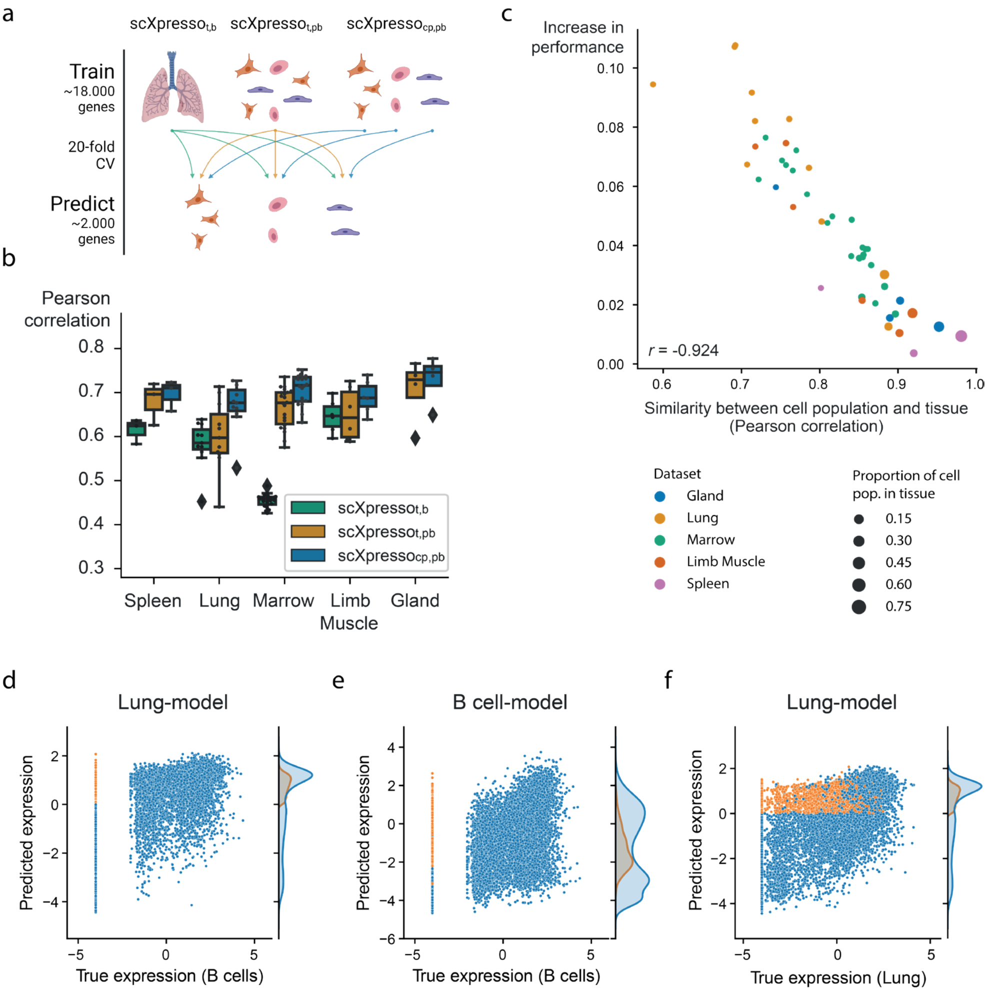


Figure 2. Comparison of the three scXpresso models for making cell population-specific predictions. A) Schematic overview of the experiment. **B)** Boxplot showing the performances of scXpresso$_{t,b}$ (tissue-specific (t) model on bulk (b) data), scXpresso$_{t,pb}$ (tissue-specific model on pseudo-bulk (pb) data), and scXpresso$_{cp,pb}$ (cell population-specific (cp) on pseudobulk (pb) data) on the cell population-specific task. Every point in the boxplot is the performance of a model on one cell population in that tissue (median Pearson correlation across the 20 folds). **C)** Similarity between a cell population and corresponding tissue (Pearson correlation between the true pseudobulk expression values) vs. the increase in performance ($\Delta_{cp,t}$, median Pearson correlation of scXpresso$_{cp,pb}$ - scXpresso$_{t,pb}$). Every dot is a different cell population and the colors represent the different tissues. **D-F)** Comparing the predictions made by the lung tissue model (lung-model) and the B cell population model (B cell-model). Genes where the lung-model predicts a too-high value are plotted in orange. **D-E)** True expression of the B cells vs. predicted expression by the **D)** lung-model and **E)** B cell-model. **F)** True expression of the lung cells vs. predicted expression of the lung model.

of B cells in the limb muscle, the models trained on B cells in the marrow and lung even outperform the model trained on B cells in the limb muscle itself (Figure 3C). But, the models trained on different cell populations within the limb muscle perform variably when predicting B cells (Figure 3D). The models trained on immune populations, e.g. T cells or macrophages, perform similarly, but the muscle-specific populations perform worse. This difference

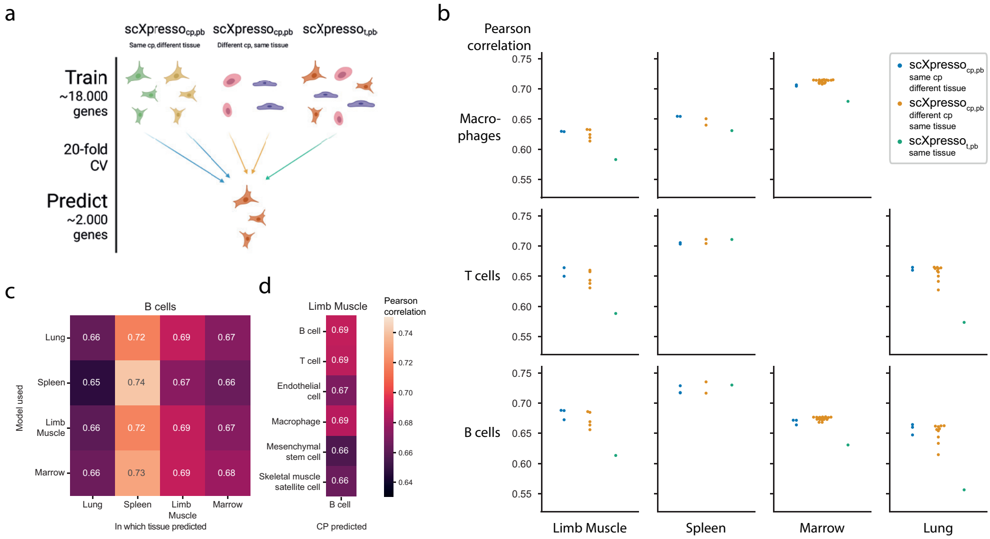


Figure 3. Comparing the predictions of scXpresso across cell populations and tissues. A) Schematic overview of the experiment. **B)** Performance (Pearson correlation) of three different types of models on different cell populations (rows) in different tissues (columns). Every dot is the median correlation of one model across the 20 folds. Since there are no T cells and macrophages defined in the Marrow and Lung dataset, these boxes are missing. **C)** Pearson correlation of different models when predicting the expression of B cells in different tissues. The rows indicate on which tissue scXpresso$_{cp,pb}$ is trained, and the columns indicate for which tissue the expression of the B cells is predicted. **D)** Pearson correlation of different scXpresso$_{cp,pb}$ when predicting the expression of B cells in the limb muscle. Again the rows indicate which model is used.

between the B cell and the endothelial, mesenchymal stem cell, and skeletal muscle satellite cell models might seem small but is significant across the 20 folds (p-value = 9.5e-07 for all three populations, one-sided Wilcoxon rank sum test [27,28]). Even though the differences are small, this indicates that our models indeed learn cell population-specific features.

6.3.3 scXpresso learns expression patterns across human brain cell populations

Next, we applied scXpresso to a human brain dataset of the motor cortex [5]. This dataset is annotated at different resolutions including a class (GABAergic, glutamatergic, and non-neuronal) and subclass (20 subclasses) level. Again, we trained models of different resolutions: a tissue- (t), class- (c), and subclass-specific (sc) model (scXpresso$_t$, scXpresso$_c$, and scXpresso$_{sc}$ respectively). We used the trained models to predict the subclass-specific expression values (Figure 4A). Since scXpresso$_t$ was trained on the tissue-specific pseudobulk expression, it predicts the same expression for all subclasses. The class-specific model, on the contrary, is a multitask model. Here, we use the predictions of the parent class to predict the expression of each subclass (i.e. subclasses belonging to the same parent class are predicted to have the same expression) (Figure S16). Similar to the Tabula Muris, we observed that increasing the resolution increases the performance: scXpresso$_{sc}$ outperforms scXpresso$_c$ which outperforms scXpresso$_t$, (Figure 4B). For some subclasses, e.g. L2/3 IT, the performance barely improves when comparing scXpresso$_{sc}$ with scXpresso$_{c}$, which happens when the true expression values of the subclass and corresponding class are strongly correlated, similar as for the Tabula Muris case (Figure S17).

Since genes with variable expression across subclasses are often interesting to study, we tested whether scXpresso$_{sc}$ can learn the correct pattern for a gene across the subclasses. For every gene, we calculate the Pearson correlation between the true and predicted expression across the subclasses. If the expression of a gene shows some variance across the subclasses, scXpresso$_{sc}$ predicts the pattern correctly (Figure 4C). An example is *CACNA1I*, a gene coding for a subtype of voltage-gated calcium channel that has been associated with schizophrenia [15,29–32]. Here scXpresso$_{sc}$ correctly learns that the expression in neuronal populations is higher than in non-neuronal (r = 0.90) (Figure 4D).

6.3.4 *In-silico* saturation mutagenesis reveals the most interesting GWAS variants

Since scXpresso can predict expression from the DNA sequence, we expect that it can also predict how the expression changes when the sequence is mutated. Therefore, we applied *in-silico* mutagenesis (ISM) to the sequence of *CACNA1I* and evaluated the predicted change in gene expression [6,7,11,33]. When comparing scXpresso$_{sc}$ predictions for the Sst Chodl subclass across all possible mutations, we find mutations in the region around the TSS to affect the expression of the *CACNA1l* gene the most (Figure 4E). When applying ISM to the 2000 highly variable genes in the data, the maximum absolute predicted effect is highest around the TSS as well (Figure S18). Note, that we did not use the TSS location as input to the model, consequently, the model correctly identified that this is the most important region for transcriptional regulation. No other regions within our input window were found that affect the expression that strongly.

Besides visualizing the mutation pattern for one subclass, we can also visualize how ISM affects two subclasses differently. As an example, we compared the scXpresso$_{sc}$ predictions for the Sst Chodl subclass and the L2/3 IT subclass (Figure S19). These predictions show that the Sst Chodl subclass is more sensitive to mutations than the L2/3 IT class for *CACNA1I*, which might be explained by the fact that *CACNA1I* is also higher expressed in Sst Chodl cells.

In addition, ISM can be used to prioritize variants of interest for diseases. *CACNA1I* is linked to 18 Schizophrenia-associated variants according to the NHGRI-EBI Catalog [34]. Two of these variants, rs7288455 and rs5757730, fall within our input region (7kb upstream and 3.5kb downstream of the *CACNA1l* TSS). Mutating the reference A allele with the C or G variant at the position of rs7288455 increases the predicted expression for all cell populations (Figure 4F). The disease-associated variant, the A allele, is expected to decrease the expression [15,34], which is in line with our predictions, although it is not known whether this is subclass-related. Our model suggests that the expression of *CACNA1I* increases the most in the Sst Chodl subclass. Interestingly, for the Sst Chodl subclass, this mutation results in one of the largest differences in *CACNA1l* expression amongst all other induced mutations (top 1% mutations with the strongest effect) (Figure S20). For the other variant, rs5757730, which

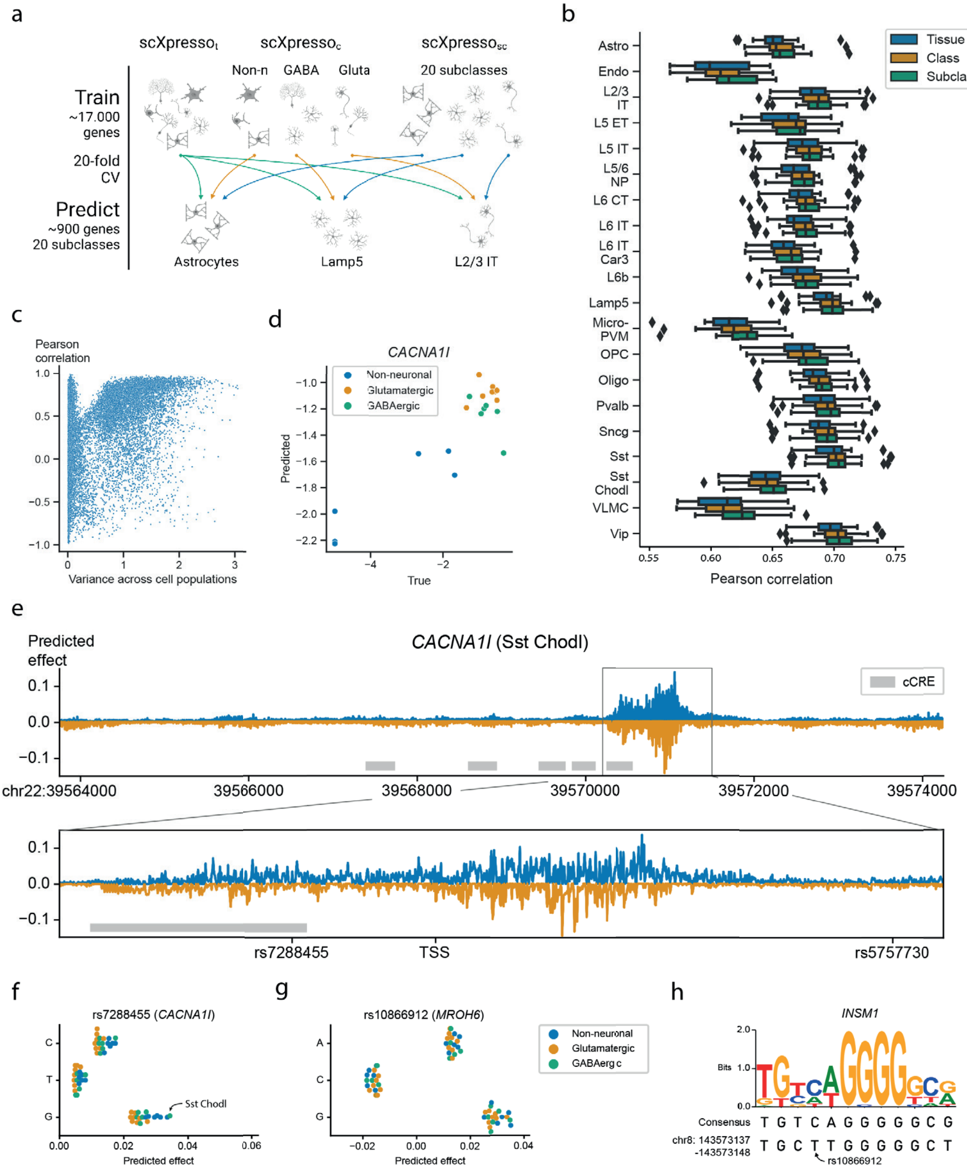


Figure 4. Performance of scXpresso on the human motor cortex. A) Schematic overview of the experiment. We train a tissue- (t), class- (c), and subclass-specific (sc) model (scXpresso$_t$, scXpresso$_c$, scXpresso$_{sc}$ respectively) to predict the subclass-specific expression levels. **B)** Boxplots showing the Pearson correlation between the true and predicted values. Every point in the boxplot is the performance on a fold (n=20). **C)** Scatterplot showing the relation between the variance of a gene across the pseudobulk values of the subclasses and the Pearson correlation between the true and predicted values across the subclasses. Every dot is a gene. **D)** True and predicted expression for *CACNA1I*. Every dot is the expression in a subclass. Dots are colored according to their class. **E)** Mutation profile for *CACNA1I* for the Sst Chodl subclass. For every position, we calculated the difference in expression for all three possible substitutions and visualized the substitution with the highest absolute predicted effect. Mutations that are predicted to increase or decrease the expression are plotted in blue and orange, respectively. The grey rectangle highlights the region around the TSS. The grey boxes indicate the positions of candidate cis-Regulatory Elements (cCREs) derived from

ENCODE data [25]. **F-G)** Predicted effect, the predicted difference between the reference and alternative allele, of the three substitutions for **F)** rs7288455 on *CACNA1I* expression, and **G)** rs10866912 on *MROH6* expression. Every dot is one subclass and the dots are colored according to the class. **H)** Sequence logo and the consensus sequence for the *INSM1* transcription factor motif together with the sequence of the reference genome (bottom line).

lies in an intronic region, we see no difference in expression (Figure S21). Further supporting our predictions, rs7288455, but not rs5757730, overlaps with an ENCODE candidate *cis*-regulatory element. These results show that scXpresso can be used to prioritize GWAS hits.

In total, there are 3,971 GWAS variants associated with Schizophrenia in the NHGRI-EBI Catalog [34]. We focused on those genes that have two or more variants in the input region (20 genes, 49 variants) (Table S5). For these variants, we predicted the effect of all possible substitutions to prioritize the likely causal variants (Figure S22). For most genes, scXpresso predicts a profound effect for only one of the variants. For instance, when substituting 'A' with 'C' for the *HLA-B* variant rs2507989, the predicted expression of *HLA-B* decreases, while none of the mutations at the other variant positions of HLA-B, i.e. rs139099016 and rs1131275, are predicted to affect the expression. Noteworthy, rs1131275 is classified as a missense variant and thus not expected to alter transcription [34]. For some genes, however, all variants seem to barely affect the expression.

Next, we checked if we could interpret the model predictions by characterizing the genomic sequences identified by scXpresso to have a strong effect on gene expression. For the *MROH-6* variant rs10866912, two substitutions are predicted to create an opposite effect. Substituting the reference 'T' with a 'C' is predicted to decrease the expression while mutating with a 'G' is predicted to increase the expression (Figure 4G). This variant is part of a binding site for the transcription factor *INSM1*, a transcriptional repressor [35] (Figure 4H). When substituting the 'T' with a 'C', the sequence of the reference genome becomes more similar to the consensus motif, while substituting with a 'G' makes the two sequences more dissimilar. This supports the predictions from scXpresso.

We compare our scXpresso predictions for these Schizophrenia variants to the predictions of Enformer, ExPecto, and Xpresso. For Enformer and ExPecto we used their pre-trained models which predict the expression for 5,313 and 218 tissues/cell lines, respectively. Here, we only focused on the predictions related to the healthy brain (77 tracks for Enformer, 27 for ExPecto). For Xpresso, there were no pre-trained models for the brain available, so we trained the Xpresso model ourselves using bulk RNA-seq samples from the precentral gyrus, which is the region containing the motor cortex (see Methods). The expression values of the precentral gyrus are correlated to the pseudobulk expression values of the motor cortex (Figure S23A, r = 0.68). Similar to scXpresso, we used a 20-fold cross-validation to train the Xpresso model. The model is well-trained and reached a similar median correlation on the precentral gyrus as the scXpresso models on the motor cortex subclasses (Figure 4B, S23B-C, r = 0.69). Figure S24 shows the predictions for all models for the variants related to Schizophrenia. Using Xpresso and ExPecto we could not predict the effect of all variants, since some genes were missing from the data and some variants were lost during conversion from Hg38 to Hg19 (Table S5) (see Methods). It's challenging to compare the predictions of the different methods since all models are trained on different brain regions or cell lines. Enformer usually predicts the same effect for the three different possible nucleotide mutations, e.g. for rs1131275 it predicts that all three substitutions decrease the expression. This variant, however, is classified as a missense variant, so we don't expect it to alter transcription [34]. For rs7288455, the variant close to *CACNA1I*, both scXpresso and Xpresso predict a similar effect, while Enformer and ExPecto predict only a very minimal effect. For rs10866912, the variant close to *MROH-6*, we showed that scXpresso could learn the TF binding site of *INSM1* while all the other models miss this pattern. These results overall illustrate the benefit of training prediction models on single-cell data.

6.4 Discussion

We presented scXpresso, a model to predict cell population-specific gene expression using the genomic sequence. We showed that scXpresso outperforms tissue-specific bulk and pseudobulk models especially when the expression profile of a cell population is dissimilar to that of the corresponding tissue. All scXpresso models reach a Pearson correlation of approximately 0.7 regardless of the cell population or tissue trained on. Additionally, the model learned the importance of the region around the TSS, transcription factor binding motifs (such as for *INSM1*), and the expression pattern of genes across different cell populations. Together, our findings show the potential of using single-cell data for predicting gene expression from sequence information in complex heterogeneous tissues.

We showed that it is possible to prioritize GWAS variants using scXpresso. Considering the expression of *CACNA1I*, we noticed that one variant, which overlaps with an ENCODE *cis*-regulatory element, is predicted to have a large effect, while another variant was predicted to have a negligible effect. The latter could be because the variant might affect splicing (which our model does not differentiate), the variant could be in a linkage disequilibrium block with other (associating) variants, or the variant could affect a more distant gene.

Comparing the predicted effects for mutations by scXpresso to other sequence-to-expression prediction models quantitatively is difficult as the true effect of these variants on specific brain regions and/or cell populations is unknown. We have shown that for a previously identified variant close to *CACNA1I* gene, both Xpresso and scXpresso predict an increase in expression, while ExPecto and Enformer predict a marginal effect. Note that, ExPecto and Enformer are not trained on specific brain regions, or cell population-specific data, but contain bigger structures such as the frontal cortex or frontal lobe. Hence, these models miss the cell population-specific effect of this variant. Training these models on cell population-specific scRNA data could be an interesting next step.

Using our model, it is not possible to test trans-effects of variants as our model uses a limited genomic sequence region as input. Consequently, we could only test two variants related to Schizophrenia for *CACNA1I,* out of the 18 variants associated with *CACNA1I [34]*. Ideally, we would increase the length of the input sequence, however, it is not easy to learn long-range interactions using CNNs. The Enformer model, which uses a 200kb sequence as input, tackles this problem by combining transformers and CNNs [11]. Unfortunately, the Enformer model predicts CAGE reads instead of expression values, so we cannot trivially extend it or use it for single-cell data. An alternative approach might be to use their well-trained model to get an embedding for every input sequence and use this embedding to predict cell population-specific expressions.

We input the DNA sequence and five half-life time features to scXpresso. However, certain transcript features, which are related to the half-life time features, can predict zeros in the scRNA-seq data [36]. Whether the observed zeros in scRNA-seq data are technical artifacts or biologically informative is an ongoing debate. We believe that the zeros are biologically informative since binarized data can be used for downstream analysis, resulting in comparable results to those obtained using scRNA-seq counts [37]. Furthermore, we would like to highlight that the performance of the cell population-specific pseudobulk models when trained on sequence-only information is also not much lower as compared to both sequence and half-life time features (Figure S9). This observation supports our conclusion that the half-life time features are not biasing the models towards scRNA-seq artifacts.

Two future enhancements that we envision that could improve the performance of our model are related to the half-life time features and the output of the model. Currently, we extract five features from the mRNA sequence to approximate the half-life time. Recently, a new model, Saluki, was developed that could predict mRNA degradation rates directly from the sequence of the gene [38]. Replacing the currently used features with those predicted by the Saluki model, or combining these features, might improve the cell population-specific predictions. A second potential improvement relates to the current output of scXpresso, which is the pseudobulk expression for every cell population, i.e. the average gene expression across all cells from that population. However, this ignores the variance within the population. It might be more beneficial to predict the distribution of gene expression across each population, instead of just one aggregated value.

In summary, we have shown the potential of predicting cell population-specific gene expression from genomic sequences by leveraging the resolution of single-cell data, opening the way for many new developments in this area.

6.5 Code and data availability

The pseudobulk expression values, trained models, and predictions are available on Zenodo: https://doi.org/10.5281/zenodo.7044908.

The code to reproduce the figures, train your own models, show the effect of variants, and do *in-silico* saturation mutagenesis can be found on GitHub: https://github.com/lcmmichielsen/scXpresso.

Bibliography

1. Lambert SA, Jolma A, Campitelli LF, Das PK, Yin Y, Albu M, et al. The Human Transcription Factors. Cell. 2018;172: 650–665. doi:10.1016/j.cell.2018.01.029
2. Vaquerizas JM, Kummerfeld SK, Teichmann SA, Luscombe NM. A census of human transcription factors: function, expression and evolution. Nat Rev Genet. 2009;10: 252–263. doi:10.1038/nrg2538
3. Nott A, Holtman IR, Coufal NG, Schlachetzki JCM, Yu M, Hu R, et al. Brain cell type-specific enhancer-promoter interactome maps and disease-risk association. Science. 2019;366: 1134–1139. doi:10.1126/science.aay0793
4. Janssens J, Aibar S, Taskiran II, Ismail JN, Gomez AE, Aughey G, et al. Decoding gene regulation in the fly brain. Nature. 2022;601: 630–636. doi:10.1038/s41586-021-04262-z
5. Bakken TE, Jorstad NL, Hu Q, Lake BB, Tian W, Kalmbach BE, et al. Comparative cellular analysis of motor cortex in human, marmoset and mouse. Nature. 2021;598: 111–119. doi:10.1038/s41586-021-03465-8
6. Kelley DR, Snoek J, Rinn JL. Basset: learning the regulatory code of the accessible genome with deep convolutional neural networks. Genome Res. 2016;26: 990–999. doi:10.1101/gr.200535.115
7. Kelley DR, Reshef YA, Bileschi M, Belanger D, McLean CY, Snoek J. Sequential regulatory activity prediction across chromosomes with convolutional neural networks. Genome Res. 2018;28: 739–750. doi:10.1101/gr.227819.117
8. Zhou J, Theesfeld CL, Yao K, Chen KM, Wong AK, Troyanskaya OG. Deep learning sequence-based ab initio prediction of variant effects on expression and disease risk. Nat Genet. 2018;50: 1171–1179. doi:10.1038/s41588-018-0160-6
9. Agarwal V, Shendure J. Predicting mRNA Abundance Directly from Genomic Sequence Using Deep Convolutional Neural Networks. Cell Rep. 2020;31: 107663. doi:10.1016/j.celrep.2020.107663
10. Zhang Y, Zhou X, Cai X. Predicting Gene Expression from DNA Sequence using Residual Neural Network. bioRxiv. 2020; 2020.06.21.163956. doi:10.1101/2020.06.21.163956
11. Avsec Ž, Agarwal V, Visentin D, Ledsam JR, Grabska-Barwinska A, Taylor KR, et al. Effective gene expression prediction from sequence by integrating long-range interactions. Nat Methods. 2021;18: 1196–1203. doi:10.1038/s41592-021-01252-x
12. Avsec Ž, Weilert M, Shrikumar A, Krueger S, Alexandari A, Dalal K, et al. Base-resolution models of transcription-factor binding reveal soft motif syntax. Nat Genet. 2021;53: 354–366. doi:10.1038/s41588-021-00782-6
13. Wesolowska-Andersen A, Zhuo Yu G, Nylander V, Abaitua F, Thurner M, Torres JM, et al. Deep learning models predict regulatory variants in pancreatic islets and refine type 2 diabetes association signals. Elife. 2020;9. doi:10.7554/eLife.51503
14. Wightman DP, Jansen IE, Savage JE, Shadrin AA, Bahrami S, Holland D, et al. A genome-wide association study with 1,126,563 individuals identifies new risk loci for Alzheimer’s disease. Nat Genet. 2021;53: 1276–1282. doi:10.1038/s41588-021-00921-z
15. Yao X, Glessner JT, Li J, Qi X, Hou X, Zhu C, et al. Integrative analysis of genome-wide association studies identifies novel loci associated with neuropsychiatric disorders. Transl Psychiatry. 2021;11: 69. doi:10.1038/s41398-020-01195-5
16. Schizophrenia Working Group of the Psychiatric Genomics Consortium. Biological insights from 108 schizophrenia-associated genetic loci. Nature. 2014;511: 421–427. doi:10.1038/nature13595
17. Tasic B, Menon V, Nguyen TN, Kim TK, Jarsky T, Yao Z, et al. Adult mouse cortical cell taxonomy revealed by single cell transcriptomics. Nat Neurosci. 2016;19: 335–346. doi:10.1038/nn.4216
18. Tasic B, Yao Z, Graybuck LT, Smith KA, Nguyen TN, Bertagnolli D, et al. Shared and distinct transcriptomic cell types across neocortical areas. Nature. 2018;563: 72–78. doi:10.1038/s41586-018-0654-5
19. Sharova LV, Sharov AA, Nedorezov T, Piao Y, Shaik N, Ko MSH. Database for mRNA Half-Life of 19 977 Genes Obtained by DNA Microarray Analysis of Pluripotent and Differentiating Mouse Embryonic Stem Cells. DNA Res. 2009;16: 45–58. doi:10.1093/DNARES/DSN030
20. Spies N, Burge CB, Bartel DP. 3′ UTR-isoform choice has limited influence on the stability and translational efficiency of most mRNAs in mouse fibroblasts. Genome Res. 2013;23: 2078–2090. doi:10.1101/GR.156919.113
21. Paszke A, Gross S, Massa F, Lerer A, Bradbury J, Chanan G, et al. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In: Wallach H, Larochelle H, Beygelzimer A, d\textquotesingle Alché-Buc F, Fox E, Garnett R, editors. Advances in Neural Information Processing Systems 32. Curran Associates, Inc.; 2019. pp. 8024–8035. Available: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf
22. Schaum N, Karkanias J, Neff NF, May AP, Quake SR, Wyss-Coray T, et al. Single-cell transcriptomics of 20 mouse organs creates a Tabula Muris. Nature. 2018;562: 367–372. doi:10.1038/s41586-018-0590-4
23. Schaum N, Lehallier B, Hahn O, Hosseinzadeh S, Lee SE, Sit R, et al. The murine transcriptome reveals global aging nodes with organ-specific phase and amplitude. bioRxiv. 2019. p. 662254. doi:10.1101/662254
24. Shen W, Le S, Li Y, Hu F. SeqKit: A Cross-Platform and Ultrafast Toolkit for FASTA/Q File Manipulation. PLoS One. 2016;11: e0163962. doi:10.1371/journal.pone.0163962
25. ENCODE Project Consortium, Moore JE, Purcaro MJ, Pratt HE, Epstein CB, Shoresh N, et al. Expanded encyclopaedias of DNA elements in the human and mouse genomes. Nature. 2020;583: 699–710. doi:10.1038/s41586-020-2493-4
26. Wolf FA, Angerer P, Theis FJ. SCANPY: Large-scale single-cell gene expression data analysis. Genome Biol. 2018;19: 15. doi:10.1186/s13059-017-1382-0

27. Mann HB, Whitney DR. On a Test of Whether one of Two Random Variables is Stochastically Larger than the Other. aoms. 1947;18: 50–60. doi:10.1214/aoms/1177730491

28. Demšar J. Statistical comparisons of classifiers over multiple data sets. J Mach Learn Res. 2006;7: 1–30. Available: https://psycnet.apa.org/fulltext/2007-03550-001.pdf

29. Ikeda M, Takahashi A, Kamatani Y, Momozawa Y, Saito T, Kondo K, et al. Genome-Wide Association Study Detected Novel Susceptibility Genes for Schizophrenia and Shared Trans-Populations/Diseases Genetic Effect. Schizophr Bull. 2019;45: 824–834. doi:10.1093/schbul/sby140

30. Li Z, Chen J, Yu H, He L, Xu Y, Zhang D, et al. Genome-wide association analysis identifies 30 new susceptibility loci for schizophrenia. Nat Genet. 2017;49: 1576–1583. doi:10.1038/ng.3973

31. Lam M, Chen C-Y, Li Z, Martin AR, Bryois J, Ma X, et al. Comparative genetic architectures of schizophrenia in East Asian and European populations. Nat Genet. 2019;51: 1670–1678. doi:10.1038/s41588-019-0512-x

32. Pardiñas AF, Holmans P, Pocklington AJ, Escott-Price V, Ripke S, Carrera N, et al. Common schizophrenia alleles are enriched in mutation-intolerant genes and in regions under strong background selection. Nat Genet. 2018;50: 381–389. doi:10.1038/s41588-018-0059-2

33. Zhou J, Troyanskaya OG. Predicting effects of noncoding variants with deep learning-based sequence model. Nat Methods. 2015;12: 931–934. doi:10.1038/nmeth.3547

34. Buniello A, MacArthur JAL, Cerezo M, Harris LW, Hayhurst J, Malangone C, et al. The NHGRI-EBI GWAS Catalog of published genome-wide association studies, targeted arrays and summary statistics 2019. Nucleic Acids Res. 2019;47: D1005–D1012. doi:10.1093/nar/gky1120

35. Castro-Mondragon JA, Riudavets-Puig R, Rauluseviciute I, Lemma RB, Turchi L, Blanc-Mathieu R, et al. JASPAR 2022: the 9th release of the open-access database of transcription factor binding profiles. Nucleic Acids Res. 2022;50: D165–D173. doi:10.1093/nar/gkab1113

36. Lipnitskaya S, Shen Y, Legewie S, Klein H, Becker K. Machine learning-assisted identification of factors contributing to the technical variability between bulk and single-cell RNA-seq experiments. bioRxiv. 2022. p. 2022.01.06.474932. doi:10.1101/2022.01.06.474932

37. Bouland GA, Mahfouz A, Reinders MJT. Consequences and opportunities arising due to sparser single-cell RNA-seq datasets. Genome Biol. 2023;24: 86. doi:10.1186/s13059-023-02933-w

38. Agarwal V, Kelley DR. The genetic and biochemical determinants of mRNA degradation rates in mammals. Genome Biol. 2022;23: 245. doi:10.1186/s13059-022-02811-x

chapter 7

Predicting cell-type-specific exon inclusion in the human brain reveals more complex splicing mechanisms in neurons than glia

Lieke Michielsen, Justine Hsu, Anoushka Joglekar, Natan Belchikov, Marcel J.T. Reinders, Hagen Tilgner*, Ahmed Mahfouz*

This chapter is submitted to *Nature Communications*
Supplementary material is available online at:
https://www.biorxiv.org/content/10.1101/2024.03.18.585465v1.supplementary-material
*Equal contribution

Alternative splicing contributes to molecular diversity across brain cell types. RNA-binding proteins (RBPs) regulate splicing, but the genome-wide mechanisms remain poorly understood. Here, we used RBP binding sites and/or the genomic sequence to predict exon inclusion in neurons and glia as measured by long-read single-cell data in human hippocampus and frontal cortex. We found that alternative splicing is harder to predict in neurons compared to glia in both brain regions. Comparing neurons and glia, the position of RBP binding sites in alternatively spliced exons in neurons differ more from non-variable exons indicating distinct splicing mechanisms. Model interpretation pinpointed RBPs, including QKI, potentially regulating alternative splicing between neurons and glia. Finally, using our models, we accurately predict and prioritize the effect of splicing QTLs. Taken together, our models provide new insights into the mechanisms regulating cell-type-specific alternative splicing and can accurately predict the effect of genetic variants on splicing.

7.1 Introduction

During RNA splicing, introns are removed from the precursor mRNA. Different combinations of exons result in different mRNA isoforms, which may differ in function [1–3]. This mechanism, called alternative splicing, causes most of the complexity of human tissues and cell types; approximately 95% of all human genes are believed to be spliced in multiple ways [4,5]. Across different tissues, the brain has the highest levels of exon skipping and one of the most distinctive patterns of alternative splicing [6].

Alternative splicing (AS) is partly regulated by RNA-binding proteins (RBPs) [7,8], which can activate or inhibit spliceosome assembly or splice site recognition. RBFOX proteins, for instance, instruct neuronal differentiation by regulating splicing of *NIN* which in turn affects the localization of the corresponding Ninein protein [9,10]. Additionally, splicing regulation often relies on the combinatorial binding of multiple RBPs. For example, the inclusion of exon 9 of *Gabrg2* is dependent on the binding of RBFOX and NOVA [11]. Splicing simulators have taken into account splicing enhancers and silencers [12] and a splicing code for tissue-dependent splicing has been elaborated [13–15]. However, the genome-wide mechanisms regulating splicing across different cell types remain largely unknown.

Long-read sequencing is an emerging technology that has made important contributions to RNA biology since its inception [16–20]. Long-read single-cell and single-nuclei sequencing in fresh [21,22] and frozen [23] tissue allows the study of alternative splicing at the cell-type level in the brain and other complex tissues. Such analyses revealed that most mouse genes show differential isoform expression across at least one pair of cell types, regions, and/or developmental time points in the brain [24,25]. In accordance with prior studies [26–28], single-nuclei isoform RNA sequencing (SnISOr-Seq) of the human adult frontal cortex revealed that exons associated with autism spectrum disorder (ASD) are variably included across cell types [23].

To understand (alternative) splicing mechanisms and the influence of RBPs, several computational methods have been developed. AVISPA, for instance, predicts alternative splicing in four tissues by extracting regulatory features, such as the length of the exon or the presence of RBP binding sites, from the mRNA sequence [14]. Other methods, including SpliceAI, DNABERT, Pangolin, and MTSplice, directly use the pre-mRNA sequence as input to their models [29–32]. However, none of the current methods predict cell-type-specific alternative splicing in a genome-wide manner, which is crucial for understanding splicing in heterogeneous tissues such as the brain.

Here, we present two methods to predict cell-type-specific exon inclusion using the pre-mRNA sequence and/or the presence of RBP binding sites in the hippocampus and frontal cortex. After training our machine learning models, we used model interpretation to study the mechanisms governing cell-type-specific exon inclusion. We focused on variable exons which we defined as exons for which the inclusion rates differ in neurons and glia. We found that the presence of RBP binding sites in variable exons compared to non-variable exons differs more in neurons than in glia. This indicates that the alternative splicing mechanism in neurons deviates more from the non-variable mechanism. Furthermore, we show that some RBPs, including QKI, have a big effect on exon inclusion in glia, that the regions close to the splice sites are most important for predicting exon inclusion, and that we can correctly predict and prioritize the effect of splicing QTLs and prioritize their effects.

7.2 Results

7.2.1 Predicting exon inclusion is more difficult in neurons than in glia

To define the rules governing exon inclusion in distinct cell types, we trained different models to predict cell-type-specific percent spliced-in (Ψ) values in the brain (Figure 1A). We focused on neurons and glia in two human brain regions, hippocampus (HPC) and frontal cortex (FC), and calculated Ψ values per exon by aggregating single-nuclei isoform RNA sequencing (SnISOr-Seq) reads from multiple individuals (Table 1, Methods) [23,25]. Most exons are either almost always included ($\Psi \approx 1$) or excluded ($\Psi \approx 0$) in an mRNA molecule (Figure 1B, S1A-C). Furthermore, most exons have similar values in neurons and glia (Figure 1C, S1D). We define exons with different inclusion rates in neurons and glia ($|\Delta\Psi_{glia-neur}| > 0.25$) as variable exons. In HPC and FC, 2,244 and 943 exons are variable respectively (Table 1). In contrast to non-variable exons, these values show a uniform distribution of Ψ (Figure 1B). Even though we used a minimum of 10 reads per exon to calculate a Ψ value (Methods), we believe these values are reliable. When comparing the Ψ values of the variable exons per individual in neurons and glia, there is a clear separation between neurons and glia (Figure S2). Since most exons are almost always included, we downsampled these exons when training the models (Methods).

First, we used a logistic regression (LR) model to predict Ψ values from RBP binding sites of 122 RBPs from the ENCODE project [8]. These RBPs were measured in two cell lines (K562, HepG2), implying that this data is not brain cell-type-specific. We generated a count matrix, indicating the number of binding sites per exon for each RBP. Since the position of an RBP can influence its function [33,34], we split these binding sites based on six possible binding

Table 1. The number of measured exons (exons for which at least 10 reads were sequenced in both the neurons and glia) and variable exons ($|\Delta\Psi_{glia-neur}| > 0.25$) in the hippocampus (HPC) and frontal cortex (FC).

	Individuals	Measured exons	Variable exons
HPC [25]	6	68,215	2,244
FC [23]	2	56,427	943

locations: 1) upstream of the exon (up to 400bp), 2) overlapping the 3’ splice site, 3) in the exon, 4) spanning the exon, 5) overlapping the 5’ splice site, and 6) downstream of the exon (up to 400bp) (Figure 1A).

Any model is strongly influenced by its training data. A model trained on all exons might be dominated by the rules governing non-variable exons, while cell-type-specific inclusion effects might be overlooked. Therefore, we trained three different models using 10-fold cross-validation and either: A) all exons (LR_{all}), B) exons with $|\Delta\Psi_{glia-neur}| > 0.1$ ($LR_{var0.1}$), or C) exons with $|\Delta\Psi_{glia-neur}| > 0.25$ ($LR_{var0.25}$) as training data (Table S1). When evaluating the models on all exons, LR_{all} showed the highest median Spearman correlation between true and predicted Ψ values on all four datasets followed by $LR_{var0.1}$ and $LR_{var0.25}$ (Figure 1D, S3). On hippocampal variable exons, however, $LR_{var0.1}$ outperformed the other models (Figure 1D). The performance increase when training on variable exons indicates that the splicing mechanism in these variable exons is somewhat different from the mechanism in non-variable exons. In the frontal cortex, the performance on neurons increased when the training data became more specific, while the performance on glia decreased (Figure S3). Surprisingly, we predicted Ψ values more accurately in glia than neurons in both brain regions (median Spearman correlation of 0.54 vs. 0.23 in HPC, and 0.57 vs. 0.10 in FC) (Figure 1D-F, S3-4). Furthermore, using $LR_{var0.25}$ to predict Ψ values of all exons resulted in lower performance for neurons compared to glia in both HPC and FC (Figure 1D, S3). Indicating that the learned splicing patterns for variable exons in neurons do not generalize to non-variable exons- likely because the underlying molecular grammar is different in the two exon sets.

7.2.2 Primary sequence is more informative for neurons

The RBP binding sites used to train the logistic regression models were measured in immune and liver cancer cell lines and are thus not cell-type specific - and may reflect glial more than neuronal splicing as shown above. Furthermore, some RBPs known to be important for splicing in the brain, such as NOVA1 and NOVA2, are not included in the ENCODE data [35,36]. To test whether this caused the low performance of the models on neurons, we trained sequence-based models - which are independent of any RBP data and comparable across different cell types. We adapted the Saluki model, a hybrid convolutional and recurrent neural network that uses mRNA sequences to predict mRNA degradation rates [37], to predict Ψ values (Methods) (Figure 1A, S5). The input sequence is 6,144 bp with the exon of interest centered in the middle. Since deep learning models need large training datasets, we trained a model using all exons ($DL_{all\text{-}seq}$) and a model using exons with $|\Delta\Psi_{glia-neur}| > 0.1$ ($DL_{var0.1\text{-}seq}$).

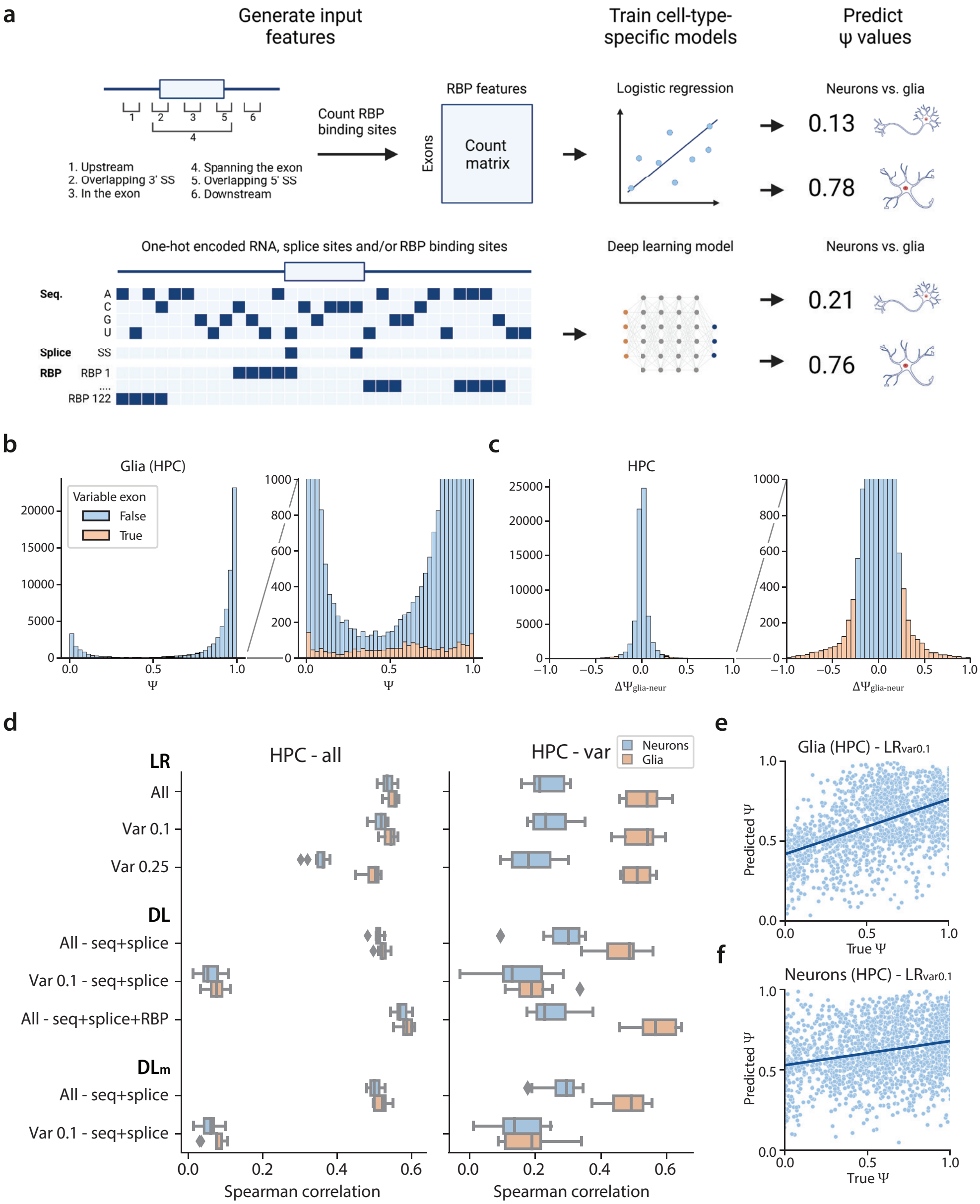


Figure 1. Overview and performance of the Ψ prediction models. A) Schematic overview of the models used to predict cell-type-specific Ψ values. **B)** Distribution of Ψ values of glia in the hippocampus. **C)** Distribution of $\Delta\Psi_{glia\text{-}neur}$ for the hippocampus. **D)** Performance of the different models during 10-fold cross-validation on all exons and the variable exons in glia and neurons in the hippocampus. **E-F)** Scatterplot showing the predictions of $LR_{var0.1}$ for variable exons in glia and neurons.

In HPC, the LR_{all} model outperformed the DL models when evaluating performance on all exons, but on variable exons, $DL_{all\text{-}seq}$ outperformed $LR_{var0.1}$ for neurons (Figure 1D). For the variable exons in neurons, primary sequence is more informative than the measured ENCODE-derived RBP-binding-site data. Even though the performance increases for neurons, the performance gap between neurons and glia remains. Thus, neuronal splicing patterns probably have more complex regulation mechanisms that we do not capture with the current models. In FC, the performance of the DL models on all exons and variable exons was considerably lower compared to HPC (Figure 1D, S6). This is likely related to the size of the training data which is significantly smaller for FC than HPC (Table S1).

Next, we combined sequence and RBP binding sites by adding a channel for every RBP which indicates the presence of a binding site ($DL_{all\text{-}seq\text{-}RBP}$) (Figure 1A, S5). This outperformed the LR models and resulted in the best-performing model for glia (median Spearman correlation of 0.54 vs. 0.57 in HPC, and 0.57 vs. 0.65 in FC) (Figure 1D, S3, S6). This improvement indicates that we can capture regulatory information from sequence beyond those present in RBP data alone. For neurons, however, $DL_{all\text{-}seq\text{-}RBP}$ had lower performance than $DL_{all\text{-}seq}$, again confirming that the ENCODE RBP data is more informative for glia than neurons.

We also trained DL models that do not use splice sites or only use RBPs as input for the neurons and glia in HPC to understand how the input channels affect performance (Figure S7). Omitting splice sites only slightly decreased the performance, which indicates that the model can recognize the splice sites quite easily from the sequence itself. For glia, using the RBPs as the only input feature results in a comparable performance to the LR_{all} model (median Spearman correlation of 0.55 vs. 0.54) and an even better performance than sequence and splice sites only (median Spearman correlation of 0.49). However, for neurons, we observe the opposite; using RBP binding sites reduces performance compared to the $DL_{all\text{-}seq}$ model (median Spearman correlation of 0.23 vs. 0.30).

7.2.3 Exon inclusion mechanisms are conserved between human and mouse

As cell-type-specific alternative splicing is partially conserved between humans and mice [25], we hypothesized that adding mouse data to our model would increase performance. We combined human HPC data with mouse HPC [25]. Since mouse FC data is not available, we combined human FC with data from the mouse visual cortex (VIS). While these two cortical regions are not identical, they do share many common characteristics. Especially in mouse HPC, few exons are variable (528) compared to VIS (1,404) (Table S2, Figure S8). Although $DL_{all\text{-}seq\text{-}RBP}$ performed best in glia, we only trained models with sequence and splice sites as input channels ($DL_{all\text{-}seq\text{-}m}$, $DL_{var01\text{-}seq\text{-}m}$) since RBP binding sites were not measured in mouse cell lines. In HPC, the performance on variable exons of both cell types slightly increased by adding the mouse data (Figure 1D). On FC, the performance on all exons increased as well (Figure S6), supporting our hypothesis that not enough training data was available to train these models on human exons alone. Similar to the human data, glial Ψ values were easier to predict than neuronal ones in mice (Figure S9).

7.2.4 The splicing mechanisms in neurons diverged more than in glia

Our above results show that neuronal Ψ values are harder to predict than glial regardless of the model or input data. Hence, splicing mechanisms in neurons might be different than in glia and more complex. However, Ψ values could be biased, making it easier to predict in glia. To exclude the latter, we used the hippocampus data to assess whether glia and neurons are similar in terms of 1) Ψ-value distributions, 2) heterogeneity within each cell type, and 3) variation across individuals.

First, comparing Ψ distributions, more values are close to 0 or 1 in glia than neurons (Figure S10AB), which is most apparent for the non-variable exons (two-sided Kolmogorov-Smirnov test, p-value < 2.2e-16). For variable exons (Figure S10B), however, both distributions are not different (two-sided Kolmogorov-Smirnov test, p-value = 0.44). Thus, data distribution differences cannot explain all observed differences between neurons and glia.

Second, to quantify the heterogeneity within a cell type, we measured the difference in Ψ values between finer cell-type classifications. For neurons, we compared the inhibitory and excitatory neurons, and for glia, we compared oligodendrocytes and astrocytes. Within glia, we have more variable exons ($|\Delta\Psi| > 0.25$) compared to neurons (831 vs. 745). In neurons, more exons have an extreme difference ($|\Delta\Psi| > 0.5$) (92 vs. 70) (Figure S10CD). Compared to the total exon number defined for both cell types in neurons and glia (28,296 and 27,047 respectively), both numbers are small. Thus, this cannot explain the difference in performance between neurons and glia.

Third, to compare the variance across individuals for glia and, separately, for neurons, we calculated Ψ values per individual instead of using the aggregated counts. We calculated the variance for an exon only if ≥3 individuals have ≥10 reads for that exon in both neurons and glia. For both non-variable and variable exons, the variance is higher in glia (two-sided paired Wilcoxon signed-rank test, p-value = 1.3e-28 and 8.9e-5 respectively) (Figure S10E). Thus, the data do not explain observed differences in performance between neurons and glia.

We then hypothesized that splicing mechanisms regulating variable exons in neurons might differ from the non-variable exons. To test this hypothesis, we compared the RBP binding profiles between variable and non-variable exons in neurons and glia (Figure 2A). We performed these comparisons for exons with a high (≥ 0.5) and a low Ψ value (< 0.5) separately. The binding profiles between variable and non-variable exons differ significantly more in neurons compared to glia in HPC (Figure 2B) and FC (Figure 2C). Non-variable exons with high Ψ values more often have a binding site at the 3' splice site for splicing factors such as U2AF1, U2AF2, and SF3B4 compared to non-variable exons with low Ψ values (Figure 2D, S11AB). In glia, variable exons show a similar pattern (Figure 2E, S11AB). However, binding sites for these splicing factors cannot differentiate between exons with high and low Ψ values in neurons (Figure 2F, S11AB), indicating that these RBP binding sites are likely not used in neurons. In the hippocampus, PTBP1 differs the most between neurons and glia (Figure S11C). PTBP1 is a position-dependent RBP: binding within or upstream of an exon represses splicing while binding downstream activates splicing in HeLa cells [38]. Our RBP binding profiles contradict these known mechanisms. In HeLa cells, however, PTBP1 is highly and PTBP2 is lowly expressed, while this is vice versa in the hippocampus (Figure S12). PTBP1 RBP binding profiles obtained from non-brain cell lines are thus less likely to reflect splicing mechanisms in the hippocampus. Strikingly, the binding profile of PTBP1 in variable exons in neurons is again considerably different from the variable exons in glia and the non-variable exons. There is no position-dependent regulation and no difference between exons with a high and low Ψ value. In the hippocampus, only one RBP, HNRNPC, showed the opposite pattern with larger differences in glia compared to neurons (Figure S11D).

7.2.5 Interpretation of LR models reveals cell-type-specific splicing mechanisms

To further pinpoint the factors underlying differences in splicing between glia and neurons, we analyzed the coefficients of the logistic regression models. These coefficients reflect the importance of each RBP binding position in regulating cell-type-specific splicing. We compared the coefficients of four models for the hippocampus (two cell types, and two training sets) and focused on features present in at least 50 exons and with a coefficient > 0.05 in at least

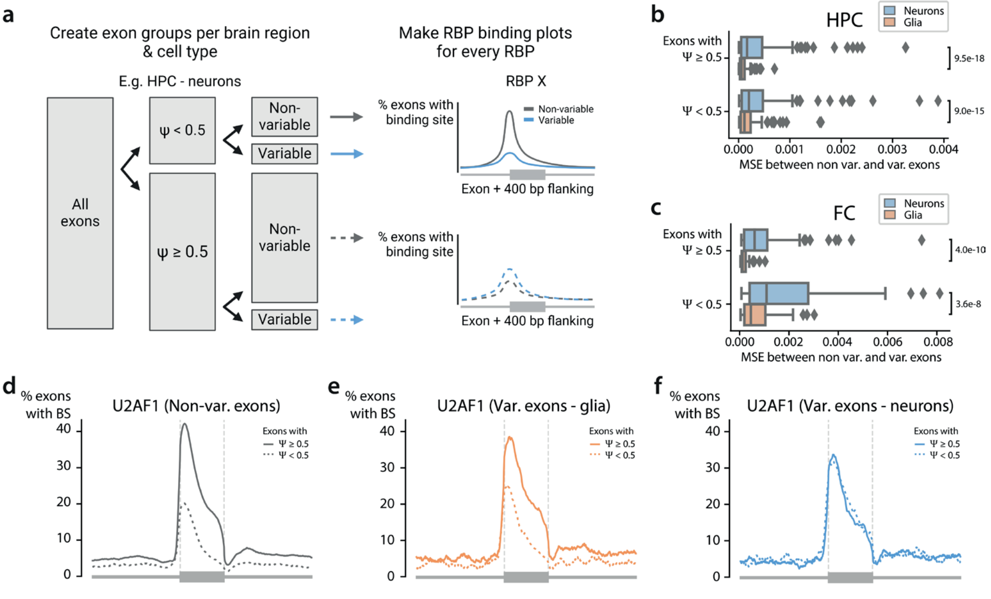


Figure 2. The difference in RBP binding profiles between non-variable and variable exons. A) Schematic overview showing how to generate the RBP binding profiles of non-variable ($|\Delta\Psi_{glia-neur}| \leq 0.25$) and variable ($|\Delta\Psi_{glia-neur}| > 0.25$) exons in neurons in the hippocampus. We generated these RBP binding profiles for every RBP and split the exons based on their Ψ value (threshold = 0.5) and their variability. We calculated the mean-squared error (MSE) between the profiles in non-variable and variable exons. We do this for the exons with a high and low Ψ value resulting in four comparisons per RBP. **B-C)** Boxplot showing the MSE between the RBP profiles in non-variable and variable exons in neurons (blue) and non-variable and variable exons in glia (orange) for the **B)** hippocampus and **C)** frontal cortex. Every point in the boxplot is one RBP. P-values are calculated using a two-sided paired Wilcoxon signed-rank test. **D-F)** Binding profile of U2AF1 in **D)** non-variable exons, **E)** variable exons in glia, and **F)** variable exons in neurons.

one model (191 out of 732 features). The model coefficients first cluster based on which exons are used during training (all vs. variable) (Figure 3A). This clustering indicates that the mechanisms for non-variable and variable exons, represented by the LR_{all} and $\text{LR}_{\text{var0.1}}$, differ more than the cell-type-specific mechanisms. The RBPs cluster into two groups: features with positive and features with negative coefficients (Figure 3A). As expected, splicing repressors, which are part of the heterogeneous nuclear ribonucleoproteins (hnRNP) family [39], have a largely negative weight in all models (Figure 3B). PTBP1, for which we saw a difference between the non-variable and variable exons in the hippocampus, is a member of the hnRNP family and has a potential position-dependent effect in glia based on the RBP binding profiles (Figure S11C). The $\text{LR}_{\text{var0.1-glia-HPC}}$ model correctly learned this effect: PTBP1 binding at the 3’ splice site and within the exon have coefficients of -0.05 and 0.01 respectively. The model coefficient for PTBP1 binding at the 3’ splice site is among the ten features that differ the most between glia and neurons (Figure 3C, $\text{LR}_{\text{var0.1-glia-HPC}}$ vs $\text{LR}_{\text{var0.1-neur-HPC}}$) which indicates a potential cell-type-specific effect corresponding to the established switch between PTBP1 and PTBP2 [40–42].

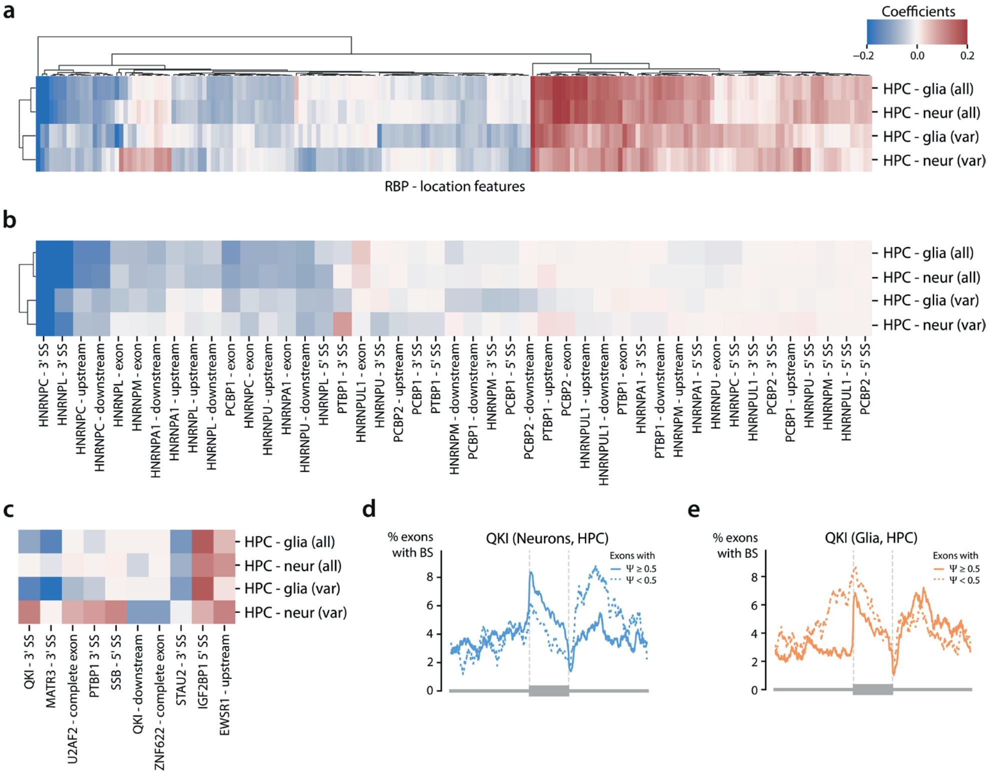


Figure 3. Interpretation of the logistic regression models. A) Heatmap showing the coefficients for the RBP-location features in the different logistic regression models. The input features are filtered using a minimum of 50 RBP sites and a value of at least 0.05 in one of the models. The values are clipped between -0.2 and 0.2. **B)** Heatmap showing coefficients of hnRNPs in the different models. **C)** Heatmap showing the top 10 cell-type-specific input features with the biggest difference between HPC-glia (var) and HPC-neur (var). **D-E)** Binding profiles of QKI in variable exons in neurons and glia.

QKI binding at the 3’ splice site has the strongest cell-type-specific effect in the hippocampus (model coefficient =-0.15 vs. 0.12 for glia and neurons respectively), which reflects differences in the RBP binding profiles (Figure 3D-E). In glia, a binding site that overlaps the 3’ splice site leads to lower inclusion rates, while the opposite happens in neurons. In the scRNA-seq data, QKI has higher expression in glia compared to neurons in the hippocampus (Wilcoxon rank sum test, adj. p-value < 2.2e-16) (Figure S13). Both observations correspond to the known mechanism of QKI, which inhibits splicing by competing with the core splicing machinery [10,43]. In mice, QKI is important during myelination and oligodendrocyte differentiation [44,45]. Its role in the human brain is less studied, but a role in oligodendrocyte formation and Schizophrenia has been suggested [46,47]. Interestingly, variable exons are enriched for QKI binding sites compared to non-variable exons (Fisher’s exact test, adj. p-value = 1.6e-13). Besides the 3’ splice site, QKI binding downstream of an exon is also in the top 10 cell-type-specific features. The effect of QKI downstream of an exon is the opposite compared to QKI binding at the 3’ splice site, which indicates a potential position-dependent effect of QKI. Such position-dependent regulation of QKI has been shown in lung cancer [48] but, to our knowledge, not in the brain.

In contrast to QKI, most of the cell-type-specific RBPs identified using our LR models are neither differentially expressed nor differentially spliced. Exceptions are STAU2, which is upregulated in neurons (Wilcoxon rank sum test, adj. p-value < 3.39e-16), and EWSR1, which is differentially spliced (Table S3). The latter could indicate that distinct isoforms of EWSR1 influence RNA splicing in different ways.

7.2.6 The sequence close to the splice sites is most important for predicting exon inclusion

Given that the RBP-binding-site data is not brain-specific and that it lacked measurements from some key RBPs, we set out to identify sequence features that influence Ψ predictions in the deep learning models. We used *in-silico* saturation mutagenesis (ISM, Methods) to systematically predict how nucleotide substitutions in the input sequence affect the predicted Ψ value [49–52]. Since $DL_{var0.1}$ performed considerably worse than DL_{all} (Figure 1D), we focused on interpreting DL_{all} for glia in the hippocampus, which had higher prediction accuracy than neurons, instead of looking for cell-type-specific effects.

Since ISM is computationally expensive, we mutated the input sequence of the 9,929 exons with $|\Delta\Psi_{glia-neur}| > 0.1$ instead of all exons. The ISM score indicates how much a mutation increases or decreases the predicted Ψ value compared to the average prediction at that position for that sequence (Methods). As expected, mutations around the splice sites and within the exon strongly affect the predicted Ψ value (Figure 4A). These results reflect the known importance of the splice site’s consensus sequence to be recognized by the splicing machinery. The two nucleotides before and after the exon -the AG acceptor and GU donor dinucleotides- have the strongest predicted effects. Looking at the maximum absolute ISM score, only mutations within a range of 50bp upstream of the 3’ splice site and 150bp downstream of the 5’ splice site have a value > 0.1 (Figure S14). This is in line with a recent computational model that predicted human splice sites using a window of 400bp on each side of the splice site and obtained an overall accuracy of 96% [53]. However, smaller values of >0.05 could be observed across almost the whole input sequence. Although distant splicing regulators have been reported [54], potential variability in distant motifs and/or their position may prevent their detection by our model.

Besides the region around the exon of interest, we observed higher-than-average ISM scores within nearby exons and their flanking region (Figure S15). The enrichment of RBP binding

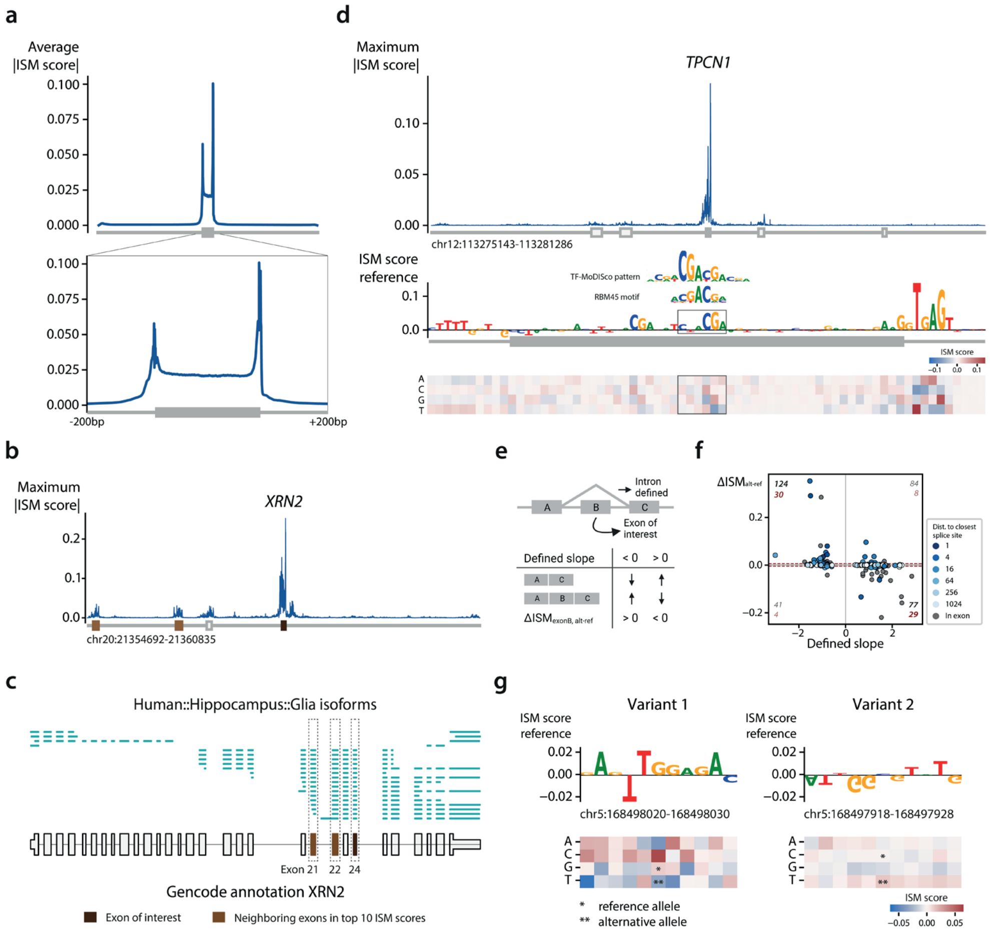


Figure 4. Interpretation of the deep learning model for glia in the hippocampus. A) Average absolute ISM score across the 9,929 exons. The mutations within the exons are binned in 300 bins. The zoomed-in plot ranges from 200bp upstream of the 3' splice site to 200bp downstream of the 5' splice site. **B)** Mutation profile for an exon in *XRN2*. The colors of the exons below the profile indicate the exon of interest and the neighboring exons which have an ISM score in the top 10. **C)** Single-cell long reads for *XRN2*. Each line is a single cDNA molecule. The bottom black track shows the Gencode annotation. **D)** Mutation profile for an exon in *TPCN1*. In the exon, a motif corresponding to RBM45 is found. **E)** Schematic overview of the sQTL analysis. **F)** Scatterplot showing the predicted effect for each variant. The color of the points indicates the distance to the closest splice site. A grey dot means that a variant falls within the exon of interest. The numbers in black and red indicate the number of predictions in a quadrant when no threshold and a threshold of 0.005 are used respectively. **G)** ISM scores for two variants related to the same exon of *RARS1*. A negative effect, corresponding to the positive slope, is predicted for the first variant. A smaller, but positive effect is predicted for the second variant.

sites in these regions could explain the higher scores. Alternatively, our model potentially recognized coordinated events between exons. To test this, we selected the top 10 exons with the highest absolute ISM scores within their neighboring exons and visualized the single-cell long reads from our data that span both exons (Methods). These reads can inform whether the two exons pair non-randomly (thus in coordination [21,23,55,56]) or randomly. Exon 24 in *XRN2* appeared twice in the top 10 list with two neighboring exons (exons 21 and 22) strongly influencing its Ψ value (Table S4). All three exons (21, 22, and 24) have a Ψ value of around 0.8 and the exons are either all included or all excluded in the single-cell long-read data, suggesting these exons are mutually associated (Figure 4BC). Mutations affecting the inclusion of one of these exons will most likely affect the other exons as well. In the top 10 scores, four other cases could pinpoint exon coordination events (Figure S16-19). In the remaining four cases, the exons pair randomly, so there is no evidence of exon-exon coordination (Figure S20-23).

To further interpret sequences with a high ISM score, we used TF-MoDISco [57] to identify motifs in sequences with large effects on exon inclusion. Since the region around the splice site had the highest ISM scores, many of the top motifs identified by TF-MoDISco correspond to the consensus splice sites and associated motifs, including the well-known AG acceptor dinucleotide, the poly-pyrimidine tract (PPT) upstream of the exon, and the extended splice donor motif with the GU dinucleotide (Supplementary File 1, Figure S24). We also found motifs that match known RBP binding motifs, which were not in our input data for the LR model, and hence could not be tested for cell-type-specific effects. For example, we found a motif corresponding to RBM45 in exon 12 of *TPCN1* (Figure 4D, Table S4), which seems to promote exon inclusion. RBM45 regulates constitutive splicing and can probably activate or repress the inclusion of an exon, but the exact mechanisms are currently unknown [58]. Taken together, characterizing important sequence features from DL models can identify splicing regulators beyond those we can identify based on available RBP measurements.

7.2.7 Prioritizing the effect of splice QTLs using the DL models

So far, we showed how LR and DL model interpretations can be used to reveal the regulatory mechanisms of RBPs governing cell-type-specific exon inclusion. Besides this fundamental knowledge, we can use our DL models to predict the effects of genetic variants on splicing. Accurately predicting these effects can help prioritize variants of interest. To test the relevance of our model predictions for genetic variants, we used splicing quantitative trait loci (sQTLs) from the hippocampus data from GTEx v8 [59]. Variants in this dataset are linked to intron-excision ratios instead of exon inclusion. We extracted introns and their corresponding variant(s) that span an exon in our data and predicted the effect of the variant(s) on that exon (Figure 4E). In total, 326 variants are within the input range of our model. These variants correspond to 122 introns and 158 exons. Some introns thus span multiple exons and most introns have multiple variants linked. For every variant, a slope indicates whether the corresponding intron is excised more or less compared to the reference allele. We expect negative slopes to correspond to an increased Ψ value of the exon of interest which would result in $\Delta ISM_{alt-ref} > 0$. Conversely a positive slope would result in $\Delta ISM_{alt-ref} < 0$ (Figure 4E). However, more complex scenarios, such as a variant affecting adjacent exons, may arise as well.

Using our model, we predicted an effect ($|\Delta ISM_{alt-ref}| > 0.005$) for 71 out of 326 variants which corresponds to 61 of the 122 introns. For 83% (59 out of 71) of these variants, our model predicts the expected effect correctly (Figure 4F, S25). Most of the variants with an effect are very close to the splice sites: 74.6% are within the exon or a distance of 15bp to either the 3' or 5' splice site. These cases most likely affect exonic splicing enhancers or the binding of U1 and U2 snRNA. For 14 of 61 introns where our model did not predict an effect, all corresponding variants are outside of the intron itself. Here, the splicing of adjacent exons is most likely altered instead of our exon of interest. For 2 of these 14 exons, all variants are even outside of the gene itself.

Three exons have multiple corresponding variants with a predicted effect. For exon 15 in *ZNF880* (Table S4), three variants have a predicted expected effect. The other two exons, however, have two variants with a contradicting predicted effect. In both cases, the variant with the biggest predicted effect is in line with the slope of the sQTL of the intron. For exon 25 in *RARS1* (Table S4), for instance, variant one is located in the exon (168,498,025; G → T) and variant two is located before the exon (168,497,923; C → T). For variant one, our model predicted the expected effect, while our model predicted the opposite for variant two (Figure 4G). Variant one, the variant with the biggest and correctly predicted effect, is located in a binding site for SRSF1 according to eCLIP data [8]. RNA recognition motif 2 (RRM2) of SRSF1 interacts with the GGA motif. A G → T mutation in the first nucleotide will thus hinder the binding of SRSF1 [60]. Variant two is located in a stretch of G's. At this location, there's a binding site for ELAVL1, a protein regulating mRNA stability, and hnRNP family member HNRNPK, which tends to repress splicing [8]. Using the DL models, we can thus correctly predict the effect for most sQTLs and prioritize their effects.

7.3 Discussion

We trained logistic regression and deep learning models to predict cell-type-specific exon inclusion in human brain samples. Since this is the first attempt to leverage long-read single-cell sequencing data for this task, we can use our models to decipher the grammar underlying cell-type specificity of splicing. Using model interpretation, we pinpointed interesting RBPs, such as QKI, that could drive differential splicing between neurons and glia. Furthermore, we show that the location of RBP binding sites differs more between variable and non-variable exons in neurons compared to glia. This indicates that the splicing mechanisms controlling exon inclusion in neurons are more different compared to the general mechanism.

For most RBPs, RBP binding profiles of non-variable exons with high and low Ψ values showed distinct patterns. Considering U2AF1 for example, exons with a high Ψ value are more likely to have a binding site close to the 3' splice site compared to exons with a low Ψ value. These RBPs behave differently in variable exons in neurons, and for most RBPs the difference between exons with a low and high Ψ value is missing. These features are thus not informative for neurons, which explains the low performance of the logistic regression models on neurons. The U2AF heterodimer, composed of U2AF1 and U2AF2, is believed to bind every polypyrimidine tract and AG dinucleotide in 3' splice site regions [61–63]. Binding may not happen on specific sites repressed by other factors. The potential binding sites are still there, but they might be used by a competing RBP in neurons. Interestingly, most RBPs are not differentially expressed or differentially spliced between neurons and glia. For these RBPs, post-translational modifications, such as phosphorylation, might differ between neurons and glia and could change their function [64,65]. Furthermore, RBP binding sites measured in non-brain cell lines might not always be representative of splicing in the hippocampus and frontal cortex. The expression of RBPs can differ dramatically between the non-brain and brain tissues as was seen for PTBP1.

The deep learning models, however, also perform poorly on the variable exons in neurons. The model trained on all exons focuses only on learning the general splicing mechanisms, and the model trained on the variable exons might not have enough training data. In glia, the model trained on all exons performs well on the variable exons. Again indicating that the variable exons in glia follow the rules of the general splicing mechanisms more. The worse performance of the $DL_{all\text{-}seq}$ models on neurons, in combination with the distinct RBP binding profiles, supports our conclusion that the splicing mechanisms in variable exons in neurons diverged from the mechanisms in non-variable exons.

A potential explanation, in line with the diverged RBP binding sites, is that splicing in neurons is less sequence-dependent. Other factors, such as chromatin features and polymerase speed [66–79], RNA methylation [80–82] as well as other modifications, and transcription factor binding sites [83], influence splicing as well. These features might explain the difference between neurons and glia. Altered chromatin accessibility or RNA methylation, could, for instance, explain why certain RBP binding sites are not used in neurons anymore. Furthermore, neuronal genes- by definition more expressed in neurons- are more susceptible to missplicing [84]. While we did not focus on missplicing, this indicates that splicing mechanisms might be different in neurons. Also, the gene expression of human neurons diverged faster from other primates compared to glia [85]. A similar divergence could have occurred with the splicing mechanisms.

For the deep learning model, we tested the effect of different lengths for the input sequence. Even though all lengths showed a very similar performance, we used a relatively long input sequence (6,144 bp) which had the advantage that we could predict the effect of more mutations. When predicting the effect of sQTLs, however, we predict a strong effect mainly for variants close to the exon of interest. The region close to the splice sites, however, still contributes the most to the predictions. This is in contrast to splice site predictions from SpliceAI, for which an input sequence of 10kb significantly outperforms 400 bp [29]. SPLAM, however, outperforms SpliceAI while only using 400 bp [53]. Of note, this does not preclude the mechanistic influence on splicing decisions by motifs further upstream. Rather, these data suggest that such distant RNA binding sites are highly variable regarding their position to the exon. This variability in position could prevent the model from detecting such motifs. Similar observations have been made for models that predict gene expression. Even though the best-performing model uses a long input sequence (196kb), only one-third of the receptive field is used during predictions and distal enhancers are not captured by the model [51,86].

Another possible advantage of a longer input sequence is that it would be possible to look at coordinated events. Exons in the human brain are often mutually associated or mutually exclusive [23,55,87–89]. Such events can even be cell-type-specific. For instance, two neighboring exons in *WDR49* are perfectly coordinated in astrocytes only [23]. Using our model, the ISM scores within neighboring exons are higher than the ISM scores of the rest of the sequence. For some exons, these higher scores indeed indicate that there is exon-exon coordination. Since exon-exon coordination is so common, predicting such events might be more beneficial than focusing on individual exons.

Furthermore, the longer input sequence enables predicting the effect of more sQTLs. However, most variants the model predicted an effect for are near the splice sites. For these variants, the model obtained a high accuracy (83%) and could be used to prioritize the effect of sQTLs as well. Nonetheless, a limitation of the current DL models is that they lack cell-type specificity. The DL models need substantial training data, so training on all exons yielded the highest performance. As a consequence, these models focused on the general splicing mechanisms and yielded better performance on variable exons in glia than neurons.

In conclusion, to increase our understanding of (alternative) splicing in the brain, we trained two types of models to predict exon inclusion in neurons and glia of the hippocampus and frontal cortex. Ideally, these models make perfect predictions such that they can be used in the clinic for predicting the effects of variants or for personal splicing predictions. The performance of our models, however, is not optimal yet. Nevertheless, we show how model interpretation yields important biological discoveries including the different mechanisms in neurons and glia. This demonstrates the potential of using long-read single-cell data for this task.

7.4 Methods

7.4.1 Calculating cell-type-specific Ψ values

For the human data, we combined SnISOr-Seq data from 6 individuals for the hippocampus and 2 individuals for the frontal cortex (Table 1). For the mouse data, we combined ScISOr-Seq2 data from two mice for the hippocampus and two mice for the visual cortex (Table S2). Scisorseqr was used to map and align reads to GRCh38 for human and mm10 for mouse to identify splice sites for each dataset separately [24]. We used IsoQuant to correct the splice sites [90]. Using all exons appearing as an internal exon in a read, we calculated:

- The number of long-read molecules containing this exon (both splice sites included): X_{in}
- The number of long-read molecules assigned to the same gene as the exon, which skipped the exon but includes 50 bases on both sides: X_{out}
- The number of long-read molecules supporting the acceptor of the exon and ending on the exon: $X_{acc\ In}$
- The number of long-read molecules supporting the donor of the exon and ending on the exon: $X_{don\ In}$
- The number of long-read molecules overlapping the exon: X_{tot}

Non-annotated exons with one or two annotated splice sites, ≥70 bases of non-exonic (in the annotation) bases, were excluded as intron-retention events or alternative acceptors/donors We then calculated:

- $\Psi_{overall} = \frac{X_{in} + X_{accIn} + X_{donIn}}{X_{in} + X_{accIn} + X_{donIn} + X_{out}}$
- $\Psi_{acceptor} = \frac{X_{in} + X_{accIn}}{X_{in} + X_{accIn} + X_{out}}$
- $\Psi_{donor} = \frac{X_{in} + X_{donIn}}{X_{in} + X_{donIn} + X_{out}}$

If $0.02 \leq \Psi_i \leq 0.98$ where $i \in \{overall, acceptor, donor\}$ in the pseudo-bulk data, the exon was kept. Next, we filtered exons based on the number of reads. We only calculate $\Psi_{overall}$ for a cell type in a certain brain region if at least 10 long-read UMIs are sequenced across the different individuals ($X_{tot} \geq 10$). Since individuals of different datasets were sequenced using a different read depth, we normalized the read counts by dividing it by the total number of reads for an individual before calculating $\Psi_{overall}$. We then calculated $\Psi_{overall}$ for each cell type (Ψ_{neur} and Ψ_{glia}) for the hippocampus and frontal cortex. If there were not enough reads, for that exon and cell type $\Psi_{overall}$ was set to “NA”. We used the cell-type labels defined in the original datasets. For neurons, we grouped the inhibitory and excitatory neurons. For glia, we grouped the oligodendrocytes, astrocytes, and oligodendrocyte precursor cells.

7.4.2 Downsampling cell-type-specific Ψ values

In the human data, many exons (30,273 out of 68,215 for the hippocampus and 45,680 out of 56,427 for the frontal cortex) have $\Psi_{neur} > 0.9$, $\Psi_{glia} > 0.9$, and $|\Delta\Psi_{glia-neur}| < 0.03$. We downsampled these to 5,000 to make the distribution less skewed towards one. In the mouse hippocampus data, 18,351 out of 23,857 exons have $\Psi = 1$ in neurons and glia, so we downsampled these to 5,000 as well. For the visual cortex, 27,073 out of 48,515 exons have $\Psi_{neur} > 0.9$, $\Psi_{glia} > 0.9$, and $|\Delta\Psi_{glia-neur}| < 0.03$. We downsampled these to 5,000.

7.4.3 RBP-binding-site data

We downloaded the eCLIP data for 122 RBPs from the ENCODE portal (https://www.encodeproject.org/metadata/?status=released&internal_tags=ENCORE&assay_title=eCLIP&biosample_ontology.term_name=K562&target.investigated_as=RNA+binding+protein&biosample_ontology.term_name=HepG2&assembly=GRCh38&type=Experiment&files.processed=true). From this file list, we used the BED files that store the peaks per replicate. We merged peaks from different replicates or cell lines to ensure one BED file per RBP.

7.4.4 Logistic regression models

The logistic regression model is implemented as one fully connected layer between the input features (the RBP binding sites) and the output (the Ψ value) with a sigmoid activation function to scale the output between 0 and 1. The models are single-task models which means that a separate model was trained for each cell type. When training the model, we use a binary cross entropy loss with L1 and L2 regularization (alpha = 0.001, and L1-ratio = 0.7), a learning rate of 0.005, and a batch size of 256. As input for the logistic regression models, we counted the number of peaks in the BED files for every RBP and exon at six locations: 1) upstream of the exon (maximum 400 bp away from the splice site), 2) overlapping the 3' splice site, 3) within the exon, 4) spanning the exon, 5) overlapping the 5 splice site, and 6) downstream of the exon (maximum 400 bp away from the splice site). Since we used the eCLIP data of 122 RBPs and there are 6 possible locations, this resulted in 732 input features for every exon (Figure 1A). If peaks of different replicates were overlapping, we counted those peaks only once. The logistic regression model is implemented in PyTorch Lightning [91,92].

7.4.5 Deep learning models

We adapted the architecture of the Saluki model [37] by removing one convolutional layer, shortening the maximum sequence from 12,288 to 6,144 bp, and changing the final activation function to a sigmoid activation function (Figure S5). The exon of interest was centered in the middle of the input sequence. The input channels of the model depend on the input features used (sequence, splice sites, and/or RBP binding sites). For the sequence, we one-hot encoded the sequence which results in four channels. If the splice sites were used as input, this added an extra channel that indicates the start and end of the exon of interest. When adding the RBP binding sites, we add a channel for every RBP which one-hot encodes whether there is a binding site in any of the replicates of the eCLIP data for that RBP based on the BED files. Similar to the logistic regression models, we trained a model for every cell type separately. Even though we adapted the Saluki model, we retrained all the weights in the model. When adding the mouse data, we adapted the same approach as Saluki and made the model a multi-head model where the weights of the convolutional and recurrent neuronal network layers are shared and the weights of the fully connected layer are species-specific (Figure S5).When training the model, we used the same hyperparameters, including the learning rate, batch size, etc., as the original Saluki model (Figure S5). For the hippocampus, we tested how input-sequence length and the number of convolutional layers affect the performance. The benefit of a longer input sequence is that the model can learn how long-distance interactions of regulatory elements affect splicing, but these models contain more parameters and are more difficult to train. The different models performed similarly which indicates that the most important information is close to the splice sites of the exon (Figure S26). The model using 6,144 bp and five channels performed slightly better for both neurons and glia and therefore we used it during all the experiments.

7.4.6 Evaluation

We evaluated the performance of the models using a 10-fold cross-validation. We ensured that the same set of exons was always in the same test fold such that we could compare the performance of the models. Exons from the same gene were always in the same test fold. When training the deep learning models on human and mouse data simultaneously, we ensured that human-mouse homologs were in the same test fold. We used biomart to obtain the human-mouse homologs. Some exons do not have any binding sites measured for any of the RBPs (5,560 exons in the hippocampus and 3,462 in the frontal cortex). This could for instance happen if certain genes were not expressed in the cell lines when the RBP binding sites were measured. Since the logistic regression model could not predict a Ψ value for these exons, we filtered these from the training set used for the logistic regression model and from all test sets (to enable a fair comparison between the logistic regression and deep learning models). The deep learning models are thus trained on more exons (Table S1). In the test set, there are 1,827 and 1,072 variable exons for the hippocampus and frontal cortex respectively. We trained all models five times for every fold and averaged the predictions across these five runs. We evaluated the performance by calculating the Spearman correlation between the true and predicted Ψ values.

7.4.7 RBP binding profiles

We generated RBP binding profiles by calculating the fraction of exons with an RBP binding site at every location (400 bp upstream of the exon- 400 bp downstream of the exon). Since exons have variable lengths, we bin the exons in 50 bins and only include exons that are at least 50 bp long in the analysis. We also filter out exons without RBP binding sites. We calculate these profiles for four different groups of exons: 1) non-variable exons with $\Psi \geq 0.5$, 2) non-variable exons with $\Psi < 0.5$, 3) variable exons with $\Psi \geq 0.5$, and 4) variable exons with $\Psi < 0.5$. To define how much the mechanisms in the variable exons diverged from non-variable exons, we calculate the mean-squared error between the RBP binding profiles of the non-variable and variable exons. We do this for the exons with a high and low Ψ separately.

7.4.8 RBP expression data

We used the 10X scRNA-seq data from the same samples to look at the gene expression of the RBPs that were measured using the eCLIP data. We used Seurat v4 for the analysis [93]. To create the heatmap in Figure S13, we normalized the data per dataset using log normalization and a scale factor of 1e6. Next, we averaged the expression over all the cells. We plotted the $\log(x+1)$ values. We used the `FindConservedMarkers()` function using the default parameters (including Bonferroni multiple testing correction) from Seurat to find differentially expressed RBPs between neurons and glia. This tests for differentially expressed genes per individual and merges the results.

7.4.9 Interpretation of logistic regression model

For the interpretation of the logistic regression models, we looked at the coefficients of the input features. To obtain one value per input feature, we average the coefficients of the 10 folds and 5 runs per fold (so the average across 50 models in total). We only compared the coefficients across models, if there were at least 50 exons with a binding site for that input feature.

7.4.10 *In-silico* saturation mutagenesis

We used *in-silico* saturation mutagenesis (ISM) to interpret how nucleotide substitutions in the input sequence affect the predictions. We did this for 9,929 exons using the $DL_{\text{all-seq-m}}$ model trained on glia in the hippocampus. For every exon, we used the fold for which that exon was in the test set. We averaged the predictions across the 5 runs. The ISM score is defined as follows:

$$ISM_{e,p,n} = \Psi_{pred,e,p,n} - \tfrac{1}{4}\sum_{i \in A,C,G,T} \Psi_{pred,e,p,i}$$

where e is the exon we predict the Ψ value for, and p and n are the position and nucleotide used at that position respectively. To visualize the ISM scores across the input sequence, we binned the upstream region, exon, and downstream region since they all had varying lengths.

7.4.11 Analysis of neighboring exons

We compared the ISM scores at the exon of interest, the neighboring exons, and the remaining sequence. We extracted the locations of annotated exons from GENCODE v35 [94]. The ISM scores for the exon of interest and the neighboring exons include the flanking sequence of 150 bp upstream and downstream of the exon. Next, we selected ten exons on the positive strand with the highest absolute ISM scores in a neighboring exon. We visualized the long-reads spanning both exons using ScisorWiz [95]

7.4.12 Motif discovery

We used TF-MoDISco-lite (v2.2.0) [57] to discover motifs using the ISM scores as input. When creating the report, we compare the found motifs to the position weight matrices from oRNAment which includes motifs found using RNAcompete and RNA-bind-n-seq experiments [8,96,97]. TF-MoDISco-lite is designed for DNA instead of RNA and tries both the forward strand and its reverse complement when finding seqlets (parts of the sequence with high ISM scores). We used the results file, to check whether the forward or reverse complement was used to generate a motif. We kept forward motifs if at least for 25 sequences the forward strand was used. We kept the reverse motif if at least for 25 sequences the reverse complement was used.

7.4.13 sQTL analysis

We used the sQTLs defined for the hippocampus in GTEx v8. These variants are linked to introns instead of exons. We predicted the effect for variants that are linked to an intron that spans an exon in our dataset (Figure 4E). For most introns, there are multiple variants linked to them. We only predicted the effect for the best variants (the variants with the lowest p-value for an intron). For most introns, there were still more than two after this filter.

7.4.14 Exon naming

We named exons after their position in the transcript by counting their position in the GTF file. A conversion from exon names to genomic coordinates can be found in Table S4.

7.5 Code and data availability

The Ψ values, predictions, and RBP binding profiles are available on Zenodo: https://zenodo.org/doi/10.5281/zenodo.10669666. The code to reproduce the figures, and train your logistic regression or deep learning models can be found on GitHub: https://github.com/lcmmichielsen/PSI_pred.

Bibliography

1. Cheng J, Zhou T, Liu C, Shapiro JP, Brauer MJ, Kiefer MC, et al. Protection from Fas-mediated apoptosis by a soluble form of the Fas molecule. Science. 1994;263: 1759–1762. doi:10.1126/science.7510905
2. Wright CJ, Smith CWJ, Jiggins CD. Alternative splicing as a source of phenotypic diversity. Nat Rev Genet. 2022;23: 697–710. doi:10.1038/s41576-022-00514-4
3. Yang X, Coulombe-Huntington J, Kang S, Sheynkman GM, Hao T, Richardson A, et al. Widespread Expansion of Protein Interaction Capabilities by Alternative Splicing. Cell. 2016;164: 805–817. doi:10.1016/j.cell.2016.01.029
4. Pan Q, Shai O, Lee LJ, Frey BJ, Blencowe BJ. Deep surveying of alternative splicing complexity in the human transcriptome by high-throughput sequencing. Nat Genet. 2008;40: 1413–1415. doi:10.1038/ng.259
5. Wang ET, Sandberg R, Luo S, Khrebtukova I, Zhang L, Mayr C, et al. Alternative isoform regulation in human tissue transcriptomes. Nature. 2008;456: 470–476. doi:10.1038/nature07509
6. Yeo G, Holste D, Kreiman G, Burge CB. Variation in alternative splicing across human tissues. Genome Biol. 2004;5: R74. doi:10.1186/gb-2004-5-10-r74
7. Gerstberger S, Hafner M, Tuschl T. A census of human RNA-binding proteins. Nat Rev Genet. 2014;15: 829–845. doi:10.1038/nrg3813
8. Van Nostrand EL, Freese P, Pratt GA, Wang X, Wei X, Xiao R, et al. A large-scale binding and functional map of human RNA-binding proteins. Nature. 2020;583: 711–719. doi:10.1038/s41586-020-2077-3
9. Zhang X, Chen MH, Wu X, Kodani A, Fan J, Doan R, et al. Cell-Type-Specific Alternative Splicing Governs Cell Fate in the Developing Cerebral Cortex. Cell. 2016;166: 1147–1162.e15. doi:10.1016/j.cell.2016.07.025
10. Fisher E, Feng J. RNA splicing regulators play critical roles in neurogenesis. Wiley Interdiscip Rev RNA. 2022;13: e1728. doi:10.1002/wrna.1728
11. Zhang C, Frias MA, Mele A, Ruggiu M, Eom T, Marney CB, et al. Integrative modeling defines the Nova splicing-regulatory network and its combinatorial controls. Science. 2010;329: 439–443. doi:10.1126/science.1191150
12. Wang Z, Rolish ME, Yeo G, Tung V, Mawson M, Burge CB. Systematic identification and analysis of exonic splicing silencers. Cell. 2004;119: 831–845. doi:10.1016/j.cell.2004.11.010
13. Barash Y, Calarco JA, Gao W, Pan Q, Wang X, Shai O, et al. Deciphering the splicing code. Nature. 2010;465: 53–59. doi:10.1038/nature09000
14. Barash Y, Vaquero-Garcia J, González-Vallinas J, Xiong HY, Gao W, Lee LJ, et al. AVISPA: a web tool for the prediction and analysis of alternative splicing. Genome Biol. 2013;14: R114. doi:10.1186/gb-2013-14-10-r114
15. Xiong HY, Alipanahi B, Lee LJ, Bretschneider H, Merico D, Yuen RKC, et al. RNA splicing. The human splicing code reveals new insights into the genetic determinants of disease. Science. 2015;347: 1254806. doi:10.1126/science.1254806
16. Sharon D, Tilgner H, Grubert F, Snyder M. A single-molecule long-read survey of the human transcriptome. Nat Biotechnol. 2013;31: 1009–1014. doi:10.1038/nbt.2705
17. Au KF, Sebastiano V, Afshar PT, Durruthy JD, Lee L, Williams BA, et al. Characterization of the human ESC transcriptome by hybrid sequencing. Proc Natl Acad Sci U S A. 2013;110: E4821–30. doi:10.1073/pnas.1320101110
18. Marx V. Method of the year: long-read sequencing. Nat Methods. 2023;20: 6–11. doi:10.1038/s41592-022-01730-w
19. Foord C, Hsu J, Jarroux J, Hu W, Belchikov N, Pollard S, et al. The variables on RNA molecules: concert or cacophony? Answers in long-read sequencing. Nat Methods. 2023;20: 20–24. doi:10.1038/s41592-022-01715-9

20. Lucas MC, Novoa EM. Long-read sequencing in the era of epigenomics and epitranscriptomics. Nat Methods. 2023;20: 25–29. doi:10.1038/s41592-022-01724-8

21. Gupta I, Collier PG, Haase B, Mahfouz A, Joglekar A, Floyd T, et al. Single-cell isoform RNA sequencing characterizes isoforms in thousands of cerebellar cells. Nat Biotechnol. 2018;36: 1197–1202. doi:10.1038/nbt.4259

22. Singh M, Al-Eryani G, Carswell S, Ferguson JM, Blackburn J, Barton K, et al. High-throughput targeted long-read single cell sequencing reveals the clonal and transcriptional landscape of lymphocytes. Nat Commun. 2019;10: 3120. doi:10.1038/s41467-019-11049-4

23. Hardwick SA, Hu W, Joglekar A, Fan L, Collier PG, Foord C, et al. Single-nuclei isoform RNA sequencing unlocks barcoded exon connectivity in frozen brain tissue. Nat Biotechnol. 2022. doi:10.1038/s41587-022-01231-3

24. Joglekar A, Prjibelski A, Mahfouz A, Collier P, Lin S, Schlusche AK, et al. A spatially resolved brain region- and cell type-specific isoform atlas of the postnatal mouse brain. Nat Commun. 2021;12: 463. doi:10.1038/s41467-020-20343-5

25. Joglekar A, Hu W, Zhang B, Narykov O, Diekhans M, Balacco J, et al. Single-cell long-read mRNA isoform regulation is pervasive across mammalian brain regions, cell types, and development. bioRxiv. 2023. p. 2023.04.02.535281. doi:10.1101/2023.04.02.535281

26. Irimia M, Weatheritt RJ, Ellis JD, Parikshak NN, Gonatopoulos-Pournatzis T, Babor M, et al. A highly conserved program of neuronal microexons is misregulated in autistic brains. Cell. 2014;159: 1511–1523. doi:10.1016/j.cell.2014.11.035

27. Parikshak NN, Swarup V, Belgard TG, Irimia M, Ramaswami G, Gandal MJ, et al. Genome-wide changes in lncRNA, splicing, and regional gene expression patterns in autism. Nature. 2016;540: 423–427. doi:10.1038/nature20612

28. Gonatopoulos-Pournatzis T, Blencowe BJ. Microexons: at the nexus of nervous system development, behaviour and autism spectrum disorder. Curr Opin Genet Dev. 2020;65: 22–33. doi:10.1016/j.gde.2020.03.007

29. Jaganathan K, Kyriazopoulou Panagiotopoulou S, McRae JF, Darbandi SF, Knowles D, Li YI, et al. Predicting Splicing from Primary Sequence with Deep Learning. Cell. 2019;176: 535–548.e24. doi:10.1016/j.cell.2018.12.015

30. Ji Y, Zhou Z, Liu H, Davuluri RV. DNABERT: pre-trained Bidirectional Encoder Representations from Transformers model for DNA-language in genome. Kelso DJ, Kelso J, editors. Bioinformatics. 2021. doi:10.1093/bioinformatics/btab083

31. Zeng T, Li YI. Predicting RNA splicing from DNA sequence using Pangolin. Genome Biol. 2022;23: 103. doi:10.1186/s13059-022-02664-4

32. Cheng J, Çelik MH, Kundaje A, Gagneur J. MTSplice predicts effects of genetic variants on tissue-specific splicing. Genome Biol. 2021;22: 94. doi:10.1186/s13059-021-02273-7

33. Kuroyanagi H. Fox-1 family of RNA-binding proteins. Cell Mol Life Sci. 2009;66: 3895–3907. doi:10.1007/s00018-009-0120-5

34. Dredge BK, Stefani G, Engelhard CC, Darnell RB. Nova autoregulation reveals dual functions in neuronal splicing. EMBO J. 2005;24: 1608–1620. doi:10.1038/sj.emboj.7600630

35. Ule J, Ule A, Spencer J, Williams A, Hu J-S, Cline M, et al. Nova regulates brain-specific splicing to shape the synapse. Nat Genet. 2005;37: 844–852. doi:10.1038/ng1610

36. Jensen KB, Dredge BK, Stefani G, Zhong R, Buckanovich RJ, Okano HJ, et al. Nova-1 regulates neuron-specific alternative splicing and is essential for neuronal viability. Neuron. 2000;25: 359–371. doi:10.1016/s0896-6273(00)80900-9

37. Agarwal V, Kelley DR. The genetic and biochemical determinants of mRNA degradation rates in mammals. Genome Biol. 2022;23: 245. doi:10.1186/s13059-022-02811-x

38. Llorian M, Schwartz S, Clark TA, Hollander D, Tan L-Y, Spellman R, et al. Position-dependent alternative splicing activity revealed by global profiling of alternative splicing events regulated by PTB. Nat Struct Mol Biol. 2010;17: 1114–1123. doi:10.1038/nsmb.1881

39. Matlin AJ, Clark F, Smith CWJ. Understanding alternative splicing: towards a cellular code. Nat Rev Mol Cell Biol. 2005;6: 386–398. doi:10.1038/nrm1645

40. Vuong JK, Lin C-H, Zhang M, Chen L, Black DL, Zheng S. PTBP1 and PTBP2 Serve Both Specific and Redundant Functions in Neuronal Pre-mRNA Splicing. Cell Rep. 2016;17: 2766–2775. doi:10.1016/j.celrep.2016.11.034

41. Boutz PL, Stoilov P, Li Q, Lin C-H, Chawla G, Ostrow K, et al. A post-transcriptional regulatory switch in polypyrimidine tract-binding proteins reprograms alternative splicing in developing neurons. Genes Dev. 2007;21: 1636–1652. doi:10.1101/gad.1558107

42. Makeyev EV, Zhang J, Carrasco MA, Maniatis T. The MicroRNA miR-124 promotes neuronal differentiation by triggering brain-specific alternative pre-mRNA splicing. Mol Cell. 2007;27: 435–448. doi:10.1016/j.molcel.2007.07.015

43. Zong F-Y, Fu X, Wei W-J, Luo Y-G, Heiner M, Cao L-J, et al. The RNA-binding protein QKI suppresses cancer-associated aberrant splicing. PLoS Genet. 2014;10: e1004289. doi:10.1371/journal.pgen.1004289

44. Darbelli L, Vogel G, Almazan G, Richard S. Quaking Regulates Neurofascin 155 Expression for Myelin and Axoglial Junction Maintenance. J Neurosci. 2016;36: 4106–4120. doi:10.1523/JNEUROSCI.3529-15.2016

45. Darbelli L, Choquet K, Richard S, Kleinman CL. Transcriptome profiling of mouse brains with qkI-deficient oligodendrocytes reveals major alternative splicing defects including self-splicing. Sci Rep. 2017;7: 7554. doi:10.1038/s41598-017-06211-1

46. Haroutunian V, Katsel P, Dracheva S, Davis KL. The Human Homolog of the QKI Gene Affected in the Severe Dysmyelination “Quaking” Mouse Phenotype: Downregulated in Multiple Brain Regions in Schizophrenia. AJP. 2006;163: 1834–1837. doi:10.1176/ajp.2006.163.10.1834
47. Åberg K, Saetre P, Jareborg N, Jazin E. Human QKI, a potential regulator of mRNA expression of human oligodendrocyte-related genes involved in schizophrenia. Proceedings of the National Academy of Sciences. 2006;103: 7482–7487. doi:10.1073/pnas.0601213103
48. Wang J-Z, Fu X, Fang Z, Liu H, Zong F-Y, Zhu H, et al. QKI-5 regulates the alternative splicing of cytoskeletal gene ADD3 in lung cancer. J Mol Cell Biol. 2021;13: 347–360. doi:10.1093/jmcb/mjaa063
49. Kelley DR, Snoek J, Rinn JL. Basset: learning the regulatory code of the accessible genome with deep convolutional neural networks. Genome Res. 2016;26: 990–999. doi:10.1101/gr.200535.115
50. Kelley DR, Reshef YA, Bileschi M, Belanger D, McLean CY, Snoek J. Sequential regulatory activity prediction across chromosomes with convolutional neural networks. Genome Res. 2018;28: 739–750. doi:10.1101/gr.227819.117
51. Avsec Ž, Agarwal V, Visentin D, Ledsam JR, Grabska-Barwinska A, Taylor KR, et al. Effective gene expression prediction from sequence by integrating long-range interactions. Nat Methods. 2021;18: 1196–1203. doi:10.1038/s41592-021-01252-x
52. Zhou J, Troyanskaya OG. Predicting effects of noncoding variants with deep learning-based sequence model. Nat Methods. 2015;12: 931–934. doi:10.1038/nmeth.3547
53. Chao K-H, Mao A, Salzberg SL, Pertea M. Splam: a deep-learning-based splice site predictor that improves spliced alignments. bioRxiv. 2023. p. 2023.07.27.550754. doi:10.1101/2023.07.27.550754
54. Lenasi T, Peterlin BM, Dovc P. Distal regulation of alternative splicing by splicing enhancer in equine beta-casein intron 1. RNA. 2006;12: 498–507. doi:10.1261/rna.7261206
55. Tilgner H, Jahanbani F, Blauwkamp T, Moshrefi A, Jaeger E, Chen F, et al. Comprehensive transcriptome analysis using synthetic long-read sequencing reveals molecular co-association of distant splicing events. Nat Biotechnol. 2015;33: 736–742. doi:10.1038/nbt.3242
56. Fededa JP, Petrillo E, Gelfand MS, Neverov AD, Kadener S, Nogués G, et al. A polar mechanism coordinates different regions of alternative splicing within a single gene. Mol Cell. 2005;19: 393–404. doi:10.1016/j.molcel.2005.06.035
57. Shrikumar A, Tian K, Avsec Ž, Shcherbina A, Banerjee A, Sharmin M, et al. Technical Note on Transcription Factor Motif Discovery from Importance Scores (TF-MoDISco) version 0.5.6.5. arXiv [cs.LG]. 2018. Available: http://arxiv.org/abs/1811.00416
58. Choi SH, Flamand MN, Liu B, Zhu H, Hu M, Wang M, et al. RBM45 is an m6A-binding protein that affects neuronal differentiation and the splicing of a subset of mRNAs. Cell Rep. 2022;40: 111293. doi:10.1016/j.celrep.2022.111293
59. GTEx Consortium. The GTEx Consortium atlas of genetic regulatory effects across human tissues. Science. 2020;369: 1318–1330. doi:10.1126/science.aaz1776
60. Cléry A, Sinha R, Anczuków O, Corrionero A, Moursy A, Daubner GM, et al. Isolated pseudo-RNA-recognition motifs of SR proteins can regulate splicing using a noncanonical mode of RNA recognition. Proc Natl Acad Sci U S A. 2013;110: E2802–11. doi:10.1073/pnas.1303445110
61. Merendino L, Guth S, Bilbao D, Martínez C, Valcárcel J. Inhibition of msl-2 splicing by Sex-lethal reveals interaction between U2AF35 and the 3’ splice site AG. Nature. 1999;402: 838–841. doi:10.1038/45602
62. Singh R, Valcárcel J, Green MR. Distinct binding specificities and functions of higher eukaryotic polypyrimidine tract-binding proteins. Science. 1995;268: 1173–1176. doi:10.1126/science.7761834
63. Wu S, Romfo CM, Nilsen TW, Green MR. Functional recognition of the 3’ splice site AG by the splicing factor U2AF35. Nature. 1999;402: 832–835. doi:10.1038/45590
64. O’Neill AC, Uzbas F, Antognolli G, Merino F, Draganova K, Jäck A, et al. Spatial centrosome proteome of human neural cells uncovers disease-relevant heterogeneity. Science. 2022;376: eabf9088. doi:10.1126/science.abf9088
65. Velázquez-Cruz A, Baños-Jaime B, Díaz-Quintana A, De la Rosa MA, Díaz-Moreno I. Post-translational Control of RNA-Binding Proteins and Disease-Related Dysregulation. Front Mol Biosci. 2021;8: 658852. doi:10.3389/fmolb.2021.658852
66. Agirre E, Oldfield AJ, Bellora N, Segelle A, Luco RF. Splicing-associated chromatin signatures: a combinatorial and position-dependent role for histone marks in splicing definition. Nat Commun. 2021;12: 682. doi:10.1038/s41467-021-20979-x
67. Petrova V, Song R, DEEP Consortium, Nordström KJV, Walter J, Wong JJL, et al. Increased chromatin accessibility facilitates intron retention in specific cell differentiation states. Nucleic Acids Res. 2022;50: 11563–11579. doi:10.1093/nar/gkac994
68. Luco RF, Pan Q, Tominaga K, Blencowe BJ, Pereira-Smith OM, Misteli T. Regulation of alternative splicing by histone modifications. Science. 2010;327: 996–1000. doi:10.1126/science.1184208
69. de la Mata M, Alonso CR, Kadener S, Fededa JP, Blaustein M, Pelisch F, et al. A slow RNA polymerase II affects alternative splicing in vivo. Mol Cell. 2003;12: 525–532. doi:10.1016/j.molcel.2003.08.001
70. Roberts GC, Gooding C, Mak HY, Proudfoot NJ, Smith CW. Co-transcriptional commitment to alternative splice site selection. Nucleic Acids Res. 1998;26: 5568–5572. doi:10.1093/nar/26.24.5568
71. Shukla S, Kavak E, Gregory M, Imashimizu M, Shutinoski B, Kashlev M, et al. CTCF-promoted RNA polymerase II pausing links DNA methylation to splicing. Nature. 2011;479: 74–79. doi:10.1038/nature10442

72. Andersson R, Enroth S, Rada-Iglesias A, Wadelius C, Komorowski J. Nucleosomes are well positioned in exons and carry characteristic histone modifications. Genome Res. 2009;19: 1732–1741. doi:10.1101/gr.092353.109

73. Hon G, Wang W, Ren B. Discovery and annotation of functional chromatin signatures in the human genome. PLoS Comput Biol. 2009;5: e1000566. doi:10.1371/journal.pcbi.1000566

74. Kolasinska-Zwierz P, Down T, Latorre I, Liu T, Liu XS, Ahringer J. Differential chromatin marking of introns and expressed exons by H3K36me3. Nat Genet. 2009;41: 376–381. doi:10.1038/ng.322

75. Nahkuri S, Taft RJ, Mattick JS. Nucleosomes are preferentially positioned at exons in somatic and sperm cells. Cell Cycle. 2009;8: 3420–3424. doi:10.4161/cc.8.20.9916

76. Schwartz S, Meshorer E, Ast G. Chromatin organization marks exon-intron structure. Nat Struct Mol Biol. 2009;16: 990–995. doi:10.1038/nsmb.1659

77. Spies N, Nielsen CB, Padgett RA, Burge CB. Biased chromatin signatures around polyadenylation sites and exons. Mol Cell. 2009;36: 245–254. doi:10.1016/j.molcel.2009.10.008

78. Iannone C, Pohl A, Papasaikas P, Soronellas D, Vicent GP, Beato M, et al. Relationship between nucleosome positioning and progesterone-induced alternative splicing in breast cancer cells. RNA. 2015;21: 360–374. doi:10.1261/rna.048843.114

79. Tilgner H, Nikolaou C, Althammer S, Sammeth M, Beato M, Valcárcel J, et al. Nucleosome positioning as a determinant of exon recognition. Nat Struct Mol Biol. 2009;16: 996–1001. doi:10.1038/nsmb.1658

80. Mendel M, Delaney K, Pandey RR, Chen K-M, Wenda JM, Vågbø CB, et al. Splice site m6A methylation prevents binding of U2AF35 to inhibit RNA splicing. Cell. 2021;184: 3125–3142.e25. doi:10.1016/j.cell.2021.03.062

81. Wang S, Lv W, Li T, Zhang S, Wang H, Li X, et al. Dynamic regulation and functions of mRNA m6A modification. Cancer Cell Int. 2022;22: 48. doi:10.1186/s12935-022-02452-x

82. Yu B, Yu X, Xiong J, Ma M. Methylation Modification, Alternative Splicing, and Noncoding RNA Play a Role in Cancer Metastasis through Epigenetic Regulation. Biomed Res Int. 2021;2021: 4061525. doi:10.1155/2021/4061525

83. Lin T-C, Tsai C-H, Shiau C-K, Huang J-H, Tsai H-K. Predicting splicing patterns from the transcription factor binding sites in the promoter with deep learning. bioRxiv. 2023. p. 2023.04.09.536141. doi:10.1101/2023.04.09.536141

84. García-Ruiz S, Zhang D, Gustavsson EK, Rocamora-Perez G, Grant-Peters M, Fairbrother-Browne A, et al. Splicing accuracy varies across human introns, tissues and age. bioRxiv. 2023. p. 2023.03.29.534370. doi:10.1101/2023.03.29.534370

85. Jorstad NL, Song JHT, Exposito-Alonso D, Suresh H, Castro-Pacheco N, Krienen FM, et al. Comparative transcriptomics reveals human-specific cortical features. Science. 2023;382: eade9516. doi:10.1126/science.ade9516

86. Karollus A, Mauermeier T, Gagneur J. Current sequence-based models capture gene expression determinants in promoters but mostly ignore distal enhancers. Genome Biol. 2023;24: 56. doi:10.1186/s13059-023-02899-9

87. Treutlein B, Gokce O, Quake SR, Südhof TC. Cartography of neurexin alternative splicing mapped by single-molecule long-read mRNA sequencing. Proc Natl Acad Sci U S A. 2014;111: E1291–9. doi:10.1073/pnas.1403244111

88. Schreiner D, Nguyen T-M, Russo G, Heber S, Patrignani A, Ahrné E, et al. Targeted combinatorial alternative splicing generates brain region-specific repertoires of neurexins. Neuron. 2014;84: 386–398. doi:10.1016/j.neuron.2014.09.011

89. Tilgner H, Jahanbani F, Gupta I, Collier P, Wei E, Rasmussen M, et al. Microfluidic isoform sequencing shows widespread splicing coordination in the human transcriptome. Genome Res. 2018;28: 231–242. doi:10.1101/gr.230516.117

90. Prjibelski AD, Mikheenko A, Joglekar A, Smetanin A, Jarroux J, Lapidus AL, et al. Accurate isoform discovery with IsoQuant using long reads. Nat Biotechnol. 2023;41: 915–918. doi:10.1038/s41587-022-01565-y

91. Falcon W, The PyTorch Lightning team. PyTorch Lightning. 2019. doi:10.5281/zenodo.3828935

92. Paszke A, Gross S, Massa F, Lerer A, Bradbury J, Chanan G, et al. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In: Wallach H, Larochelle H, Beygelzimer A, d\textquotesingle Alché-Buc F, Fox E, Garnett R, editors. Advances in Neural Information Processing Systems 32. Curran Associates, Inc.; 2019. pp. 8024–8035. Available: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf

93. Hao Y, Hao S, Andersen-Nissen E, Mauck WM, Zheng S, Butler A, et al. Integrated analysis of multimodal single-cell data. Cell. 2021;0. doi:10.1016/j.cell.2021.04.048

94. Frankish A, Diekhans M, Ferreira A-M, Johnson R, Jungreis I, Loveland J, et al. GENCODE reference annotation for the human and mouse genomes. Nucleic Acids Res. 2019;47: D766–D773. doi:10.1093/nar/gky955

95. Stein AN, Joglekar A, Poon C-L, Tilgner HU. ScisorWiz: visualizing differential isoform expression in single-cell long-read data. Bioinformatics. 2022;38: 3474–3476. doi:10.1093/bioinformatics/btac340

96. Benoit Bouvrette LP, Bovaird S, Blanchette M, Lécuyer E. oRNAment: a database of putative RNA binding protein target sites in the transcriptomes of model species. Nucleic Acids Res. 2020;48: D166–D173. doi:10.1093/nar/gkz986

97. Ray D, Kazan H, Cook KB, Weirauch MT, Najafabadi HS, Li X, et al. A compendium of RNA-binding motifs for decoding gene regulation. Nature. 2013;499: 172–177. doi:10.1038/nature12311

chapter 8

Discussion

Single-cell RNA sequencing (scRNA-seq) has massively increased our understanding of tissue compositions, cellular interactions, and developmental processes. Especially in heterogeneous tissues such as the brain, this single-cell resolution led to many newly discovered cell types, insights into the specificity of cell types for particular brain regions or layers, and the proportions of cell types across the brain [1–5]. Besides generating massive datasets, smaller publicly available datasets are combined into tissue-specific reference atlases, such as the Human Lung Cell Atlas [6]. However, analyzing individual datasets or creating these atlases is still mainly done using unsupervised methods.

In this thesis, we introduced several supervised methods to solve two broad tasks: 1) automatic cell-type identification in scRNA-seq data, and 2) understanding cell-type-specific (post-)transcriptional regulation. In part I, we benchmarked different cell-type classification methods for scRNA-seq data (chapter 2), developed scHPL (chapter 3) and treeArches (chapter 4) to automatically match cell types across datasets to construct a reference atlas with corresponding cellular hierarchy, and developed TACTiCS to match cell types across species (chapter 5). In part II, we showed how scRNA-seq with the corresponding cell-type labels can improve our understanding of transcriptional regulation (chapter 6) and alternative splicing (chapter 7) by developing cell-type-specific feature-prediction models. However, for both tasks, several challenges remain that we will discuss in the sections below.

8.1 What is a cell type?

In simple eukaryotic organisms, such as C. Elegans, every adult consists of the same amount of cells- 959 in hermaphrodites and 1031 in males [7,8]. This low and consistent number of cells allows researchers to study every cell individually. Studying more complex organisms, such as humans, similarly is challenging since we consist of approximately 37 trillion cells, and this number varies across individuals due to, for instance, differences in height [9]. Categorizing all these cells into cell types enhances our understanding of cells and facilitates effective communication and comparison of results across studies.

Is this discrete grouping that we use repeatedly throughout this thesis optimal, or would a continuous spectrum be beneficial? At least at a high level, cell types seem separate categories. For example, a muscle fiber differs from a neuron regarding its function, morphology, and the genes expressed. Still, both arise from the same stem cell and become continuously more specialized. At what stage during development would one consider these cells differentiated enough to call them different cell types?

Furthermore, due to perturbations, such as stimulations or pathogens, cells can transition to another cell type or state. Should these possible responses be considered in our definition as well [10]? In the pancreas, some alpha cells can, for instance, change into beta cells, which can occur naturally in persons with diabetes [11]. Also in the immune system, naive T-cells transition into memory cells after activation [12]. Both are considered different cell types with a gradient containing the transitioning cells in between.

Despite this evidence for a more continuous spectrum, we still focus on cell-type classification since most downstream methods require cells from the same cell type or cell-type labels as input. This downstream analysis can be a relatively simple task, such as testing for differentially expressed genes between healthy and diseased cells of the same cell type. But for more complex tasks, such as detecting expression quantitative trait loci (eQTLs), the cell-type labels may be beneficial as well. A cell-type-specific eQTL analysis can reveal the effect of variants that were previously hidden when analyzing the complete sample [13]. Also in Chapters 6 and 7, we rely on the cell-type labels to improve our understanding of transcriptional regulation and alternative splicing. Especially for heterogeneous tissues, using this increased cell-type-specific resolution improved the performance.

A potential alternative could be to redevelop current downstream methods such that they produce similar results, but do not rely on cell-type labels. An example is Milo, which tests for differential abundance between two samples [14]. First, cells are assigned to neighborhoods and afterwards, Milo tests whether cells from a certain condition are enriched or depleted within each neighborhood. Cell-type labels are unneeded during this analysis and will thus not bias the results. For the sequence-based models, this problem could be overcome by predicting the features at the cell instead of cell-type resolution as is done by scBasset [15] and seq2cells [16]. As datasets grow bigger and bigger, this might become computationally too expensive at some point. However, the results of cell-type-agnostic methods might be harder to interpret. As a solution, cells could be aggregated into cell types again solely for interpretation. Then, at least the cell-type labels do not bias the analysis itself.

Since cells exist in a continuous spectrum, a second alternative is moving from binary to fuzzy cell-type labels. Using fuzzy labels, a cell can belong to multiple cell types simultaneously with different probabilities. A probability above zero for two cell types can indicate that a cell is transitioning between these two. For scRNA-seq data, this approach has been explored for clustering methods [17,18], but not yet for classification methods. During classification, the posterior probability could easily indicate which cell types a cell belongs to.

8.2 Consistent cell-type classification

Since most downstream methods rely on discrete cell-type labels, cells must be labeled consistently to enable combining or comparing information from different datasets. For instance, the sc-eQTL consortium aims to find how variants affect gene expression in immune cell types by combining datasets from multiple labs containing hundreds of individuals [19]. In every individual, the cell types should thus be defined similarly. A high precision in cell-type annotations might be even more important than a high accuracy. Since unsupervised methods are subjective and time-consuming, an automatic supervised approach is needed here.

Ideally, such a classifier is trained on a reference atlas that combines data from enough individuals so that inter-individual variation and rare cell types are captured. The cell types in such a reference atlas should not be characterized as in a periodic table but in a hierarchy [20]. A hierarchical classifier divides the classification problem into smaller subproblems which improves the classification performance. We showed that a hierarchical linear SVM outperforms a flat linear SVM in Chapter 3. Besides, when using a hierarchical classifier, users can easily choose their resolution of interest. Using Azimuth [21], an easy-to-use web portal, cells can also be annotated at different resolutions. However, these resolutions are not connected in a hierarchy. Consequently, a cell can, for instance, be labeled as a CD8+ T-cell and CD4+ memory T-cell, which is impossible and therefore inconsistent.

Reference atlases exist for many human and mouse tissues and can be downloaded from platforms, such as Azimuth [21] or CELLxGENE [22,23]. For these reference atlases, either 1) one big dataset is used (e.g. the human PBMC reference containing eight individuals [21]), 2) multiple datasets are combined and re-annotated manually (e.g. the human lung cell atlas containing 107 individuals from 14 datasets [6]), or 3) multiple annotated datasets are combined using scHPL and their labels are manually refined (e.g. the mouse kidney atlas combining data from 59 mice from 8 datasets [24]).

However, many datasets are still annotated using unsupervised methods even though a reference atlas for that specific tissue is available [25–27]. Why is this the case? Researchers might not trust supervised methods since their performance is not perfect yet. In Chapter 2, however, we showed that cell-type classification is a relatively easy problem at a low resolution since almost all methods perform (nearly) perfectly. The performance of most methods drops when increasing the resolution or complexity of the data. For most reference atlases, however, the performance is not benchmarked per resolution, making it hard to know how consistent label transfer will be.

Another complicating factor is the batch effects between the reference atlas and the unlabeled dataset. Batch effects are technical variations between datasets due to variations in labs, protocols, sequencing depths, etc. This technical variation has to be removed while preserving the biological variation. This is a complex problem since the effects are usually non-linear and the ground truth is unknown. Benchmark studies showed that methods including scVI [28] and Harmony [29] perform well for this task. For most methods, however, parameters have to be tuned for optimal performance, which might decrease the usability.

Interestingly, researchers are imperfect when annotating a scRNA-seq dataset manually as well. In Chapter 3, we applied scHPL to multiple annotated PBMC datasets, which resulted in a hierarchy with unexpected edges. Visualizing marker genes in the individual datasets indicated that cells had been wrongly annotated in the original datasets. Amongst others, the authors had swapped two cell-type labels, which explained the incorrect hierarchy. We experienced that scHPL is a great tool for discovering such misannotations. Cells can be relabeled based on this unexpected hierarchy.

Besides being subjective and time-consuming, another problem with manual annotation is a missing naming convention for cell types. CELLxGENE resolves this problem by forcing users to use Cell Ontology terminology when uploading their datasets. A downside of the Cell Ontology is that this hierarchy only consists of names but lacks information about the cell type, such as its function, morphology, or transcriptomic profile. Consequently, cell types from different datasets with the same name could have a different underlying expression pattern. The most straightforward solution might seem to add marker genes to Cell Ontology, which can be used to identify cell types. In the benchmark in Chapter 2, however, we noticed that methods relying on marker genes perform worse during cell-type identification, most likely because of the sparsity of scRNA-seq data.

Ideally, all datasets from a similar tissue in CELLxGENE are not harmonized based on the names but based on the expression profile in a data-driven way using tools such as scHPL and treeArches. These tools can be enhanced by reflecting (inter-individual) variation in the width of a branch and allowing for fuzzy labels at the leaf nodes. The growing amount of data poses a challenge and as such both methods must become computationally more efficient. The resulting reference atlases should be updated continuously with newly generated data.

8.3 Automatically detecting new cell types

Even though many reference atlases are being constructed [6,21,24], these will never be complete since rare and diseased cell types might be missing. In the human lung cell atlas, for instance, six rare cell types were not defined in any of the individual datasets and had not been defined in the lung before, but could be discovered when combining multiple datasets [6]. Besides, new viruses, such as SARS-Cov-2, can infect cells from different tissues and perturb these cells [30,31]. Identifying such diseased cell types is important for drug or therapy development. Adding such data to a reference atlas leads to new insights in both healthy and diseased samples.

To detect rare or diseased cell types automatically, a classifier needs a rejection option. In Chapter 2, we benchmarked the rejection options of scRNA-seq cell-type identification methods by removing a cell type completely from the data. Here, we noticed that the linear SVM, which had the highest classification performance, performed poorly since it relied on the posterior probability. In Chapters 3 and 4, we introduced scHPL and improved the rejection option by incorporating distance metrics. This improved the detection of unknown cells but still did not perform perfectly. Diseased cells, such as inflamed monocyte-derived macrophages, are immediately rejected (labeled “unknown”) instead of labeled as internal node (e.g. macrophages), which would be preferred.

A hierarchical classifier that can return internal nodes of the hierarchy, so-called “partial rejection”, is beneficial according to a recent benchmark [32]. Here, they only evaluated how a full or partial rejection option affected the classification performance and not whether new cell types could be detected. Detecting new cell types using reference atlases should be benchmarked properly in upcoming benchmarks. An example experiment would be to remove one cell type from the training data and test whether the classifier correctly rejects cells from that cell type in the test dataset.

Ideally, cell-type identification and data integration methods should be benchmarked simultaneously. Data integration considerably influences whether these new cell types can be detected. During data integration biological variation should be preserved and technical variation should be removed. If the difference between a diseased and healthy cell type of two samples is seen as a technical artifact, this difference can be removed as well. Regardless of the cell-type identification method used afterwards, the cell type will never be detected as a new cell type. Ideally, a diseased and healthy sample are sequenced together, so there are no batch effects. As such the difference between biological and technical variations between the reference atlas and these new samples can be detected more easily [33].

8.4 Towards cell-type-specific sequence-based models

Studying tissues untargeted and at a high resolution using scRNA-seq has led to the discovery of many new cell types. Since these cell types are defined based on their transcriptional profile, the underlying transcriptional regulation must be unique for every cell type. In Chapter 6, we aimed to unravel these cell-type-specific mechanisms by training sequence-based models using scRNA-seq data with the corresponding cell-type labels to predict gene expression. In Chapter 7, we focused on alternative splicing mechanisms by training models to predict cell-type-specific exon inclusion in the brain. Interpreting which motifs guide the model to make certain predictions, increases our understanding of the biological mechanisms underlying transcriptional regulation and alternative splicing.

Furthermore, these models aid in understanding how variants affect a cell type. Approximately 95% of the GWAS variants fall in non-coding regions [34]. Usually, only an association between a group of variants and a trait is discovered, but it remains unclear which variant causes a trait due to linkage disequilibrium, through which mechanism a variant acts, and which cell type is most disrupted. Models that use the genome to predict, for instance, transcription or splicing in a cell-type-specific way can address these problems.

In Chapter 6, we showed that cell-type-specific models always outperformed the tissue-specific models when predicting cell-type-specific gene expression levels. The difference in performance becomes most apparent if a tissue and cell type are dissimilar. Even though this increase was significant, we were unable to pinpoint what caused this increase such as cell-type-specific transcription factor binding sites.

To reliably predict the cell-type-specific effect of variants, our models, as well as other state-of-the-art sequence-based models, such as Enformer [35] and SpliceAI [36], have to overcome several limitations: 1) missing cell-type-specificity, 2) ignoring distal regulatory elements, and 3) incorrectly predicting personalized gene expression. I will discuss these limitations and potential solutions in the coming sections.

8.5 Missing cell-type-specificity of sequence-based models

The cell-type-specificity or tissue-specificity of sequence-based models is not thoroughly evaluated. Enformer is trained on 5,313 genomic tracks including different tissues and measurement techniques such as CAGE and DNase-seq reads, and predicts different values for very dissimilar cell types, such as keratinocytes and monocytes. However, an evaluation for more similar tracks, such as 77 CAGE tracks related to the brain, is missing. We noticed the same for Pangolin [37], a model to predict tissue-specific splicing. Pangolin outperformed

SpliceAI, the tissue-agnostic model, but no tissue-specific regulatory elements were discussed. Ideally, the models should be evaluated using cell-type-specific variants, but the ground truth for most variants is missing. A missing ground truth makes proper benchmarking impossible. A feasible alternative is to evaluate the models' performance on marker genes for specific cell types or whether the models correctly learn in which cell type a gene is higher expressed using for instance the log-fold change or the difference between two cell types.

Exploiting the current models as pre-trained models could be beneficial for learning cell-type-specific mechanisms. Some cell types are so similar that it is challenging to train a complete model with millions of parameters from scratch to learn these subtle differences. Seq2cells, for instance, extracts an embedding from Enformer and trains a simple model, a multi-layer perceptron, to predict the cell-type-specific gene expression [16]. Seq2cells assumes that all regulatory features are stored in the embeddings and the simpler model only needs to learn how to combine these during the fine-tuning step.

8.6 Limited context of sequence-based models

In our models, the region around the transcription start site and splice sites contributed most to the predictions of gene expression and exon inclusion. These regions are most important for transcription and splicing since RNA polymerase and the spliceosome bind there respectively. However, this signal dominates the predictions entirely, and as such the predicted effect of mutations further away is negligible. While mutations in enhancers far away or deep intronic variants can cause a disease [38–40]. A recent benchmark showed that other models do not capture distal regulatory elements either [41]. Even though Enformer inputs a sequence of 196kb, it incorrectly predicts the effect of variants in distal regulatory elements.

For splicing models, this has yet to be investigated, but since the model architectures and training strategies are similar, we can assume the models suffer here as well. Interestingly, SpliceAI, which inputs 10kb around the splice sites, was recently outperformed by Splam [42], a model that only uses 400 bp, indicating that regions further away might not be needed to predict splicing accurately. However, SpliceAI and Splam are both classification methods that predict whether a certain site is a splice junction instead of how often the junction is used. Distal variants may affect the latter more.

8.7 Sequence-based models are data-hungry

Current sequence-based models still suffer from limited training data. For instance, only a few genes are cell-type-specific or regulated by distal regulatory elements. Few examples in the training data make it difficult for models to learn the patterns. However, the number of genes or exons in the human genome limits the size of the training data, so this cannot be easily increased. To overcome this, several models, including Enformer, are trained on human and mouse data simultaneously to increase the size of the training data [35,43]. The weights of the first layers in the model are shared across the species exploiting that regulatory elements are partially conserved. The final fully connected layer is species-specific to allow learning of species-specific mechanisms as well. In Chapter 7, we applied this trick when training exon-inclusion models, which improved their performance. In general, the current models only combine human and mouse, while data from more closely related species is available. For instance, for a cell-type-specific model predicting gene expression in the brain, scRNA-seq data from five primates could be combined [44].

8.8 Personalized sequence-based models

The third limitation is that current sequence-based models cannot predict variation of gene expression across individuals yet [45,46]. Ideally, for every individual genome, these models would predict the correct expression level, i.e. making personalized predictions. However, when evaluating models using variants found across individual genomes, Enformer predicted the wrong direction of effect for one-third of the tested variants. We did not evaluate making personalized predictions in our models, but since our models rely on models that were evaluated in the benchmark, we assume they incorrectly predict this as well.

State-of-the-art expression and splicing prediction models are all trained on the reference genome. However, the predicted genomic features were measured in individuals with specific variants in their genomes. Recent benchmarks suggested that training on individual genomes could improve personalized gene expression predictions [45,46]. Training on individual genomes might enhance learning of the effect of distal regulatory elements as well because of the increased variance in the training data. Recently, BigRNA [47] was released which predicts gene expression in 51 tissues for 70 individuals. For each individual, both haplotypes are input to identical instances of the model and the output is combined. Their results look promising, but the personalized gene expression task has not been evaluated for this model yet.

8.9 What should sequence-based models predict?

One might also question whether predicting gene expression or exon inclusion directly from the sequence is the most optimal approach to reach the goal of predicting the effect of mutations. Measurement techniques are noisy and the measured gene expression does not directly reflect how often a gene is transcribed in a cell. A gene can be highly transcribed but rapidly degraded as well due to (aberrant) splicing isoforms. Also in healthy tissues or cell types, alternative splicing is a way to control gene expression levels [48,49]. If the inclusion of an exon activates nonsense-mediated decay, this exon might not be measured or only in low levels even though it was originally highly included. An alternative would be to train the models on RNA-sequencing data of samples where nonsense-mediated decay was blocked, but this data is scarce.

Instead of predicting gene expression directly, it might be beneficial to predict intermediate layers, such as chromatin accessibility. Models trained to predict cell-type-specific chromatin accessibility in the drosophila brain [50] or for human melanoma [51] could be used to design cell-type-specific enhancers [52]. These models are not limited by the number of genes in the genome but are trained on differentially accessible regions between cell types.

This increased the size of the training data and might explain the cell-type-specificity of the models. However, the designed enhancers are only 500 bp. The effect of these enhancers was tested using a luciferase assay which means that these enhancers are inserted before the transcription start site of the luciferase gene. The effect of distal enhancers is thus not tested during the design. ExPecto [53] and their recent successor ExPectoSC [54] try to overcome this by first predicting 2002 regulatory features for the 40kb region around the transcription start site and using this to train a simpler model to predict gene expression.

An alternative could be to input chromatin accessibility measurements, or similar regulatory features, to the models [55]. This improves the cell-type-specificity since the input data is different now for every cell type or tissue. Another advantage is that these models can extrapolate to new cell types as long as chromatin accessibility data is available for that cell type. Evaluating the effect of variants or model interpretation becomes more complicated though since the input sequence cannot be *in-silico* mutated anymore as it is unknown how a mutation will affect the chromatin accessibility input track.

ENCODE-rE2G [56] combines a cell-type-specific input with an interesting training strategy: instead of training on healthy data, the model is trained on perturbation data. This logistic regressor predicts whether an element, a part of the DNA sequence, regulates a gene based on extracted features from the cell-type-specific DNase and cell-type-agnostic features, such as the distance between the element and the gene of interest. Since the model learns the relation between an element and the gene, it is not biased towards features close to the transcription start site and learns distal regulatory elements as well. However, they assume that a variant that falls in an element is always linked to the gene, and the direction of effect is not predicted. Instead of using the extracted features, a sequence-based model with a similar training strategy might be beneficial here.

8.10 Final remarks

Single-cell RNA sequencing has revolutionized our understanding of heterogeneous tissues. In this thesis, we presented several methods to automatically identify cell types in scRNA-seq data and use scRNA-seq data to increase the resolution of current sequence-based models. However, when analyzing scRNA-seq data, or using this data to train sequence-based models, we should remember that cells or cell types are not isolated compartments, but that they interact and communicate with each other. Many spatial transcriptomics datasets are now generated to focus on this. Ideally, we integrate this spatial information into the sequence-based models.

Not only do neighboring cells influence which genes are expressed, but the expression of other genes in a cell can influence the gene of interest as well. A more holistic view might be needed instead of predicting the expression of one gene at a time. Also when predicting splicing, we know that exons are very often coordinated. Using a different transcription start site might determine the complete isoform used. Predicting the inclusion of individual exons might be very difficult or near impossible in such a case. Ideally, sequence-based models would predict the expression of multiple isoforms simultaneously in the future.

Bibliography

1. Zeisel A, Hochgerner H, Lönnerberg P, Johnsson A, Memic F, van der Zwan J, et al. Molecular Architecture of the Mouse Nervous System. Cell. 2018;174: 999–1014.e22. doi:10.1016/j.cell.2018.06.021
2. Hodge RD, Bakken TE, Miller JA, Smith KA, Barkan ER, Graybuck LT, et al. Conserved cell types with divergent features in human versus mouse cortex. Nature. 2019; 1–8. doi:10.1038/s41586-019-1506-7
3. Saunders A, Macosko EZ, Wysoker A, Goldman M, Krienen FM, de Rivera H, et al. Molecular Diversity and Specializations among the Cells of the Adult Mouse Brain. Cell. 2018;174: 1015–1030.e16. doi:10.1016/j.cell.2018.07.028
4. Bakken TE, Jorstad NL, Hu Q, Lake BB, Tian W, Kalmbach BE, et al. Comparative cellular analysis of motor cortex in human, marmoset and mouse. Nature. 2021;598: 111–119. doi:10.1038/s41586-021-03465-8
5. Siletti K, Hodge R, Mossi Albiach A, Lee KW, Ding S-L, Hu L, et al. Transcriptomic diversity of cell types across the adult human brain. Science. 2023;382: eadd7046. doi:10.1126/science.add7046
6. Sikkema L, Ramírez-Suástegui C, Strobl DC, Gillett TE, Zappia L, Madissoon E, et al. An integrated cell atlas of the lung in health and disease. Nat Med. 2023;29: 1563–1577. doi:10.1038/s41591-023-02327-2
7. Kimble J, Hirsh D. The postembryonic cell lineages of the hermaphrodite and male gonads in Caenorhabditis elegans. Dev Biol. 1979;70: 396–417. doi:10.1016/0012-1606(79)90035-6
8. Sulston JE, Horvitz HR. Post-embryonic cell lineages of the nematode, Caenorhabditis elegans. Dev Biol. 1977;56: 110–156. doi:10.1016/0012-1606(77)90158-0
9. Bianconi E, Piovesan A, Facchin F, Beraudi A, Casadei R, Frabetti F, et al. An estimation of the number of cells in the human body. Ann Hum Biol. 2013;40: 463–471. doi:10.3109/03014460.2013.807878
10. Fleck JS, Camp JG, Treutlein B. What is a cell type? Science. 2023;381: 733–734. doi:10.1126/science.adf6162
11. Chakravarthy H, Gu X, Enge M, Dai X, Wang Y, Damond N, et al. Converting Adult Pancreatic Islet α Cells into β Cells by Targeting Both Dnmt1 and Arx. Cell Metab. 2017;25: 622–634. doi:10.1016/j.cmet.2017.01.009
12. Raphael I, Joern RR, Forsthuber TG. Memory CD4+ T Cells in Immunity and Autoimmune Diseases. Cells. 2020;9. doi:10.3390/cells9030531
13. Van Der Wijst MGP, Brugge H, De Vries DH, Deelen P, Swertz MA, Franke L. Single-cell RNA sequencing identifies celltype-specific cis-eQTLs and co-expression QTLs. Nat Genet. 2018;50: 493–497. doi:10.1038/s41588-018-0089-9
14. Dann E, Henderson NC, Teichmann SA, Morgan MD, Marioni JC. Differential abundance testing on single-cell data using k-nearest neighbor graphs. Nat Biotechnol. 2022;40: 245–253. doi:10.1038/s41587-021-01033-z
15. Yuan H, Kelley DR. scBasset: sequence-based modeling of single-cell ATAC-seq using convolutional neural networks. Nat Methods. 2022;19: 1088–1096. doi:10.1038/s41592-022-01562-8
16. Schwessinger R, Deasy J, Woodruff RT, Young S, Branson KM. Single-cell gene expression prediction from DNA sequence at large contexts. bioRxiv. 2023. p. 2023.07.26.550634. doi:10.1101/2023.07.26.550634
17. Mallik S, Zhao Z. Multi-Objective Optimized Fuzzy Clustering for Detecting Cell Clusters from Single-Cell Expression Profiles. Genes . 2019;10. doi:10.3390/genes10080611
18. Wang J, Xia J, Tan D, Lin R, Su Y, Zheng C-H. scHFC: a hybrid fuzzy clustering method for single-cell RNA-seq data optimized by natural computation. Brief Bioinform. 2022;23. doi:10.1093/bib/bbab588
19. van der Wijst MG, de Vries DH, Groot HE, Trynka G, Hon C-C, Bonder M-J, et al. The single-cell eQTLGen consortium. Elife. 2020;9. doi:10.7554/eLife.52155
20. Domcke S, Shendure J. A reference cell tree will serve science better than a reference cell atlas. Cell. 2023;186: 1103–1114. doi:10.1016/j.cell.2023.02.016
21. Hao Y, Hao S, Andersen-Nissen E, Mauck WM, Zheng S, Butler A, et al. Integrated analysis of multimodal single-cell data. Cell. 2021;0. doi:10.1016/j.cell.2021.04.048
22. Megill C, Martin B, Weaver C, Bell S, Prins L, Badajoz S, et al. cellxgene: a performant, scalable exploration platform for high dimensional sparse matrices. bioRxiv. 2021. p. 2021.04.05.438318. doi:10.1101/2021.04.05.438318
23. CZI Single-Cell Biology Program, Abdulla S, Aevermann B, Assis P, Badajoz S, Bell SM, et al. CZ CELL×GENE Discover: A single-cell data platform for scalable exploration, analysis and modeling of aggregated data. bioRxiv. 2023. p. 2023.10.30.563174. doi:10.1101/2023.10.30.563174
24. Novella-Rausell C, Grudniewska M, Peters DJM, Mahfouz A. A comprehensive mouse kidney atlas enables rare cell population characterization and robust marker discovery. iScience. 2023;26: 106877. doi:10.1016/j.isci.2023.106877
25. Yazar S, Alquicira-Hernandez J, Wing K, Senabouth A, Gordon MG, Andersen S, et al. Single-cell eQTL mapping identifies cell type-specific genetic control of autoimmune disease. Science. 2022;376: eabf3041. doi:10.1126/science.abf3041
26. Sirkis DW, Warly Solsberg C, Johnson TP, Bonham LW, Sturm VE, Lee SE, et al. Single-cell RNA-seq reveals alterations in peripheral CX3CR1 and nonclassical monocytes in familial tauopathy. Genome Med. 2023;15: 53. doi:10.1186/s13073-023-01205-3
27. Zhu H, Chen J, Liu K, Gao L, Wu H, Ma L, et al. Human PBMC scRNA-seq-based aging clocks reveal ribosome to inflammation balance as a single-cell aging hallmark and super longevity. Sci Adv. 2023;9: eabq7599. doi:10.1126/sciadv.abq7599
28. Xu C, Lopez R, Mehlman E, Regier J, Jordan MI, Yosef N. Probabilistic harmonization and annotation of single-cell transcriptomics data with deep generative models. Mol Syst Biol. 2021;17: e9620. doi:10.15252/msb.20209620

29. Korsunsky I, Millard N, Fan J, Slowikowski K, Zhang F, Wei K, et al. Fast, sensitive and accurate integration of single-cell data with Harmony. Nat Methods. 2019;16: 1289–1296. doi:10.1038/s41592-019-0619-0

30. Liao M, Liu Y, Yuan J, Wen Y, Xu G, Zhao J, et al. Single-cell landscape of bronchoalveolar immune cells in patients with COVID-19. Nat Med. 2020;26: 842–844. doi:10.1038/s41591-020-0901-9

31. Lotfollahi M, Naghipourfar M, Luecken MD, Khajavi M, Büttner M, Wagenstetter M, et al. Mapping single-cell data to reference atlases by transfer learning. Nat Biotechnol. 2022;40: 121–130. doi:10.1038/s41587-021-01001-7

32. Theunissen L, Mortier T, Saeys Y, Waegeman W. Uncertainty-aware single-cell annotation with a hierarchical reject option. bioRxiv. 2023. p. 2023.09.25.559294. doi:10.1101/2023.09.25.559294

33. Dann E, Cujba A-M, Oliver AJ, Meyer KB, Teichmann SA, Marioni JC. Precise identification of cell states altered in disease using healthy single-cell references. Nat Genet. 2023;55: 1998–2008. doi:10.1038/s41588-023-01523-7

34. Leslie R, O'Donnell CJ, Johnson AD. GRASP: analysis of genotype-phenotype results from 1390 genome-wide association studies and corresponding open access database. Bioinformatics. 2014;30: i185–94. doi:10.1093/bioinformatics/btu273

35. Avsec Ž, Agarwal V, Visentin D, Ledsam JR, Grabska-Barwinska A, Taylor KR, et al. Effective gene expression prediction from sequence by integrating long-range interactions. Nat Methods. 2021;18: 1196–1203. doi:10.1038/s41592-021-01252-x

36. Jaganathan K, Kyriazopoulou Panagiotopoulou S, McRae JF, Darbandi SF, Knowles D, Li YI, et al. Predicting Splicing from Primary Sequence with Deep Learning. Cell. 2019;176: 535–548.e24. doi:10.1016/j.cell.2018.12.015

37. Zeng T, Li YI. Predicting RNA splicing from DNA sequence using Pangolin. Genome Biol. 2022;23: 103. doi:10.1186/s13059-022-02664-4

38. Vaz-Drago R, Custódio N, Carmo-Fonseca M. Deep intronic mutations and human disease. Hum Genet. 2017;136: 1093–1111. doi:10.1007/s00439-017-1809-4

39. Smemo S, Campos LC, Moskowitz IP, Krieger JE, Pereira AC, Nobrega MA. Regulatory variation in a TBX5 enhancer leads to isolated congenital heart disease. Hum Mol Genet. 2012;21: 3255–3263. doi:10.1093/hmg/dds165

40. Claringbould A, Zaugg JB. Enhancers in disease: molecular basis and emerging treatment strategies. Trends Mol Med. 2021;27: 1060–1073. doi:10.1016/j.molmed.2021.07.012

41. Karollus A, Mauermeier T, Gagneur J. Current sequence-based models capture gene expression determinants in promoters but mostly ignore distal enhancers. Genome Biol. 2023;24: 56. doi:10.1186/s13059-023-02899-9

42. Chao K-H, Mao A, Salzberg SL, Pertea M. Splam: a deep-learning-based splice site predictor that improves spliced alignments. bioRxiv. 2023. p. 2023.07.27.550754. doi:10.1101/2023.07.27.550754

43. Kelley DR. Cross-species regulatory sequence activity prediction. Ma J, editor. PLoS Comput Biol. 2020;16: e1008050. doi:10.1371/journal.pcbi.1008050

44. Jorstad NL, Song JHT, Exposito-Alonso D, Suresh H, Castro-Pacheco N, Krienen FM, et al. Comparative transcriptomics reveals human-specific cortical features. Science. 2023;382: eade9516. doi:10.1126/science.ade9516

45. Huang C, Shuai RW, Baokar P, Chung R, Rastogi R, Kathail P, et al. Personal transcriptome variation is poorly explained by current genomic deep learning models. Nat Genet. 2023;55: 2056–2059. doi:10.1038/s41588-023-01574-w

46. Sasse A, Ng B, Spiro AE, Tasaki S, Bennett DA, Gaiteri C, et al. Benchmarking of deep neural networks for predicting personal gene expression from DNA sequence highlights shortcomings. Nat Genet. 2023;55: 2060–2064. doi:10.1038/s41588-023-01524-6

47. Celaj A, Gao AJ, Lau TTY, Holgersen EM, Lo A, Lodaya V, et al. An RNA foundation model enables discovery of disease mechanisms and candidate therapeutics. bioRxiv. 2023. p. 2023.09.20.558508. doi:10.1101/2023.09.20.558508

48. Zheng S. Alternative splicing and nonsense-mediated mRNA decay enforce neural specific gene expression. Int J Dev Neurosci. 2016;55: 102–108. doi:10.1016/j.ijdevneu.2016.03.003

49. Nickless A, Bailis JM, You Z. Control of gene expression through the nonsense-mediated RNA decay pathway. Cell Biosci. 2017;7: 26. doi:10.1186/s13578-017-0153-7

50. Janssens J, Aibar S, Taskiran II, Ismail JN, Gomez AE, Aughey G, et al. Decoding gene regulation in the fly brain. Nature. 2022;601: 630–636. doi:10.1038/s41586-021-04262-z

51. Atak ZK, Taskiran II, Demeulemeester J, Flerin C, Mauduit D, Minnoye L, et al. Interpretation of allele-specific chromatin accessibility using cell state-aware deep learning. Genome Res. 2021; gr.260851.120. doi:10.1101/gr.260851.120

52. Taskiran II, Spanier KI, Dickmänken H, Kempynck N, Pančíková A, Ekşi EC, et al. Cell type directed design of synthetic enhancers. Nature. 2023. doi:10.1038/s41586-023-06936-2

53. Zhou J, Theesfeld CL, Yao K, Chen KM, Wong AK, Troyanskaya OG. Deep learning sequence-based ab initio prediction of variant effects on expression and disease risk. Nat Genet. 2018;50: 1171–1179. doi:10.1038/s41588-018-0160-6

54. Sokolova K, Theesfeld CL, Wong AK, Zhang Z, Dolinski K, Troyanskaya OG. Atlas of primary cell-type-specific sequence models of gene expression and variant effects. Cell Rep Methods. 2023;3: 100580. doi:10.1016/j.crmeth.2023.100580

55. Zhang Z, Feng F, Qiu Y, Liu J. A generalizable framework to comprehensively predict epigenome, chromatin organization, and transcriptome. Nucleic Acids Res. 2023;51: 5931–5947. doi:10.1093/nar/gkad436

56. Gschwind AR, Mualim KS, Karbalayghareh A, Sheth MU, Dey KK, Jagoda E, et al. An encyclopedia of enhancer-gene regulatory interactions in the human genome. bioRxiv. 2023. p. 2023.11.09.563812. doi:10.1101/2023.11.09.563812

ACKNOWLEDGEMENTS

The past four years of my PhD have been a great journey. I had the privilege of meeting many inspiring individuals, traveling to beautiful places, and embracing different cultures and their cuisines. It was sometimes challenging, but the encouragement of family, friends, and colleagues lifted me through these moments of doubt and propelled me forward. Thanks to everyone around me, I could grow not only as a scientist but also as a person. I will be forever grateful for all your support.

First, I thank my promoters: Ahmed, Marcel, and Boudewijn. Ahmed, thank you for your enthusiasm, being approachable, and just going for a coffee during the pandemic. All the (scientific) discussions we had, but also your professional and personal advice, made me grow, and I will truly miss this. Marcel, your always critical questions and insanely good feedback raised the level of all my work. It is inspiring to see how you lead the DBL group with such a friendly atmosphere where everyone can flourish. Boudewijn, thank you for allowing me to do this PhD and for your constant positive attitude.

Tamim, working on the benchmark together went so smoothly that it made me spoiled and naive about how collaborations and the whole publication process go. I enjoyed how we continued discussing each other's work. I hope we can stay in touch! Laura, you’re the best PhD buddy someone can wish for. Thanks for all the adventures during conferences, for all the good chats, and for answering all my wet lab questions. Yasin, you connect everyone in the DBL group with your enthusiasm and energy. Thank you for all the 3 o’clock good coffee breaks. Gerard, thanks for all the brainstorming sessions and endless sharing of all your ideas. Kirti, I am still impressed by how you made TACTiCS work. Inez, I really enjoyed working on your project together. Paul, thanks for all the ‘gezelligheid’ you brought to the group. Gabriel, you’re the best marathon supporter. Sander, all your biking is impressive and inspiring. Mostafa, thank you for introducing me to vegetarian Egyptian food. Mo, you have a great sense of humor, please stay like that.

The office in Leiden, Qirong, Mikhael, Benedetta, Claudio, and Alireza, I like how we made it more and more our own spot and all the delicious snacks everyone brought. Please keep my plant alive. There is not enough space to thank all the other DBL and Mahfouz lab members. Learning about your research and exchanging exciting highlights and challenging moments was always fascinating. Thank you for providing such a welcoming atmosphere.

Hagen, thank you for inviting me to your lab in NYC. I enjoyed all the discussions we had, they opened up a new way of looking at science for me. Justine, thanks for exploring NYC together and collaborating on the project. Anoushka, you’re inspiring to work with and your enthusiasm is contagious. The rest of the lab, thanks for making me feel at home. These three months were probably the best of my PhD, hope to see you soon!

The YoungCB board, Kathi, Swier, Liting, Mostafa, and Architha, thank you for all the nice lunch meetings and organizing all the YoungCB events together.

De liefste vrienden, Pauline, Annabel, Jill, Nina, Charlotte, Janne, Silke, Lieke, Emma, Lisa, Nina, Charlotte, bedankt voor alle kopjes thee, het samen sporten, en de vele spelletjes, zodat ik mijn hoofd weer even leeg kon maken. M'n huisgenoten van de BT, Simone, Paulina, Laura, Lisa, ik had me geen betere huisgenoten kunnen voorstellen tijdens de lockdown en de eerste jaren van m'n PhD.

Jacqueline & Ron, er zijn weinig plekken waar ik zo tot rust kom als bij jullie thuis. Bedankt daarvoor en voor alle waardevolle gesprekken die we altijd hebben.

Janneke & Brechje, jullie zijn de beste zussen die ik me kan wensen. Brechje, bedankt voor alle belletjes toen ik in NY zat, maar kom de volgende keer maar wel gewoon langs. Janneke, ik weet niet waar ik nu gestaan had zonder alle goede gesprekken tijdens onze hardloop- en fietsrondjes. Pieter, binnenkort ga ik echt alle boeken lezen die je hebt aangeraden. Jan, je bent niet alleen de beste fietsenmaker die ik ken, maar je voelt praktisch als een broer voor me.

Mama & papa, ik weet niet hoe ik mijn dankbaarheid kan uitspreken voor jullie onverwaardelijke steun. Bedankt voor jullie geloof in mij en alle mooie levenslessen.

M'n steun en toeverlaat, Jimmy. Bedankt dat je er altijd voor me bent. Jij laat me meer genieten van het leven en weet me altijd gerust te stellen als ik weer eens te serieus of gestresst ben. Op naar het volgende avontuur!

CURRICULUM VITAE

Feb 28, 1996	Born in 's Hertogenbosch, the Netherlands

Education

2007-2013	Pre-university education, Gymnasium Juvenaat, Bergen op Zoom, the Netherlands
2013-2016	BSc in Nanobiology, Delft University of Technology & Erasmus University Rotterdam, the Netherlands
2017-2020	MSc in Computer Science - Specialization Bioinformatics, Delft University of Technology, the Netherlands
2020-2024	PhD in Bioinformatics, Leiden University Medical Center, the Netherlands

Professional experiences

2016-2017	Intern, ZF-Screens, Leiden, the Netherlands
2018-2019	Intern, Future Genomics Technologies, Leiden, the Netherlands

Awards & fellowships

2020	KHMW Young Talent Award, Pfizer prize for the Life Sciences (3rd prize)
2022	EMBO Scientific Exchange Grant
2022	KNAW van Leersum Grant

LIST OF PUBLICATIONS

L. Michielsen, J. Hsu, A. Joglekar, N. Belchikov, M.J.T. Reinders, H. Tilgner, A. Mahfouz, "Predicting cell-type-specific exon inclusion in the human brain reveals more complex splicing mechanisms in neurons than glia", *bioRxiv* (2024)

L. Michielsen, M.J.T. Reinders, A. Mahfouz, "Predicting cell population-specific gene expression from genomic sequence", *Frontiers in Bioinformatics* (2024)

L. Michielsen, M. Lotfollahi, D. Strobl, L. Sikkema, M.J.T. Reinders, F.J. Theis, A. Mahfouz, "Single-cell reference mapping to construct and extend cell-type hierarchies", *NAR Genomics and Bioinformatics* (2023)

K. Biharie, **L. Michielsen**, M.J.T. Reinders, A. Mahfouz, "Cell type matching across species using protein embeddings and transfer learning", *Bioinformatics* (2023)

L. Michielsen, M.J.T. Reinders, A. Mahfouz, "Hierarchical progressive learning of cell identities in single-cell data", *Nature Communications* (2021)

T. Abdelaal, **L. Michielsen**, D. Cats, D. Hoogduin, H. Mei, M.J.T. Reinders, A. Mahfouz, "A comparison of automatic cell identification methods for single-cell RNA sequencing data", *Genome Biology* (2019)

S.M.H. Huisman, B. van Lew, A. Mahfouz, N. Pezzotti, T. Höllt, **L. Michielsen**, A. Vilanova, M.J.T. Reinders, B.P.F. Lelieveldt, "BrainScope: interactive visual exploration of the spatial and temporal human brain transcriptome", *Nucleic Acids Research* (2017)