Stories in Single Cell RNA sequencing

Thesis by

Eduardo da Veiga Beltrame

In Partial Fulfillment of the Requirements for the Degree of
Doctor of Philosophy

CALIFORNIA INSTITUTE OF TECHNOLOGY
Pasadena, California

2022
Defended 2021-10-29

munfred.com
ORCID 0000-0002-1529-9207

Dedicated to the Public Domain.

Acknowledgements

I owe many many thanks.

First of all, to Caltech and the bioengineering department, for taking a chance on me. And to Jané Kondev, my undergraduate advisor at Brandeis University, for telling me to apply to Caltech.

To the CMS happy hour, the most important and memorable of all Caltech events. To everyone that organizes and attends it. To Adam Wierman for signing off the tab and not minding half of the BBE department crashing it. To the Athenaeum staff for putting up with this rambunctious crowd until well past closing time, and to Adam Avery for making the best margaritas.

To the ISP staff, for putting out the best orientation, the source of many lasting friendships. To Daniel Yoder and Laura Kim, for putting up with my scrumpidulous emails and always responding in kind.

To the Caltech Library, for hosting the Techlab, and for all the other things the library does. In particular, to Kara Whatley, Tom Morrell, Stephen Davison, Donna Wrublewsk, Christine Ngo, and Paula Gaetos.

To the BBE staff for always being helpful and good sports no matter that. In particular, to Lauren Breeyear, Kenya Zeygler, Sue Zindle, Linda Scott, and Manny de La Torre.

To the Caltech Y, for being a hub for student life and thought beyond science. To Greg Fletcher for coordinating it all.

To Lior Pachter for being my first advisor and putting up with me as much as possible, and to everyone in the Pachter Lab. To Ali Sina Booeshaghi for a lot of work and discussions done together. To Jase Gehring for being a great mentor and teaching me almost everything that I learned about wetlab at Caltech. To Taleen Dilanyan for pushing forward our wetlab project and doing a much better job than me.

To Paul Sternberg for taking me in at the beginning of my third year, and giving me all the support and intellectual liberty one could ever ask for. To everyone at the Sternberg Lab and at WormBase. To Mark Zhang for turning an analysis done in a hunch into an ex-

perimental paper. To Valerio Arnaboldi for helping turn ideas and prototypes into real tools for people to use at WormBase.

To the Caltech Sovereignty Club and everyone that ever attended a meeting (Wednesdays 6 pm!) and especially to those unfazed who somehow kept coming back. I continue to learn a great deal about the world in our many discussions. To Michael Zhang for teaching me much of what I know about the Roman Empire and several other places, and having the most discerning and critical eye of current and historical events of anyone I know. To Austin Liu for telling me all the things about Singapore, blockchain and the Celestial Empire. To Rajashik Tarafder for educating me on India. To Josh Hejna for showing up and helping me move. To Kenny for being the best editor the Tech will ever know. To James Tyrwhitt-Drake for many years of discussion on science, art, and the point of everything. To Brener Martins for helping me explain the complexities of Brazil to a captive audience.

To Dylan Bannon for seriously putting up with my bad jokes and always being willing to embark on the latest crazy side project.

To David Brown and Tatyana Dobreva, for being the most formidable duo I have ever seen, always ready to bounce ideas. I am glad that we can continue to hang out doing crazy startups in Silicon Valley.

To Valentine Svensson for teaching me more than anyone else about statistics, single cell and cocktails, while somehow never tiring of answering my random questions.

To all my Caltech friends and people always willing to explain things, bounce ideas or work on projects: Aiden Aceves, Andrey Shur, Bruce Wittmann, Florian Schäfer, Guruprasad Raghavan, Jacob Barlow, Jeff Parker, Jeremy Berntstein, Michael Flynn and William Poole.

To my mom Angela da Veiga Beltrame for having always incentivized me to learn and pursue my interests.

To everyone at the Vilson Groh Institute in Florianópolis, for showing that the thorniest problems of Brazilian society have a clear solution, and for creating a path for so many more people to have access to the opportunities that I had. The effectiveness of your work motivates me to dream even higher in changing the world.

Contents

List of Figures

List of Tables

Abstract

This thesis describes the projects I have worked on since starting the Caltech bioengineering program in fall 2017. The general theme of my projects is that they are all about single cell RNA sequencing (scRNA-seq), spanning the experimental and computational realms.

Chapter 1 is an introduction explaining the essential concepts and is meant to be readable by a wide audience. For the other chapters, each one describes a separate project in a succinct manner, including links to the related preprint, published paper or code repositories at the start of each chapter.

Chapter 2 describes the scVI generative model for scRNA-seq data and the scvi-tools framework, which forms the basis of many of my computational projects.

Chapter 3 describes an open source 3D printable syringe pump system that was developed envisioning facilitating many kinds of experiments, in particular droplet based scRNA-seq.

Chapter 4 describes a new way of fabricating hydrogel beads with unique DNA barcodes that are used for scRNA-seq experiments.

Chapter 5 describes a database listing most published scRNA-seq studies that I helped create, and provides a useful overview of the state of the field.

Chapter 6 describes the kallisto bus workflow, which is used for pre-processing scRNA-seq data, going from FASTQ file to gene count matrix in a very efficient manner.

Chapter 7 describes a new way of using scVI to quantify the trade-off in the quality of scRNA-seq of a given dataset when surveying more cells or sequencing more reads per cell.

Chapter 8 describes tools developed for the WormBase users to leverage scRNA-seq data on *C. elegans*, and which can be deployed with any other scRNA-seq dataset.

Chapter 9 describes a remarkably successful offshoot of the development of these tools: a simple scVI based analysis and visualization strategy for finding candidate marker genes using *C. elegans* scRNA-seq data, which was experimentally validated by members of the Sternberg lab.

Published Content and Contributions

Principles of open source bioinstrumentation applied to the poseidon syringe pump system.

A. Sina Booeshaghi, Eduardo da Veiga Beltrame, Dylan Bannon, Jase Gehring and Lior Pachter.

Scientific Reports. 2019 August 27.

doi: 10.1038/s41598-019-48815-9

E.B. helped with the design the poseidon system and oversaw hardware printing and design. A.S.B. and E.B. tested the poseidon system. J.G., A.S.B. and E.V.B. formulated the design principles. A.S.B., E.B., J.G. and L.P. wrote the manuscript.

Quantifying the tradeoff between sequencing depth and cell number in single-cell RNA-seq.

Valentine Svensson, Eduardo da Veiga Beltrame, and Lior Pachter.

bioRxiv 2019

doi: 10.1101/762773

E.V.B. performed data processing and subsampling. V.S., E.B., and L.P. interpreted results and wrote the manuscript.

Patent US20200102556A1: *Efficient combinatorial bead barcoding.*

Eduardo da Veiga Beltrame, Jase Gehring, Akshay Tambe, Lior S. Pachter, and Taleen Dilanyan.

Available at: patents.google.com/patent/US20200102556A1/en

E.B. helped in initial development of the technology prior to filing of the provisional patent.

A curated database reveals trends in single-cell transcriptomics

Valentine Svensson, Eduardo da Veiga Beltrame, and Lior Pachter.

Database. 2020.

doi: 10.1093/database/baaa073

E.B. aided with gathering data, analysis, and writing the manuscript.

Modular, efficient and constant-memory single-cell RNA-seq preprocessing

Páll Melsted, A. Sina Booeshaghi, Lauren Liu , Fan Gao, Lambda Lu, Joseph Min, Eduardo da Veiga Beltrame , Kristján Eldjárn Hjörleifsson , Jase Gehring, and Lior Pachter.

Nature Biotechnology. 2021 April 1.

doi: 10.1038/s41587-021-00870-2

A.S.B. and E.B. designed and produced the comparisons between Cell Ranger and kallisto bustools. A.S.B., E.B., P.M. and L.P. planned the manuscript.

Single cell tools for WormBase

Eduardo da Veiga Beltrame, Valerio Arnaboldi, and Paul W. Sternberg.

bioRxiv 2021

doi: 10.1101/2021.07.04.451030

E.B. conceived and designed the tools, implemented scdefg and provided help with the development of wormcells-viz and wrote the manuscript.

scvi-tools: a library for deep probabilistic analysis of single-cell omics data.

Adam Gayoso, Romain Lopez, Galen Xing, Pierre Boyeau, Katherine Wu, Michael Jayasuriya, Edouard Melhman, Maxime Langevin, Yining Liu, Jules Samaran, Gabriel Misrachi, Achille Nazaret, Oscar Clivio, Chenling Xu, Tal Ashuach, Mohammad Lotfollahi, Valentine Svensson, Eduardo da Veiga Beltrame, Carlos Talavera-López, Lior Pachter, Fabian J. Theis, Aaron Streets, Michael I. Jordan, Jeffrey Regier, and Nir Yosef.

bioRxiv 2021.04.28.441833

doi: 10.1101/2021.04.28.441833

E.B created tutorials and documentation.

1 Introduction

Aside from viruses, all living things are made of cells, self replicating bags of molecules. There are many types of molecules in the cell, but here we will focus almost exclusively on only three important kinds: DNA, RNA, and protein. They are polymers, chains of a few kinds of molecules that serve as building blocks, called monomers. Different monomers, being different molecules, have different properties (size, charge, how flexible they are), and their sequence determines the properties of the polymer. This is why sequencing is such an important tool in molecular biology: it allows us to identify what molecules are present in a sample, and what their properties are.

In proteins the monomers are amino acids. There are 20 of them. All you need to remember is that amino acids can have very different properties and are really versatile, enabling proteins to do all kinds of things in the cell, such as chemical reactions. Proteins that perform chemical reactions are called enzymes.

In DNA (deoxyribonucleic acid) the monomers are nucleotides: adenine (A), thymine (T), guanosine (G) and cytosine (C). The backbone of DNA contains a sugar molecule called deoxyribose that has an oxygen atom making it very stable and rigid. This stability makes DNA an excellent medium to store the genetic information of the cell. When a DNA molecule is paired with another containing a complementary sequence, it forms the famous DNA double helix structure.

In a cell the DNA molecules with the instructions for everything the cell does are called the genome. In all multicellular organisms the cell genome is tightly tucked away inside the nucleus, a compartment from which molecules cannot easily get in or out of. Having a nucleus is the defining feature of eukaryotes. Many single celled organisms do not have a nucleus, they are prokaryotes and archaea, and their genome is floating all around the cell in as one or more more big pieces of DNA.

In RNA (ribonucleic acid) the monomers are adenine (A), uracil (U), cytosine (C), and guanine (G). The information encoded in the sequence of bases in a piece of DNA can be copied into an equivalent sequence in an RNA molecule. This process is called transcription, it is done by an enzyme called RNA polymerase. RNA molecules produced from a DNA template are called transcripts. A major function of RNA in the cell is to serve as a template for copying some information from a stretch of DNA and taking it to other places in the cell to make proteins. The kinds of RNA that are specifically being used to make protein are called messenger RNA (mRNA). Stretches of the genome that contain information that encodes transcripts are called genes.

The backbone of RNA is a ribose sugar. It similar to deoxyribose, which forms the backbone of DNA, but without an oxygen atom. That makes RNA floppier so it can fold in many kinds of different structures and perform other useful things in the cell. The best example of this is the ribosome, the molecular machinery that makes protein by reading the sequence of amino acids to add from a molecule of mRNA.

To summarize: The process of going from DNA to RNA is called transcription, done by RNA polymerase enzymes. These RNA molecules are transcripts. The control of this process by the cell is transcriptional regulation. The segments of DNA containing the information to make transcripts are called genes. Going from RNA to protein is called translation. Translation is done by ribosomes, large molecular machines made of RNA and protein. The process of producing the molecules that are used by the cell (which sometimes are the RNA molecules themselves, sometimes proteins) is called gene expression.

When RNA is first transcribed it is all located in the nucleus, and to perform most of its functions it must be exported out of the nucleus into the cytoplasm (the rest of the cell). However, RNA that will become proteins, called messenger RNA (mRNA) must first be processed before being exported. That is because not all of the content of a gene may be an exon, a region which encodes a part of a protein. So the pre-mRNA undergoes splicing, a process where introns, the parts that don't encode protein, are removed and the ends are joined together, leaving a mature mRNA made entirely of exons. Splicing is done by the spliceosome, a molecular machine located in the nucleus that, like the ribosome, is made of both protein and of RNA. This is summarized in Figure 1.1.

This flow of information also reflects how hard it is to study RNA, DNA and protein in cells. Between 2007 and 2010 with the introduction of next-generation sequencing technologies (NGS) the costs for DNA sequencing plummeted by about ten thousand times to about ten cents per million base-pairs sequenced[1]. Now that cost is at about one cent per million base-pairs. Decreasing costs made our ability to read DNA a commodity tool, and makes it very convenient to have

[1] The NIH tracks the cost of DNA sequencing and briefly discusses it's evolution here: `https://www.genome.gov/about-genomics/fact-sheets/DNA-Sequencing-Costs-Data`

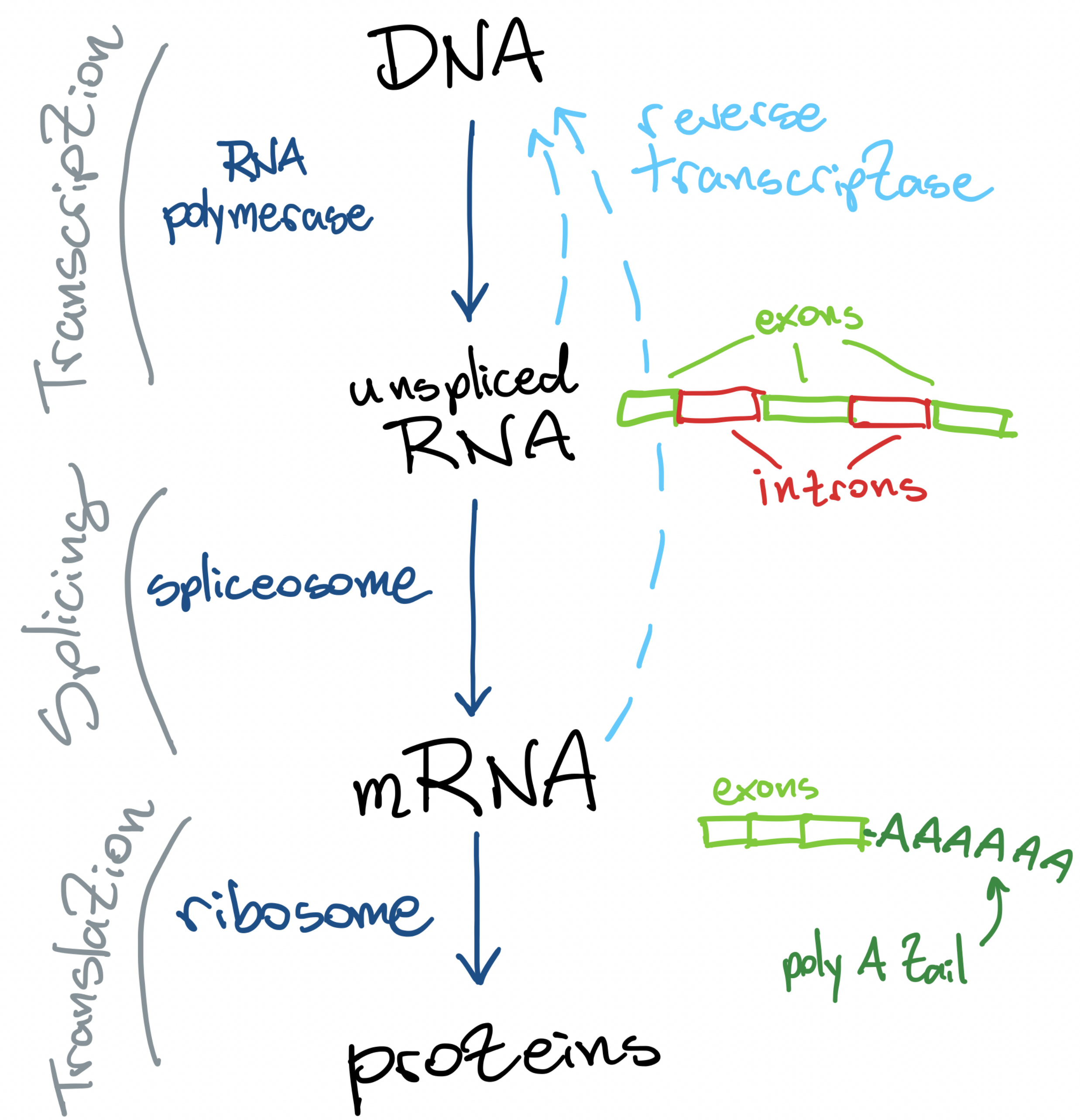


Figure 1.1: An overview of the processes that happen inside the cell when going from DNA to protein.

sequences be the information output of many biological experiments. If the experiment to answer a biological question can be turned into a sequencing problem, it can be done cheaply, and it can be done at scale.

Because we can convert RNA to DNA using reverse transcriptase enzymes, it is possible to use DNA sequencing to study RNA molecules. We can't do the same thing with proteins, and so it is harder to study protein molecules than it is to study DNA or RNA.

The DNA of an individual is essentially the same across all cells. DNA is a very stable molecule, and the genome has to be very stable so that it can be reliably copied and passed on to the next generation of cells. With RNA it is a different story. Every cell at each moment will have a different composition of RNA molecules, which reflects what the cell is doing at that point in time. Transcriptional regulation is how the cell controls how much and what kinds of RNA molecules it makes.

Understanding transcriptional regulation is key for understanding, controlling, modifying and engineering biological systems. To study transcription it is necessary to measure the RNA in the cell. Experiments that use sequencing to look at the composition of RNA molecules in a sample are commonly referred to as RNA-seq experiments[2].

Because the RNA in a cell changes all the time, there is an endless amount of RNA-seq experiments that could be performed in a single species - even in a single individual. One of the most informative things to look at are the differences between individual cells and populations of cells, or tissues. A tissue sample will contain multiple cell types, and each cell might be in a different stage doing different things. The average of the RNA contents of all the cells in a tissue is usually going to be very different from the contents of each cell, because usually cells in a tissue will have very different content. This heterogeneity of a population of molecules in cells is what makes it important to measure their RNA content many times and under multiple conditions.

Not all types of RNA are heterogeneous. For many types of RNA the composition across cells is largely the same, and it doesn't change much over time - these kinds of RNA are very homogeneous across cells. Ribosomes, which are made of a few several dozen molecules of RNA and protein are the prime example of RNA homogeneity in cells. Cells need to make a lot of protein very quickly, so they have a lot of ribosomes, and sometimes up to 80% of the RNA in a cell is ribosomal RNA, which will always have the same composition. Thus it is not interesting to measure the ribosomal RNA of cells multiple times under multiple conditions, because it doesn't change much

[2] If interested, the best resource I know of to learn all the important aspects of RNAseq experiments (most of which also apply to scRNA-seq) is the RNA-seqlopedia, written by the Cresko Lab of the University of Oregon: `https://rnaseq.uoregon.edu/`

Fortunately eukaryote organisms evolved a mechanism to add a special tag called polyadenylation tail to mRNA that allows for capturing and sequencing only mRNA molecules, which change all the time and are very interesting to measure. As part of pre-mRNA processing, dozens to hundreds of A letters (adenine molecules) are added to the end of every mRNA molecule. This long AAAAAAA... sequence makes it straightforward to capture, amplify and sequence the mRNA of a sample by using a single probe that is complementary to the poly A tail - a poly T probe: TTTTTT. . .

Because of this quirk of biology, it is possible to look at all the mRNA in cells without the need to capture everything else, which would include all of the homogeneous ribosomal RNA, and which would significantly increase sequencing costs and make many experiments impractical. There are other heterogeneous RNA molecules, such as long non coding RNAs (lncRNA) that do not have poly A tails. As a casualty of convenience, these other kinds of RNA that do not have a poly A tail do not get studied nearly as much as mRNA. It is possible that we are missing fundamental parts of the puzzle by focusing too much on only mRNA, but only time will tell.

An alternative strategy to capturing everything is to use specific probes to capture only things you already were interested in ahead of time. This latter approach is what was used in microarrays[3], which were invented in the 1980s and widely used into the 2010s, when RNA-seq became a very popular technique due to lower requencing costs and the ability to capture all mRNA without deciding what to look for ahead of time.

There are multiple variations of experimental procedures for performing RNA-seq, but it typically involves the following main steps.

1. A sample containing cells is homogenized, meaning the cells are broken using reagents such as detergents, and the RNA is released in solution. Typically the sample will be enriched for mRNA and then purified.

2. Using the reverse transcriptase enzyme, a strand of DNA that is complementary to each RNA piece is created. This DNA is referred to as cDNA (complementary DNA).

3. The long cDNA pieces, that can be several thousand of base pairs long (many transcripts are very long!) are broken into smaller pieces of no more than a few hundred base pairs each, as the sequencing platform usually requires short pieces.

4. Extra sequences are added to the ends of each cDNA molecule so that they can be sequenced and the cDNA is amplified (meaning many copies of each cDNA molecule are produced). The exact

[3] For the interested reader, here is a nice historical review of the invention of microarrays:

Michael C. Pirrung and Edwin M. Southern. The genesis of microarrays. *Biochemistry and Molecular Biology Education*, 42(2):106–113, 2014. doi:10.1002/bmb.20756 steps vary depending on protocol. The process of preparing a collection of cDNA molecules for sequencing is typically referred to as library preparation.

5. After library preparation the sample is sequenced, and a list of the sequences in each cDNA fragment is created. Processing and analyzing this list of strings containing the four letters ATCG comprises most of the discipline of bioinformatics.

Single cell RNA sequencing (scRNA-seq)

As sequencing costs decreased it became possible to process more RNA-seq samples in a single experiment. But the samples for "bulk" RNA-seq samples are essentially a smoothie of a piece of tissue, and this makes it hard to look at the cellular heterogeneity. For example, if a tissue consists of many different cell types, like the brain, then it is hard to obtain a pure sample that has only one cell type in it, such as neurons or glial cells, because different cell types are physically intermingled and it is hard to separate them.

Additionally, because cells of the same type can be in different states doing different things, looking at heterogeneity at the individual cell level requires a way of separating and sequencing the RNA from individual cells. For example, a cell undergoing division (in the mitotic phase, when cell growth stops) will be expressing different genes than a cell that is quiescent or growing (in the interphase).

This is the fundamental motivation behind single cell RNA sequencing (scRNA-seq) and all other single cell techniques: to be able to look at heterogeneity and understand how the cells in a sample are different from each other it is necessary to measure each cell individually. If a tissue does not have a lot of cellular heterogeneity, then there is not much more to be learned from looking at individual cells than from an average aggregate. The difference between single cell RNA sequencing and bulk RNA sequencing is that between drinking a smoothie and tasting individual berries. While on average the smoothie and berries will look the same, the smoothie taste masks the heterogeneity of individual members of the berry population.

The output data of a scRNA-seq experiment is a gene count matrix, containing one row for each cell, and one column for each gene from which an mRNA came from. Across all cells usually between 10.000 to 20.000 genes are seen, and a typical experiment will survey a few thousand cells, although it is currently possible to survey tens or even hundreds of thousands of cells in larger experiments.

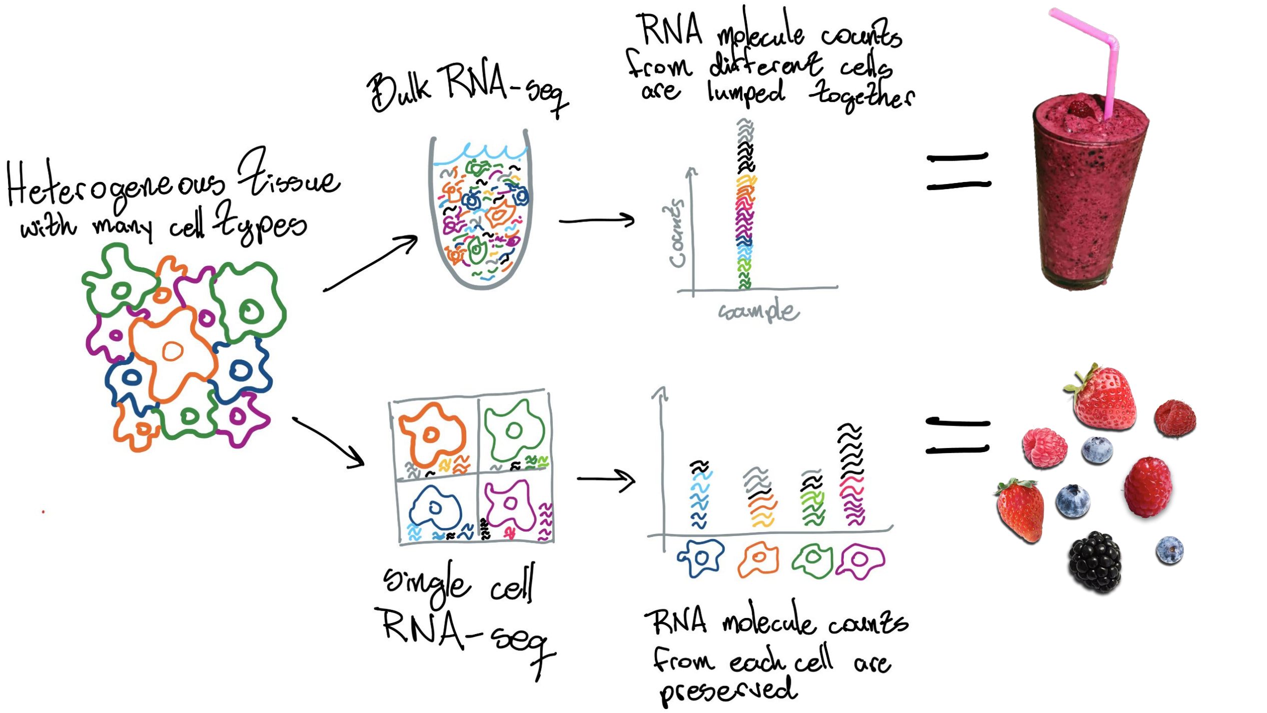

Heterogeneous tissue with many cell types
Bulk RNA-seq
RNA molecule counts from different cells are lumped together
counts
sample
=
single cell RNA-seq
RNA molecule counts from each cell are preserved
=

Figure 1.2: The difference between single cell RNA sequencing and bulk RNA sequencing is that between drinking a smoothie and tasting individual berries. While on average the smoothie and berries will taste the same, the smoothie taste masks the heterogeneity of individual members of the berry population.

Steps in scRNA-seq experiments

A Typical scRNA-seq has three main steps: cell isolation, barcoding, and library preparation. Chemically speaking, scRNA-seq is very similar to (bulk) RNA-seq with very little starting material - a typical mammalian cell has on the order of a hundred thousand mRNA molecules[4]. Nevertheless, a now typical scRNA-seq experiment sequences between about 500 to 10.000 original mRNA molecules per cell, which suggests we are surveying 1-10% of the mRNA molecules in each cell.

Cell Isolation: Cells are physically dissociated and usually isolated physically, being separated into different containers, for example, in the wells of a -well or 384-well plate. Three kinds of approaches are used for the physical separation of cells: plate based, droplet based, and split-pool based methods. Methods are frequently grouped according to the cell isolation method, which we discuss more below.

Barcoding: Upon reverse transcription, two short sequences (10-20bp) are added in addition to the mRNA cDNA sequence: a unique sequence corresponding to the original cell, plus a random sequence corresponding to the original molecule, called a unique molecular identifier (UMI). Most scRNA-seq methods have UMIs, as their absence makes accurate quantification of original molecules signifi-

[4] A mammalian cell typically has 10-20pg of RNA and the average mRNA molecule has 2200 bases, corresponding to about 500 daltons, so 10pg of RNA corresponds to about one million RNA molecules. It is estimated that 2-5% of the RNA in a human cell is mRNA, thus 20,000-50,000 mRNA molecules is a reasonable estimate for the number of mRNAs to expect in a typical cell. This estimate shouldn't be taken blindly, as different cell types can vary wildly on their sizes, and mammalian cells tend to be larger compared to other species. But the take home message is that scRNA-seq seems to be capturing a large fraction of the mRNA in cells.

See the three relevant BioNumbers entries for this estimate at:
`https://bionumbers.hms.harvard.edu/bionumber.aspx?id=111204`,
`https://bionumbers.hms.harvard.edu/bionumber.aspx?id=101469&ver=1`,
`https://bionumbers.hms.harvard.edu/bionumber.aspx?id=111540`

cantly harder[5].

Library preparation: Cells then undergo the same procedure performed for bulk RNA-seq samples, but on a much smaller scale of individual reactions.

Main kinds of scRNA-seq cell isolation methods

There are now hundreds of studies describing different methods and protocols for performing scRNA-seq. Often these methods are tweaks and improvements on existing methods. Broadly speaking, there are three main ways in which cells may be isolated and barcoded: in physical containers (plate methods), in microfluidics emulsions (droplet methods) and via sequential split-pool barcoding.

Plate based methods: Cells are manually or robotically isolated in physical compartments such as 96 or 384 microwell plates, with a typical throughput of hundreds or thousands of cells. Barcoding happens by adding a distinct DNA barcode to each well.

Droplet based methods: Cells are encapsulated in a droplet using a microfluidic device. In addition to a cell, each droplet also encapsulates a DNA coated bead, and this DNA has a unique sequence for each bead. Upon lysis, the cell releases it's mRNA which is captured by the bead DNA and reverse transcribed so that the barcode is added to the cDNA pieces.

Split-pool methods: Cells are not all physically isolated at once. Instead, the sequence and the transcript sequence are now on the same piece of DNA. Cells are manually or robotically split into a few dozen or few hundred separate compartments. Within each compartment a partial barcoding happens, adding a common barcode to all cells in it. Cells are then mixed back together, and then split again, repeating this procedure. The number of potential barcodes that can be created is given by the number of compartments to the power of the number of rounds. By making the number of potential barcodes much greater than the number of cells (e.g. a million barcodes with ten thousand cells) the number of cells with the same barcode can be made very small, and thus each cell can be considered to receive a unique barcode.

[5] That's because when individual molecules are copied via PCR (polymerase chain reaction) different molecules have different numbers of copies made. If there are no UMIs, it is impossible to tell how many mRNA molecules (transcripts) of each gene there were originally, because two transcripts of the same gene often have the exact same sequence. In order to quantify original abundances without UMIs it is necessary to make estimates of how much each transcript gets amplified based on their sequence (which can cause amplification biases), and this is a hard challenge.

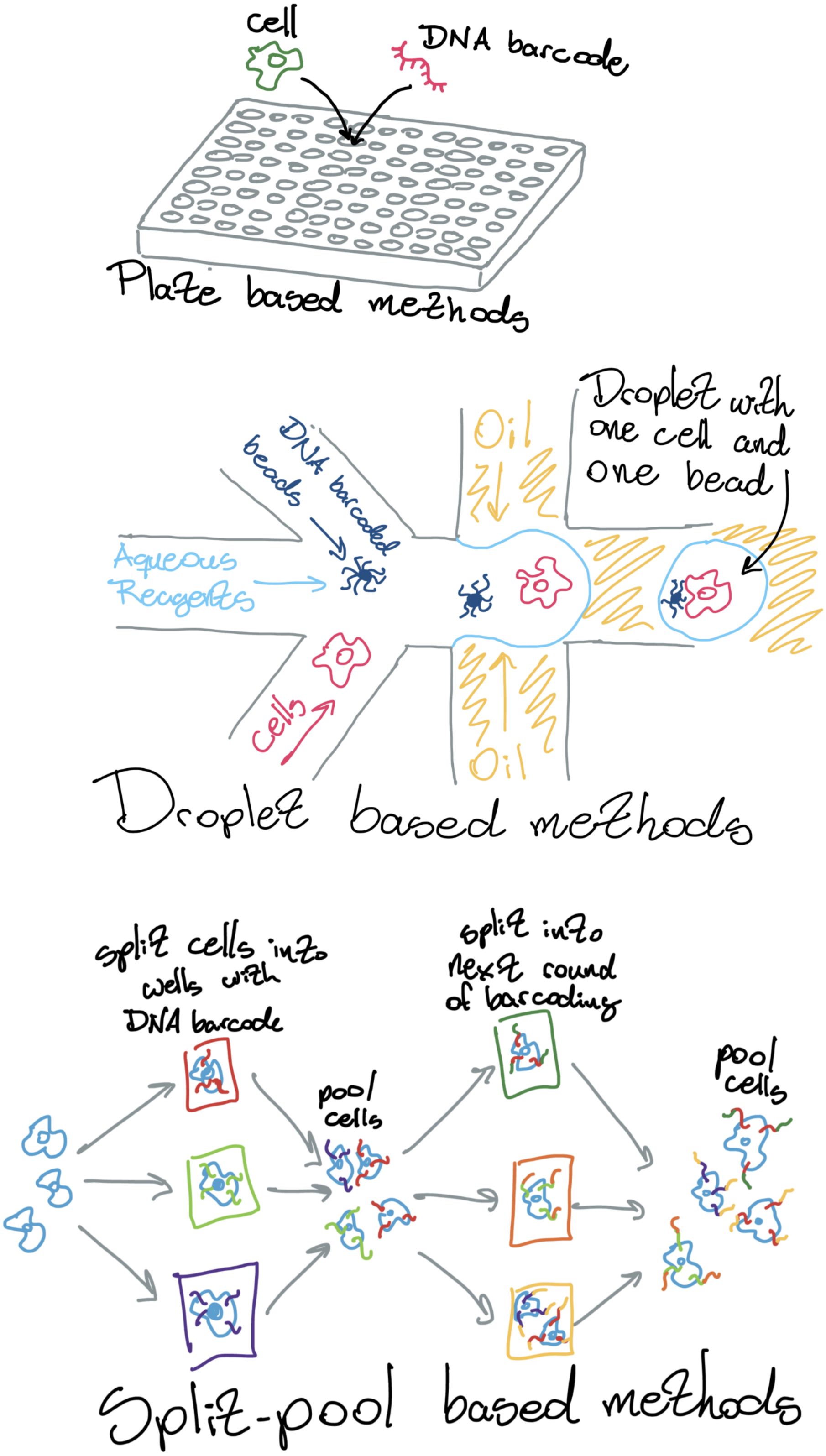


Figure 1.3: Conceptual illustration of the three barcoding strategies: plate based, droplet based, and split-pool based methods.

Annotating scRNA-seq data

The gene count matrix produced by a scRNA-seq experiment contains the information of how many mRNA molecules from each gene were seen in a cell. Identifying what types of cells exist in a sample based on the transcripts seen is a task commonly referred to as annotation.

Annotation is fundamental, because stratification of data in a sample is the whole point of scRNA-seq: if you don't annotate you can't stratify groups to compare, for example neurons vs glial cells. Without identifying the subpopulations present in a sample, scRNAseq yields information equivalent to bulk RNA-seq, where the transcripts of all cells are averages.

Annotation is perhaps the biggest challenge when datasets are the first of their kind: when doing an experiment on a new species, a new tissue, or a new technique for the first time. In such cases direct comparison with existing datasets may not be possible. These novel datasets typically need to undergo exhaustive manual annotation, and the annotated dataset forms a stepping stone for subsequent studies in related systems.

When a biological system has been extensively studied using other methods, it is often possible to annotate cell types by carefully reviewing the literature for marker genes, which are genes that are only expressed in one or a few cell types. With a list of marker genes for each cell type of interest in hand it is possible to annotate them in a sample.

Some systems such as blood have only a few cell different cell types, all already well studied and with distinct marker genes. This enables annotation of cell identity with confidence. However, when there are dozens or hundreds of potential cell types, and when not all of them have well established markers (or when the known markers cannot delineate different states), assigning cell identity is a more subjective task.

In these cases the common approach is to try to cluster (group) the cells based on some measure of similarity, compare each cluster with the remaining cells, and then based on the genes that are different between the two groups (differentially expressed genes), attempt to identify a subset of them matching the profile of a known tissue.

This process has many caveats. There are many different workflows that could be used to define clusters. The degree to which clusters should be broken down or combined is also subjective. Ideally each cluster should correspond to a single cell type, but how to know when that is the case? Even when cell types are well defined, if there are many of them, multiple rounds of clustering, inspection, and sub-clustering may be needed.

One strategy to address some of this is to use a hierarchical taxonomy instead of a flat one. In addition to reflecting the fact that certain cell types can have subtypes and sub-subtypes, a taxonomy also naturally reflects the uncertainty in our classification. For example there might be clear neuronal markers that distinguish them from glia and other brain cells, but some neuron types might be better characterized than others, meaning that as we go down in the taxonomy the uncertainty increases. It is fair to say that annotation of cell types in a new biological sample is the most challenging and time consuming step of scRNA-seq analysis.

scRNA-seq and machine learning

Over the past few years, the amount of data being generated with scRNA-seq techniques has scaled exponentially, with the number of cells surveyed in a single study going from dozens to millions[6]. Each year hundreds of studies are published, often containing dozens of experiments and tens of thousands or hundreds of thousands of cells each. This is because for a few thousand dollars, it is now possible to profile tens of thousands of single cells in a standard experiment[7].

An interesting feature of scRNA-seq data is that it is very standardized: a big sparse matrix of cells by gene counts, and at the moment most of it is produced with one technology commercialized by one company, 10x Genomics (the same way that almost all sequencing is done with technology commercialized by Illumina).

Because the data generated by a scRNA-seq experiment captures data from all mRNA in the cell, data generated to answer one particular biological question may lend itself to answering other questions that look at different aspects of biological variation. For example, in one of the projects described here we were able to use published scRNA-seq data on *C. elegans* to identify new neuronal marker genes.

As we develop better techniques to integrate, annotate and compare cells, old data gains new value. It may be reanalyzed together with new data, and the number of questions and experiments one could conceive from the large datasets being created is much greater than any individual lab could pursue, and the great boon of sharing data is enabling reanalysis and asking new questions.

This stands in contrast to the way most experimental science is conducted: with experiments carefully designed to gather the right data to answer a specific question. As the amount of public data continues to grow exponentially, we will have to learn to come back to old data armed with new questions, principled methods, and a lot of scruples so as not to end up fixating on artifacts. Perhaps not fixating

[6] Valentine Svensson, Roser Vento-Tormo, and Sarah A. Teichmann. Exponential scaling of single-cell RNA-seq in the past decade. *Nature Protocols*, 13(4):599–604, April 2018. doi:10.1038/nprot.2017.149

[7] A standard scRNA-seq experiment from 10x Genomics (`https://10xgenomics.com`) will cost $1000-2000 for 10-20k cells plus another $1000-2000 for sequencing.

on artifacts will be easier to do with old data, since with new data there is typically a much greater motivation to find "something".

It is interesting to note that this is not a new phenomena in the biological sciences - very much the same kind of thinking was spurred by microarrays, and long after writing the preceding paragraph, I ran into the following passage in this 2006 article on microarrays by Jörg D. Hoheisel [8]. Nothing new under the sun, as they say:

Microarray technology has initiated an experimental approach that is based on unbiased sample screening and accumulation of data, preceding the formulation of hypotheses. To an extent, it has placed data production before intellectual concepts, although of course further and more detailed studies are required to confirm and refine the hypotheses that result from such studies. In this respect, biology is becoming more similar to physics. Although the value of this approach in biology is still a subject of debate, physics has clearly demonstrated its power. However, even those who are used to microarray technologies sometimes still need to dissociate themselves more fully from a hypothesis-driven view, as it is not data production but data interpretation that is still often biased by pre-existing ideas.

At the same time that this tidal wave of biological data started, driven primarily by ever decreasing sequencing costs, another revolution started unfolding in statistics and computer science, driven primarily by ever decreasing computing costs. In the past decade the broad field of machine learning saw very rapid development as ever larger neural networks were successfully applied in all kinds of ways to all kinds of data, such as image classification, speech recognition, and control systems.

The current wave of excitement got started in 2012 with the success of AlexNet, a neural network for image classification that performed significantly better than everything else at the time[9]. This caused an ongoing flood of attention, investment, and research developing all kinds of extremely clever algorithms for dealing with data. Often these ideas are tested in toy and benchmarking scenarios (such as standard sets of images or texts used in benchmarking), because that's where a lot of curated data is readily available.

But as time goes on and the dust settles, people start applying the most promising ideas and algorithms to new domains, and seeing what works well and what doesn't. This is where machine learning really impacts science, when expert domain knowledge is coupled with judicious application of suitable algorithms for the system at hand.

Single cell RNA sequencing in particular is at a really interesting point[10], because the data is all in a standard format (a matrix), there is a lot of it publicly available, and there is a lot of interest and potential for doing something useful with this data. At the same time,

[8] Jörg D. Hoheisel. Microarray technology: beyond transcript profiling and genotype analysis. *Nature Reviews Genetics*, 7(3):200–210, March 2006. doi:10.1038/nrg1809

[9] The story of how AlexNet kicked off the current excitement wave is chronicled in this 2018 Quartz article: `https://web.archive.org/web/20210301025354/https://qz.com/1307091/the-inside-story-of-how-ai-got-good-enough-to-dominate-silicon-valley/`

[10] The other really interesting high-throughput standardized kind of data being created in biology is imaging data, for which translation of machine learning methods should be even more straightforward, since many of them are already developed for images.

there aren't yet enough people thinking about it from the machine learning side. It has only been 9 years since AlexNet, and industry has so far soaked up most of the machine learning trained researchers, and I think the landscape will change dramatically in the coming years as machine learning researchers turn their attention to other kinds of standardized data.

This bonanza of computational methods is a blessing (and a temporary headache) for biologists drowning in data. While we now have efficient methods for dealing with high throughput scRNA-seq data, there are *many* of methods[11]. These tools range from unpublished scripts, preprints, to entire frameworks maintained by several people, and it is not at all clear which one is the "best". Rapid development also means that careful benchmarking and comparison ends up on the back burner. It will likely take a few more years for the dust to settle.

For example, in the beginning of scRNA-seq, many methods for doing differential expression and visualization were directly taken from bulk RNA seq. Although the experimental methods are very similar, the data from single cell and bulk RNA seq are different, and linear algebra techniques should not be judiciously applied[12].

Even though they are commonly applied, techniques like normalization and log transformation are prone to introducing artifacts and false variability on scRNA-seq data. Given a matrix of gene counts X, normalization is the practice of taking the counts X_{ig} of cell i and gene g and dividing them by a cell scaling factor S_i to obtain a new normalized value X_{ig}/S_i . Log transformation usually means adding a *pseudocount* c, (where usually $c = 1$) to the original count value and taking the log of that, to obtain a new value $log(X_{ig} + c)$, but could also mean performing that operation with the normalized values; $log(X_{ig}/S_i + c)$. For a discussion on systematic errors caused by normalization and log transformation, see Lun 2018[13]. For another discussion on these issues and problems that might arise due to normalization and log transformation, as well as a proposed alternative to standard PCA that does not rely on the Gaussian assumption see Townes 2019[14].

On the next chapter we will talk about one way to deal with these challenges: using bayesian generative models, particularly in the context of the scvi-tools framework (scvi-tools.org), which offers an extensible collection of generative models tailored for scRNA-seq data.

[11] `https://scrna-tools.org` currently counts 1059 tools developed for dealing with scRNA-seq data.

Luke Zappia, Belinda Phipson, and Alicia Oshlack. Exploring the single-cell RNA-seq analysis landscape with the scRNA-tools database. *PLOS Computational Biology*, 14(6):e1006245, June 2018. doi:10.1371/journal.pcbi.1006245

[12] For example, the commonly used principal component analysis method models data as coming from a continuous multivariate distribution, and optimizes a gaussian likelihood. The fact that normal distributions are not appropriate for dealing with low count values (where discreteness becomes apparent) and that poisson or negative binomial distributions should be used was already discussed by Anders and Huber in 2010 in the paper where they introduced the popular DESeq package for performing differential expression on bulk RNAseq data.

Simon Anders and Wolfgang Huber. Differential expression analysis for sequence count data. *Nature Precedings*, March 2010. doi:10.1038/npre.2010.4282.1

[13] Aaron Lun. Overcoming systematic errors caused by log-transformation of normalized single-cell RNA sequencing data. page 404962, August 2018. doi:110.1101/404962

[14] F. William Townes, Stephanie C. Hicks, Martin J. Aryee, and Rafael A. Irizarry. Feature selection and dimension reduction for single-cell RNA-Seq based on a multinomial model. *Genome Biology*, 20(1):295, December 2019. doi:10.1186/s13059-019-1861-6

Challenges in academic bioinformatics software development

Many common development issues with bioinformatics tools are a direct consequence of the common incentives in academia, where the publication prestige incentivizes people to claim to have done something novel often leads people to try to reimplement the wheel instead of adding onto existing tools, so that they can be better framed as something new and made from scratch. In a "hot" and rapidly field like scRNA-seq this causes the following issues which are frequently witnessed in academic software development:

Software is often developed by a single person. This increases the odds of bugs and problems with the implementation, because no one else reviews the code. Peer reviewers will not typically review someone's code (it is a lot of work) and just because code is open source and available it does not mean other people will check it. In practice having multiple people working on a codebase is the surest way to decrease the occurrence of errors, while at the same time making the code more readable for others.

The codebase is frequently not developed further after publication. Often this is because the academic incentives diminish after having a publication accepted, or because the person who wrote the code graduates and leaves the lab.

Workflows and methods are developed as ad hoc tools. Frequently new workflows are not thought through and many steps incorporate arbitrary (unjustified) choices. Doing ad-hoc things is an integral part of science and experimentation, but once something works, or seems to work, people tend to forget it was done ad-hoc. So people will extrapolate the context in which it is supposed to work, while disregarding the need for benchmarking or validation because everyone else is doing it. This is akin to developing a "superstition" or a "myth", and happens in science occasionally.

Software is hard to discover, deploy, and compare. There are so many tools, often annoying to install and run, that nobody can feasibly do an exhaustive search. People just use what their lab friends are using.

Software is rarely benchmarked outside original study. Benchmarking is a lot of work, and there is no incentive for further validating the software after publication. Additionally, most benchmarks are often only done in humans and mice because those are the most popular organisms, but people tend to assume that a method will work on all other scRNA-seq data. For example, one aspect of mammals is that they have large cells, while many other organisms such as *C. elegans* have small cells.

Some suggestions

I highlight these problems here because they are really pervasive - anyone working with scRNA-seq bioinformatics will have encountered them - and they merit more attention and discussion. I don't have a simple solution to prescribe, but I do have a few things I think people should have in mind when developing bioinformatics software.

Have independent code reviews. If developing code that will be used by other people, or that will be part of a publication (and thus potentially used by others), always ask a friend to do a code review. The ideal scenario would be to have your friend developing the software with you, so that if something is broken or hard to understand, it will be brought to your attention.

Be scrupulous about your own work while it is still being developed. We tend to want to do everything as fast as possible, but good work takes time. In reality it is always a compromise, but I think that starting from a mindset that good work takes time is helpful.

Consider how your software is going to be maintained after publication. If the software being provided is just meant to reproduce the publication this is less of an issue, but if you claim that other people should be able to use your software, then it is imperative that at least one person be responsible for maintaining the codebase after publication, and it is really important to discuss this with other authors.

Consider extending existing software rather than developing from scratch. Carefully assess what the landscape is, and whether there is already a codebase or framework to which your software workflow could be added. It is not always possible to do this, since frequently there will not be a good match, but if it is possible it's a win-win.

- More people will be looking at your code.

- It will make your code better, since there will likely be certain guidelines on how the contributed code should be structured.

- It will make your software easier to find and use, since there are other people already installing and using the framework.

- It may reduce the burden of maintaining the code, and will help keeping it up to date with other software dependencies make it easy to install, since the other developers will likely already be focusing and longer term support of the codebase.

Overview of next chapters

This chapter provided an introduction explaining the essential concepts of scRNA-seq. For the other chapters, each one describes a separate project in a succinct manner, including links to the related preprint, published paper, or code repositories at the start of each chapter.

Chapter 2 describes the scVI generative model for scRNA-seq data and the scvi-tools framework, which forms the basis of many of my computational projects.

Chapter 3 describes an open source 3D printable syringe pump system that was developed envisioning facilitating many kinds of experiments, in particular droplet based scRNA-seq.

Chapter 4 describes a new way of fabricating hydrogel beads with unique DNA barcodes that are used for scRNA-seq experiments.

Chapter 5 describes a database listing most published scRNA-seq studies that I helped create, and provides a useful overview of the state of the field.

Chapter 6 describes the kallisto bus workflow, which is used for pre-processing scRNA-seq data, going from FASTQ file to gene count matrix in a very efficient manner.

Chapter 7 describes a new way of using scVI to quantify the trade-off in the quality of scRNA-seq of a given dataset when surveying more cells or sequencing more reads per cell.

Chapter 8 describes tools developed for the WormBase users to leverage scRNA-seq data on *C. elegans*, and which can be deployed with any other scRNA-seq dataset.

Chapter 9 describes a remarkably successful offshoot of the development of these tools: a simple scVI based analysis and visualization strategy for finding candidate marker genes using *C. elegans* scRNA-seq data, which was experimentally validated by members of the Sternberg lab.

2 *Single cell variational inference*

This chapter is related to the published study:

scvi-tools: a library for deep probabilistic analysis of single-cell omics data

Adam Gayoso, Romain Lopez, Galen Xing, Pierre Boyeau, Katherine Wu, Michael Jayasuriya, Edouard Melhman, Maxime Langevin, Yining Liu, Jules Samaran, Gabriel Misrachi, Achille Nazaret, Oscar Clivio, Chenling Xu, Tal Ashuach, Mohammad Lotfollahi, Valentine Svensson, Eduardo da Veiga Beltrame, Carlos Talavera-López, Lior Pachter, Fabian J. Theis, Aaron Streets, Michael I. Jordan, Jeffrey Regier, and Nir Yosef.

bioRxiv 2021.04.28.441833

doi: 10.1101/2021.04.28.441833

Author contributions: *A.G., R.L, and G.X. contributed equally. A.G. designed the scvi-tools application programming interface with input from G.X. and R.L. G.X. and A.G. lead development of scvi-tools with input from R.L. G.X. reimplemented scVI, totalVI, AutoZI, and scANVI with input from A.G. R.L. implemented Stereoscope with input from A.G. Data analysis in this manuscript was led by A.G., R.L., and G.X with input from N.Y. A.G, R.L, P.B, E.M, M.L, Y.L, J.S, G.M, A.N, O.C. worked on the initial version of the codebase (scvi package), with input from M.I.J, J.R and N.Y. R.L, E.M and C.X contributed the scANVI model, with input from J.R and N.Y. A.G implemented totalVI with input from A.S and N.Y. T.A. implemented peakVI with input from A.G. A.G implemented scArches with input from M.L, F.T and N.Y. V.S. made several contributions to the codebase, including the LDVAE model. P.B. contributed the differential expression programming interface. E.B and C.T.L provided tutorials on differential expression and deconvolution of spatial transcriptomics, with input from L.P. K.W implemented CellAssign in the codebase with input from A.G.. M.J. made general code contributions and helped maintain scvi-tools. N.Y. supervised all research. A.G, R.L, G.X, J.R and N.Y wrote the manuscript.*

Bayesian generative models

At the same time that the current flood of scRNA-seq data started being generated a few years ago, new machine learning methods were being created that could be successfully applied to it. There is a broad class of machine learning models, called generative models, where the modelling emphasis is put on being able to generate data that looks like the input data utilizing the learned parameters. Arguably every time someone calls something a "model" it should capture something about the underlying dynamics, but generative models make this more explicit. Most generative models rely on Bayes rule for making inferences about the input data, so that they can learn probability distributions that reflect what the data looks like. The really cool thing about generative models is that because you can just simulate new data, it becomes trivial to just compute any statistic of interest in the simulated data.

The gist of Bayesian statistics is Bayes rule, which says that the posterior - the probability of a model given the data - should be proportional to likelihood - the probability of the data given the model, how likely we are to observe a set of data - times the prior - how likely the model is in the first place, which is something we arbitrarily come up with. We can write this as:

$$P(model|data) \propto P(data|model) \cdot P(model)$$

It is necessary to work with normalized quantities so that they behave as probability distributions, so the right side is divided by the marginal likelihood, $P(data)$.

Using x for our *data* (also called evidence) and z for *model* (typically called an unobserved, or latent variable) we can describe the posterior distribution $p(z|x)$ as an integral to be solved:

$$p(z|x) = \int_z \frac{p(x|z)p(z)}{p(x)} dz$$

This integral is very often mathematically intractable for all but the simplest cases. A lot of the recent rise in popularity of Bayesian methods is due to better and easier ways to calculate the posterior, and this is often what many recent advances in machine learning conceptually boils down to. That was the innovation introduced by variational autoencoders, a way of numerically calculating the posterior by instead optimizing a mathematical lower boundary of $p(z|x)$

Variational Autoencoders (VAEs)

In practice, once a bayesian model is specified, the computational challenge is how to calculate the posterior. A lot of recent advances in machine learning and statistics often boil down to being better ways to calculate the posterior. One such method is that of variational autoencoders, (VAEs), introduced in 2014 by Kingma and Welling[1].

VAEs quickly became a popular approach to modelling complicated distributions[2]. In a VAE an encoder neural network is trained to learn a nonlinear function $q_\phi(z|x)$ that maps an input point into low dimensional latent representation z. This encoding process is constrained by having a generative model that can approximate the true posterior $p(x|z)$ with an approximate one with learned parameters θ, denoted by $p_\theta(x|z)$, and can take the latent parameters to generate data that looks like the input data. The mathematical insight that makes VAEs useful is that they do not attempt to optimize (learn) the posterior distribution directly, and instead use optimize a function that is a lower bound for the posterior, called evidence lower bound (ELBO).

Soon VAEs were applied to single cell data, where the gene count matrix is a sparse matrix of integer numbers representing counts sampled in a process akin to drawing from an urn of balls (modelled by a multinomial distribution). Modelling the data like this provides a way to go from the discrete gene count matrix to a vector space $\mathbb{R}^n$ of n latent parameters.

One of the first descriptions of using VAEs for scRNA-seq data was scVI, first described in 2017[3], and other implementations such as scVAE[4] soon followed. Implementations of traditional autoencoders for scRNA-seq, such as DCA[5] were also described around the same time.

Because the latent space is a vector space, linear algebra techniques can be readily applied. Another important feature of VAEs is that they allow for modeling the specific data generating process, and in the case of scRNA-seq this enables taking into account relevant features such as cell size, sequencing depth, and dropouts. Because the generative model is devised for the physical process being modelled, it may be extended to model other phenomena that may be considered relevant, such as background RNA, or other experimental modalities, such as counting the number of proteins in each cell together with RNA.

The low dimensional latent representation of the data is directly amenable to classical and newer clustering and visualization techniques developed in machine learning. And having a generative process allows us to perform bayesian inference to learn about any

[1] Diederik P. Kingma and Max Welling. Auto-Encoding Variational Bayes. May 2014. arXiv: 1312.6114

[2] VAEs which uses a method called variational inference to calculate the posterior for machine learning models that have an architecture similar to that of autoencoder models, hence the name. Note that despite similarities, VAEs are different in their underpinning math than autoencoders. which have been around since the 80s. For more details on the math, see:

Carl Doersch. Tutorial on Variational Autoencoders. January 2021. arXiv: 1606.05908

[3] Romain Lopez, Jeffrey Regier, Michael Cole, Michael Jordan, and Nir Yosef. A deep generative model for gene expression profiles from single-cell RNA sequencing. January 2018. arXiv: 1709.02082

[4] Christopher H. Grønbech, Maximilian F. Vording, Pascal N. Timshel, Capser K. Sønderby, Tune H. Pers, and Ole Winther. scVAE: Variational auto-encoders for single-cell gene expression datas. page 318295, May 2018. doi:10.1101/318295

[5] Gökcen Eraslan, Lukas M. Simon, Maria Mircea, Nikola S. Mueller, and Fabian J. Theis. Single cell RNA-seq denoising using a deep count autoencoder. page 300681, April 2018. doi:10.1101/300681

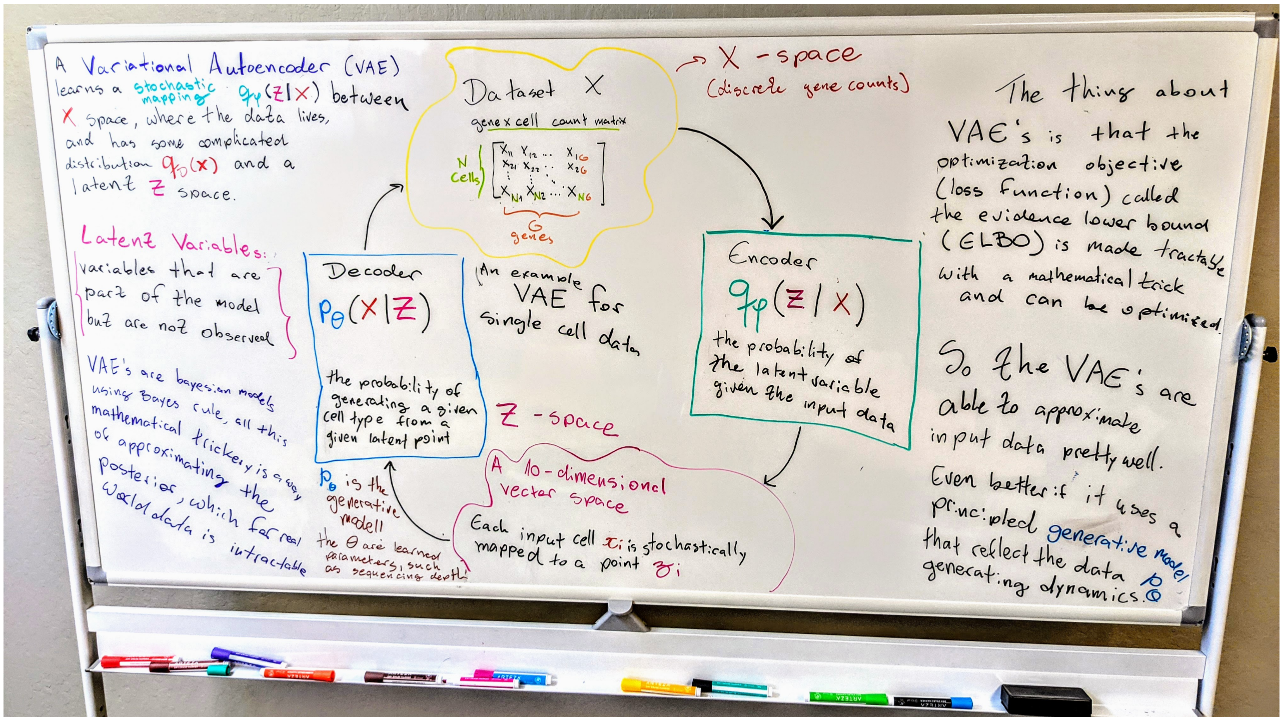

A Variational Autoencoder (VAE) learns a stochastic mapping $q_\varphi(Z|X)$ between X space, where the data lives, and has some complicated distribution $q_D(X)$ and a latent Z space.
Latent Variables: variables that are part of the model but are not observed
VAE's are bayesian models using Bayes rule, all this mathematical trickery is a way of approximating the posterior, which for real world data is intractable
Dataset X
gene x cell count matrix
N Cells
X_{11} X_{12} ... X_{1G}
X_{21} X_{22} ... X_{2G}
X_{N1} X_{N2} ... X_{NG}
G genes
X -space
(discrete gene counts)
Decoder
$p_\theta(X|Z)$
the probability of generating a given cell type from a given latent point
p_θ is the generative modell
the θ are learned parameters, such as sequencing depth!
An example VAE for single cell data
Encoder
$q_\varphi(Z|X)$
the probability of the latent variable given the input data
Z -space
A 10-dimensional vector space
Each input cell x_i is stochastically mapped to a point z_i
The thing about VAE's is that the optimization objective (loss function) called the evidence lower bound (ELBO) is made tractable with a mathematical trick and can be optimized.
So the VAE's are able to approximate input data pretty well.
Even better if it uses a principled generative model p_θ that reflect the data generating dynamics.

Figure 2.1: A conceptual representation of the blocks in a VAE for scRNA-seq.

quantities of interest, such as gene expression levels. Comparing two cell groups amounts to simulating new cells from each group, and comparing the statistics of interest.

scvi-tools

Currently the most well developed and broadly used variational autoencoder model for single cell data is scVI, which is a model part of the scvi-tools framework scvi-tools.org[6]. The scvi-tools framework provides many utility functions for sampling from the learned latent space and performing bayesian hypothesis testing. Because the data generation model can be modified to reflect our assumptions about underlying processes, the scvi-tools framework can be extended to model other aspects of scRNA-seq data.

Currently there are many models implemented in scvi-tools framework: these include models tailored for performing cell type classification and label transfer across batches, modelling single cell protein measurements, performing gene imputation in spatial data, using a linear decoder to allow for interpretation of the learned latent space, and modelling chromatin accessibility data.

[6] Adam Gayoso, Romain Lopez, Galen Xing, Pierre Boyeau, Katherine Wu, Michael Jayasuriya, Edouard Melhman, Maxime Langevin, Yining Liu, Jules Samaran, Gabriel Misrachi, Achille Nazaret, Oscar Clivio, Chenling Xu, Tal Ashuach, Mohammad Lotfollahi, Valentine Svensson, Eduardo da Veiga Beltrame, Carlos Talavera-López, Lior Pachter, Fabian J. Theis, Aaron Streets, Michael I. Jordan, Jeffrey Regier, and Nir Yosef. scvi-tools: a library for deep probabilistic analysis of single-cell omics data. page 2021.04.28.441833, April 2021. doi:10.1101/2021.04.28.441833

scVI model description

Other chapters of this thesis utilized the scVI model, and a brief explanation of the scVI generative model is provided in this section.

scVI is based on a hierarchical Bayesian model that leverages variational inference parametrized by neural networks to approximate a complex posterior distribution - the technique introduced by VAEs. This use of VAEs leveraging the pytorch machine learning framework makes it a very efficient model to train even on large scRNA-seq datasets.

Let the output of a scRNAseq experiment be a matrix of counts with N rows (the number of cells) and G columns (the number of genes), where each entry x_{ng} is an integer representing how many transcripts of gene g where seen in cell n. scVI is a generative hierarchical Bayesian model for scRNAseq data with conditional distributions parametrized by neural networks for each gene . There are technical variables to account for different batches (s_n) and for library size (l_n, which can be interpreted as cell size or sequencing depth). Thus the number of networks being trained is $2 \cdot G \cdot K$, where K is the total the number of batches (datasets).

Conditional distribution $p\left(x_{ng} \mid z_n, l_n, s_n\right)$ is a zero-inflated negative binomial distribution (ZINB) to model the kinetics of stochastic gene expression with some entries replaced by zeros.

The neural networks f_w^g and f_h^g use dropout regularization and batch nomalization to model gene expression while accounting for library sizes and batch effects respectively. Each network typically has 3 fully connected-layers, with 128-256 nodes each. The activation functions are ReLU, exponential, or linear. f_w has a final softmax layer to represent normalized expected frequency of gene expression. Weights for some layers are shared between f_w and f_h.

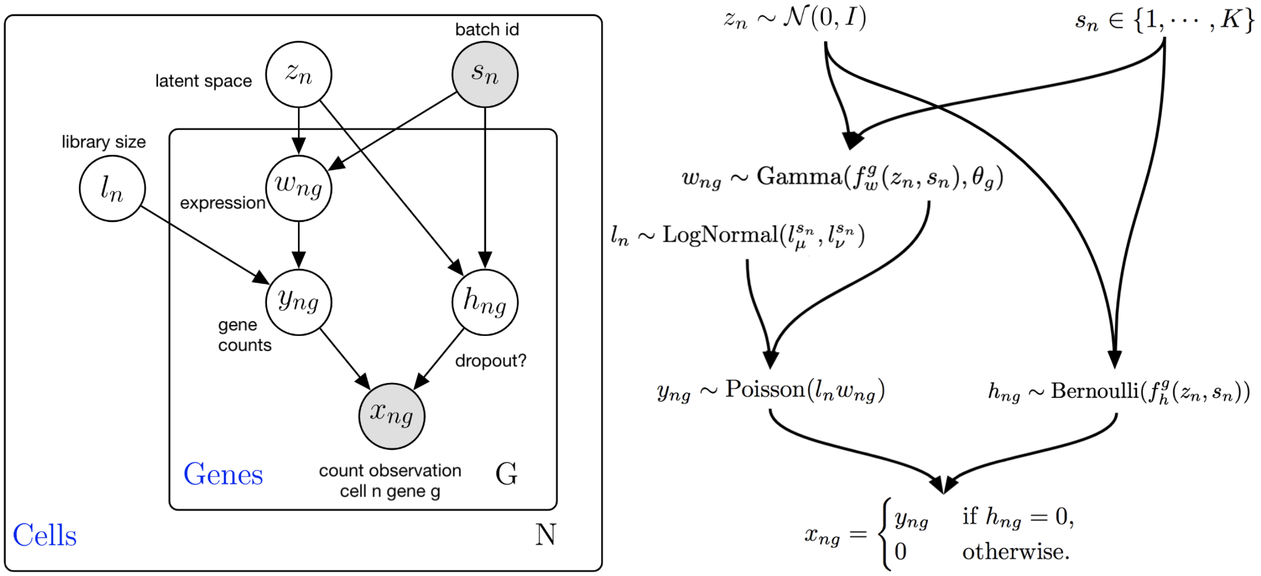

batch id
latent space
z_n
s_n
library size
l_n
w_{ng}
expression
y_{ng}
h_{ng}
gene counts
dropout?
x_{ng}
Genes
count observation cell n gene g
G
Cells
N
$z_n \sim \mathcal{N}(0, I)$
$s_n \in \{1, \cdots, K\}$
$w_{ng} \sim \text{Gamma}(f_w^g(z_n, s_n), \theta_g)$
$l_n \sim \text{LogNormal}(l_\mu^{s_n}, l_\nu^{s_n})$
$y_{ng} \sim \text{Poisson}(l_n w_{ng})$
$h_{ng} \sim \text{Bernoulli}(f_h^g(z_n, s_n))$
$x_{ng} = \begin{cases} y_{ng} & \text{if } h_{ng} = 0, \\ 0 & \text{otherwise.} \end{cases}$

Figure 2.2: The scVI graphical model with annotations

Table 2.1: The scVI model variables.

Variable	Description
$s_n \in \{1...K\}$	Batch annotation index.
$l_v^{s_n} = (l_\sigma^{s_n})^2 \in \mathbb{R}_+$	$l_\sigma^{s_n}$ is a technical effect variable modelling cell size variance in a batch dependent manner.
$l_\mu^{s_n} \in \mathbb{R}_+$	Technical effect variable modelling mean cell size in a batch dependent manner.
$l_n \sim \text{LogNormal}\left(l_\mu^{s_n}, l_v^{s_n}\right)$	Technical effect variable modelling library size. This can be interpreted as capturing both cell size and sequencing depth, the number of samples drawn from each cell.
$z_n \sim \mathcal{N}(0, I)$	Low-dimensional latent variable (10D default) random vector for cell n.
f_w^g	Neural network that map the latent space to gene expression. Typically has 3 fully connected-layers, with 128-256 nodes each. The activation functions are ReLU, exponential, or linear.
f_h^g	Neural network that decodes batch effects. Typically has 3 fully connected-layers, with 128-256 nodes each. The activation functions are ReLU, exponential, or linear.
$\theta_g \in \mathbb{R}_+$	A gene specific inverse dispersion parameter optimized during variational inference.
$w_{ng} \sim \text{Gamma}\left(f_w^g\left(z_n, s_n\right), \theta_g\right)$	Accounts for the stochasticity of gene g expressed in cell n.
$y_{ng} \sim \text{Poisson}\left(l_n w_{ng}\right)$	Underlying expression level for gene g in cell n.
$h_{ng} \sim \text{Bernoulli}\left(f_h^g\left(z_n, s_n\right)\right)$	Indicates whether a particular entry is a dropout due to technical effects.
$x_{ng} = \begin{cases} y_{ng} & \text{if } h_{ng} = 0 \\ 0 & \text{otherwise.} \end{cases}$	Observed gene expression, corresponds to one entry in the gene count matrix, the number of counts observed for gene g in cell n.

3 Principles for open source bioinstrumentation

This chapter is related to the published study:

Principles of open source instrumentation applied to the poseidon syringe pump system.

A. Sina Booeshaghi, Eduardo da Veiga Beltrame, Dylan Bannon, Jase Gehring, and Lior Pachter.

Scientific Reports 9, 12385 (2019).

doi: 10.1038/s41598-019-48815-9

Project blog post: liorpachter.wordpress.com/tag/poseidon/

Project website: pachterlab.github.io/poseidon

Author contributions: *J.G. conceived of the project and developed the initial design for the syringe pumps. A.S.B. designed the syringe pump system and microscope, and implemented the poseidon software. E.V.B. helped with the design of the poseidon system and oversaw hardware printing and design. A.S.B. and E.V.B. tested the poseidon system. J.G., A.S.B. and E.V.B. formulated the design principles. D.B. developed an initial version of the software. A.S.B., E.V.B., J.G. and L.P. wrote the manuscript.*

The rise of open source scientific instrumentation

As the open source movement continues to advance and mature, with free and open source software now being a staple of technology development, the same principles and ideals started being applied to hardware. The open source hardware movement [1] gained momentum in recent years due in large part to:

- The development of an ecosystem of open source electronics boards, driven largely by the Arduino Project and the Raspberry Pi Foundation.
- The rapid evolution of desktop 3D printers, which was enabled by the expiration of key patents and the RepRap academic project, which kickstated developing the first desktop 3D printers as open source designs[2].

Among open source hardware designs, a growing number of them are laboratory instrumentation[3]. Examples include systems for microscopy[4], fluorescence imaging[5], and micro-dispensers[6].

Open source hardware is particularly well suited for research, not only because of lower cost, but especially because their open nature enables the users to change and tweak the design to fit their own specialized needs. Another advantage of the open source ecosystem is that by virtue of having a large set of designs, software, and commonly used off-the-shelf parts that are shared across many designs, even when creating a new, instrument the user is rarely starting from scratch.

For example, the development of desktop 3D printers on the RepRap project borrowed heavily from standard software and hardware CNC (Computer Numeric Control) tool, and today most desktop 3D printers share the same control language as CNC tools, G-code. As the industry matured and the market kept growing, open source designs, electronics boards, software, and parts for 3D printers kept being published and improved. As a result, the market is now flooded with cheap and interoperable open source hardware and software that can be used to build generic gantry systems, not only 3D printers. This yields another cycle of innovation where these parts are adapted for new instruments.

But to be adopted beyond the creators lab, open source instruments must be easy to build, deploy, and utilize. In this project we created the poseidon system of syringe pumps and used the lessons we learned as an example to illustrate how a few design principles can facilitate adoption of an open source bioinstrument and help development of other projects.

[1] For a nice overview, read the book: Alicia Gibb, Steven Adabie, and Ed Baafi. *Building open source hardware DIY manufacturing for hackers and makers*. Addison-Wesley, Upper Saddle River, NJ, 2015. OCLC: 904585844

[2] See the original RepRap paper, before the editors got rid of the fantastic figure 13. It reads like prose, almost poetry: `https://reprap.org/mediawiki/images/d/da/Jones-et-al-paper.pdf`. The less exciting published version is available at Rhys Jones, Patrick Haufe, Edward Sells, Pejman Iravani, Vik Olliver, Chris Palmer, and Adrian Bowyer. RepRap – the replicating rapid prototyper. *Robotica*, 29(1), January 2011. doi: 10.1017/S026357471000069X

[3] Joshua M. Pearce. Building Research Equipment with Free, Open-Source Hardware. *Science*, 337(6100):1303–1304, September 2012. Publisher: American Association for the Advancement of Science; and Joshua M. Pearce. Cut costs with open-source hardware. *Nature*, 505(7485), January 2014. doi: 10.1038/505618d

[4] Andre Maia Chagas, Lucia L. Prieto-Godino, Aristides B. Arrenberg, and Tom Baden. The €100 lab: A 3D-printable open-source platform for fluorescence microscopy, optogenetics, and accurate temperature control during behaviour of zebrafish, Drosophila, and Caenorhabditis elegans. *PLOS Biology*, 15(7), July 2017. doi: 10.1371/journal.pbio.2002702

[5] Isaac Nuñez, Tamara Matute, Roberto Herrera, Juan Keymer, Timothy Marzullo, Timothy Rudge, and Fernán Federici. Low cost and open source multi-fluorescence imaging system for teaching and research in biology and bioengineering. *PLOS ONE*, 12(11), November 2017. doi: 10.1371/journal.pone.0187163

[6] C. J. Forman, H. Tomes, B. Mbobo, R. J. Burman, M. Jacobs, T. Baden, and J. V. Raimondo. Openspritzer: an open hardware pressure ejection system for reliably delivering picolitre volumes. *Scientific Reports*, 7(1), May 2017. doi: 10.1038/s41598-017-02301-2

The poseidon system

In this project we developed the poseidon system, a 3D printable system comprised of 3 syringe pumps and a microscope controller station.

The poseidon system takes advantage of components used in 3D to assemble and control the pumps, allowing for the construction of 3 pumps and the microscope controller station in less than an hour for under $400 using off-the-shelf and 3D printed components. Because the poseidon system is open source, it can be customized and the software adapted to new use cases. That stands in contrast to commercial pumps, which typically cost between $500 to $3000 a piece, and cannot be easily adapted to new use cases.

In true open source fashion, it's design drew from previously published designs for open source pumps by Wijnen 2014[7] and a scRNA-seq microfluidics station by Stephenson 2018[8].

All project files are available in the poseidon GitHub repository at github.com/pachterlab/poseidon, and documentation is hosted on the project website at pachterlab.github.io/poseidon/

In addition to documentation, the following materials are provided to users:

- Computer Aided Design (CAD) files of the 3D printed components.
- Controller software and a graphical user interface to control the pumps.
- Arduino firmware to send and receive motor commands from the controller.
- Bill of materials for sourcing and purchasing materials.
- Detailed assembly instructions of hardware components.
- Single click executable files for Mac, Windows, Linux, and Raspberry Pi OS.

[7] Bas Wijnen, Emily J. Hunt, Gerald C. Anzalone, and Joshua M. Pearce. Open-Source Syringe Pump Library. *PLOS ONE*, 9(9), September 2014. doi: 10.1371/journal.pone.0107216

[8] William Stephenson, Laura T. Donlin, Andrew Butler, Cristina Rozo, Bernadette Bracken, Ali Rashidfarrokhi, Susan M. Goodman, Lionel B. Ivashkiv, Vivian P. Bykerk, Dana E. Orange, Robert B. Darnell, Harold P. Swerdlow, and Rahul Satija. Single-cell RNA-seq of rheumatoid arthritis synovial tissue using low-cost microfluidic instrumentation. *Nature Communications*, 9(1), February 2018. doi: 10.1038/s41467-017-02659-x

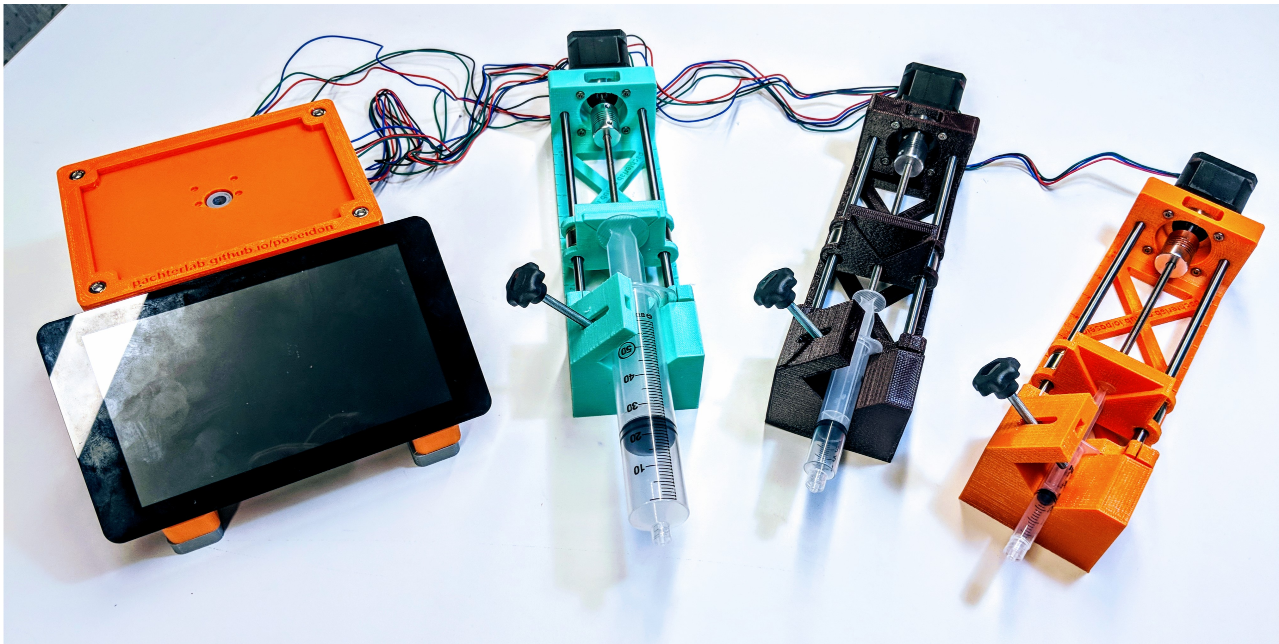


Figure 3.1: The assembled poseidon system with 3 sysringe pumps and microscope station.

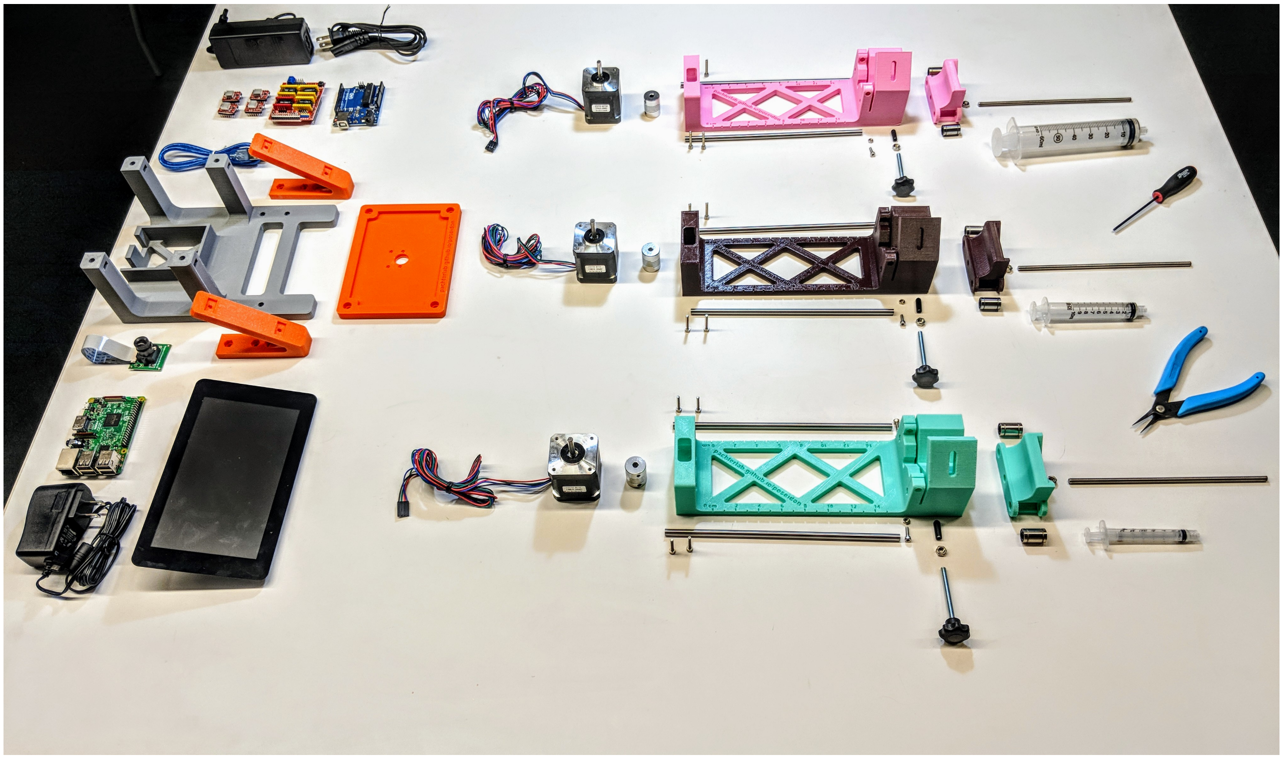


Figure 3.2: Exploded view of all the components needed for assembling three pumps and the microscope controller station.

Principles of open source bioinstrumentation

While cost is often the main motivation for developing and adopting open source instrumentation, other factors help drive the adoption and success of a design. In particular, it is important to keep in mind that in a lab there are always two kinds of users:

- Those who want to take a design and use it in a straightforward manner in their projects.
- Those who want to tweak, improve, and adapt designs to their needs and new use cases.

A successful open source instrumentation system design that appeals to the needs of both audiences benefits from keeping a few important principles in mind during development.

- **Functionality:** developing for an application.
- **Robustness:** designing with variation in mind.
- **Simplicity:** making it easy to source, build, and operate.
- **Modularity:** building independent and individual units.
- **Benchmarking:** validating with standard protocols.
- **Documentation:** describing the design completely.

Functionality: Being good enough for the application at hand

In engineering, a functional requirement defines a specific function that a hardware or software system must implement. The notion of being "good enough" attempts to capture the many tradeoffs that can be made during the development. If the instrument solves the problem at hand, but does so at the cost of one of the many different factors such as precision, accuracy, speed, cost or complexity, then the developer must decide if those tradeoffs are acceptable given the application.

For the poseidon pumps, there were the followiing functional requirements, focused on using the system for microfluidic applications:

1. The syringe pumps should be precise enough to make monodisperse emulsions on droplet generation microfluidics chips, with flow rates on the order of a few microliters per hour.
2. There should be a microscope with sufficient magnification to verify the quality of the emulsions.

3. The control software should be able to run at least 3 pumps independently, with a simple interface allowing the user to easily change flow rates, syringe type or diameter, and perform a flow rate gradient.

4. The software should also enable using the microscope.

While these are the minimum requirements, in order to enable other kinds of experimental setups we sought to ensure the pumps operate reliably with flow rates ranging from a few ul per hour up to several ml/min, as needed to perform a protein purification experiment for example. For microfluidics experiments, a high speed camera is useful to inspect chip operation, but is expensive and not essential to running common setups. Having a microscope, however, is essential to ensure the chip is running and that the output is as expected. The requirements are not stringent: a 20X amplification is sufficient for most use cases, and this allowed experimenting with a variety of very cheap cameras for the poseidon microscope, and ultimately settling on a cheap USB microscope camera.

Robustness: Tolerating variations in manufacturing and operation

Nobody wants to deal with a finicky instrument. This means not only the possibility of failure during operation, but also having a construction process that is easy to get right the first time. The scientist in the lab knows too well that, while it may be hard to get something to work, it can be much harder to get it to work reliably. The same is true when developing open source designs, the bulk of the work often goes into making a design simpler, more reliable, and easier for other people to use. Ensuring robustness takes time because a lot of it demands attention to small details, and repeated testing.

For example, much of open source hardware relies on 3D printed components which are cheap to make and very flexible, but can introduce some variability from printer to printer. Thus it is desirable to ensure designed pieces afford some mechanical tolerance. With the poseidon pumps an unforeseen hardware issue was that the mechanism used to secure the linear rods in place was not holding them well - when there was much resistance to sliding the syringe plunger, the rods would be displaced and the printed plastic body would bend. A redesign of the printed body was necessary to solve this issue, which was only identified after we started using the pumps in our own experiments.

On the software side, robustness demands both testing to ensure correct functionality, but also reducing user operation errors by offering a well tested user interface. Once the poseidon pumps started being used for experiments in our lab and other labs, usability issues became apparent. For example, one version of the software by default configured the stepper motors to use a different microstepping than the hardware had configured, which the users only found out by realizing the flow rates were wrong in the middle of a run. Usage of the software during an experiment also revealed small usability issues that had to be corrected, such as using a drop-down menu for choosing the flow direction instead of a toggle button.

Simplicity: Making it easy to source parts, build and operate

The user has a limited amount of patience and effort to spend getting the instrument to work. Simplifying the process of buying parts, construction and operation of a device is fundamental for driving adoption.

For any open source hardware design, sourcing components should be as easy as possible. During development, using off-the-shelf components should always be a top priority, and incorporating harder to find parts in a design needs to be weighed carefully. To facilitate the process of sourcing parts, a bill of materials should be up to date and accurate regarding where a component can be purchased from. When trying to replicate open source designs, purchasing parts from several different vendors is often a friction point for the user. While sometimes a specialized item is only sold by one vendor, it is ideal to have more than one option.

For the poseidon project, it was a design consideration that users should be able to purchase all the components from Amazon. This was possible because most of the components of the project are used in desktop 3D printers, which enjoys a thriving online market.

During assembly, it is important to avoid dealing with specialized equipment when building an instrument. Even soldering a circuit board may put off new adopters. While using specialized assembly processes is sometimes unavoidable, striving for a simple assembly process is an important consideration. An excellent way to assess the difficulty of assembly is to have people unfamiliar with the project replicate the assembly using only the documentation available.

With the poseidon pumps it was possible to design around most of these constraints, and we have verified that assembly of a single pump by a new user following the instruction video usually takes less than 15 minutes, requiring only pliers and screwdrivers.

This aim also applies to software, where installation and operation should be as simple as possible. Dependency on external software libraries should be minimized, as this is a common source of trouble for the user, who needs to install additional software packages and ensure all versions match the ones required by your software. From the user perspective, having a single binary executable file for the software is ideal.

For the poseidon project, it was possible to compile the GUI Python scripts into binary executable files for Mac, Windows and Unix. The custom poseidon Arduino firmware needs to be flashed by following the official instructions. If using a Raspberry Pi to operate the poseidon system, installation requires flashing a SD card with the official version of the Raspbian OS image.

Modularity: Designing for independent and interoperable units

Because some users will want to adapt a design to new use cases, it is important to consider how easily a design can be taken apart, tweaked and re-purposed. A design composed of modules that plug into each other is easier to re-purpose than a tightly integrated device. When a design is not modular, adapting one feature of the instrument for a new use may require re-designing it from scratch.

A design is also made easier to tweak when common and standardized parts and connectors are used. Standardization is ubiquitous in both open source hardware and software projects. For example, open source desktop 3D printers use a common set of screws, rods, extruders and electronics, and are often variations of a few common, popular and proven designs. The standardization of 3D printer parts designs means in turns that they can be readily adapted for new use cases. In fact, almost all of the components for the poseidon pumps are used in open source desktop 3D printers.

For example, some users of the poseidon pumps might want to run more than 3 pumps at once. This tweak is streamlined by the fact that the pumps are an independent unit, and because they use the common NEMA17 stepper motor and connector they can be controlled with a variety of electronics boards.

Benchmarking: Validating system use cases

The user needs to know if the instrument design is applicable for the problem at hand. Thus it is important to describe protocols where the instrument was successfully applied, and provide some benchmarking on instrument performance. Open source instruments may not always perform as well as commercial systems, but they are still good enough for many applications. Direct comparison with commercial instruments can be important in inspiring confidence for the user, but it is even more important to clearly identify known shortcomings of a design.

Documentation: Describing the system to other humans

Even a simple assembly can be made hard without instructions, and the quality of the instructions provided is the first thing users will notice when deciding whether to tackle a new project. Videos, photographs and descriptions are fundamental for showcasing a design and attracting users, and should quickly convey how much effort the user should expect to invest. For assemblies, videos are often the most helpful piece of documentation for the user, and do not need to take much time and effort to produce.

While some users are willing to spend time tinkering with your design, sufficient documentation makes it faster and easier to understand a design, and makes it more attractive for new contributors to improve on it. Users who want to tweak a design will also benefit from understanding your design decisions - both those motivated by technical considerations, and those motivated by user feedback. How to implement each feature is the result of thought and iteration from the designer, but what is learned may not be apparent by just looking at the design. Documenting lessons learned and why design features were implemented a certain way is important - sometimes tweaks that seem to be an improvement will create a failure mode that is not apparent.

For the poseidon pumps, a 10 minute video recorded on a cellphone showing the assembly process and hosted on YouTube and linked on the poseidon webpage is perhaps the most helpful part of the instructions. In making the poseidon documentation website, we also strove to use clearly labeled photos of the hardware with short written instructions, as this makes it easier for prospective users to grasp the design and expected time investment at a glance.

The development and documentation of open source projects benefits tremendously from making use of version control repositories, which streamline remote collaboration and tracking development. For the poseidon project, we used the online repository GitHub, which allows for version control and documentation of each change made and makes it simple to create attractive user guides and documentation, as can be seen on pachterlab.github.io/poseidon.

4 *Efficient combinatorial bead barcoding*

This chapter summarizes the content of the pending patent:

US20200102556A1 - Efficient combinatorial bead barcoding.

Inventors: Eduardo da Veiga Beltrame, Jase Gehring, Akshay Tambe, Lior S. Pachter and Taleen Dilanyan.

Available at: patents.google.com/patent/US20200102556A1/en

Author contributions: *E.B. helped in initial development of the technology prior to filing of the provisional patent.*

Overview

DNA barcoded beads are essential components of droplet based methods for scRNA-seq. Those beads must have a unique DNA barcode sequence with billions of copies within each bead, but which are distinct between two beads. They must also have a capping sequence that will hybridize with the desired RNA transcripts - commonly those are poly T sequences for capturing polyadenylated mRNA transcripts.

Building on the published inDrops protocol[1] for scRNA-seq, the aim of this project was the development of a method for combinatorial barcoding of hydrogel beads that uses the concept of Primer Exchange Reaction (PER) introduced by Kishi 2018[2].

PER relies on a phenomena called branch migration to reuse a catalytic piece of DNA that forms a hairpin. The catalytics cycle starts with a toe-hold primer that has a short complementary sequence to which the hairpin anneals to, at the 3′ end of the primer that will be extended. A DNA polymerase with strand displacing activity extends the primer and displaces the rest of the hairpin, but it stops after a few bases when it reaches a blocking sequence. This blocking sequence can be a modified base, or simply a missing nucleotide. In the experiments performed only A, T and C were present in solution, so GGG was the stopping sequence. Once stalled the polymerase eventually falls off, and the hairpin complementary sequence competes with the newly synthesized sequence in a process called branch migration. When the hairpin fully anneals back with itself, the remainder toehold is not enough to reliably keep it in place, and it eventually falls off, free to catalyze another reaction. This catalytic activity of the DNA hairpin is extremely efficient, and a single hairpin can catalyze hundreds to thousands of reactions, dramatically lowering the cost of DNA in comparison to barcoding methods using ligation reactions.

We performed PER in a hydrogel substrate of small hydrogel beads, and dubbed the method iPER, for immobilized Primer Exchange Reaction. The method significantly lowers labour and reagent costs for producing barcoded beads for scRNA-seq, and enables users to add any capture sequence to the beads, not only poly T sequences. This opens up the possibility of performing targeted scRNA-seq. For example, panels of capture sequences could be designed to enrich a library for transcription factors, non coding RNAs, or bacterial genes.

[1] Allon M. Klein, Linas Mazutis, Ilke Akartuna, Naren Tallapragada, Adrian Veres, Victor Li, Leonid Peshkin, David A. Weitz, and Marc W. Kirschner. Droplet Barcoding for Single-Cell Transcriptomics Applied to Embryonic Stem Cells. *Cell*, 161(5):1187–1201, May 2015. doi:10.1016/j.cell.2015.04.044

[2] Jocelyn Y. Kishi, Thomas E. Schaus, Nikhil Gopalkrishnan, Feng Xuan, and Peng Yin. Programmable autonomous synthesis of single-stranded DNA. *Nature Chemistry*, 10(2), February 2018. doi:10.1038/nchem.2872

iPER bead production protocol

The final iPER bead barcode structure is shown in the figure below. The PER reaction uses over a hundred times less DNA compared to previous primer extension reactions, requires no washing between steps, and is highly efficient and specific. This is because in this reaction hairpins with specific sequences act as catalysts for DNA extension and are not consumed in the process, as shown in the figure below.

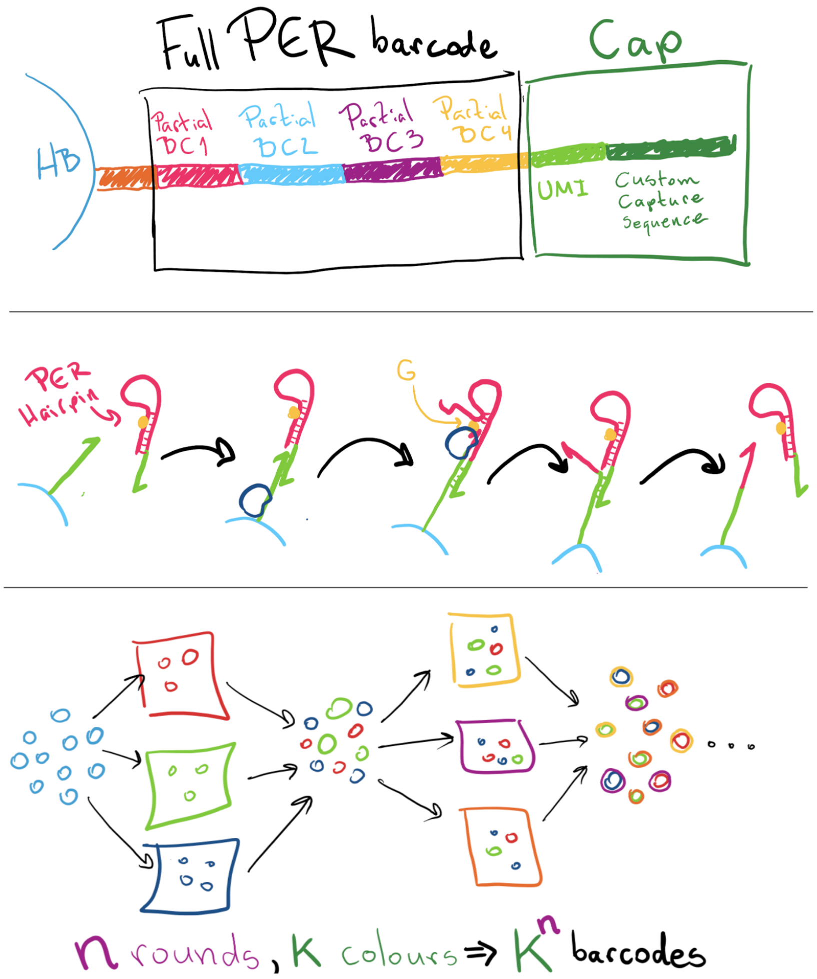


Figure 4.1: Overview of the iPER beads manufacturing process.

Top: Depiction of the barcoded bead. Each hydrogel bead is coated with billions of copies of the same single stranded DNA sequence. The single stranded DNA attaches to the hydrogel bead by using an acrydite moiety, which is crosslinked with the hydrogel. Partial barcode sequences 1-4 are then combinatorially added in 4 extension steps using a split-pool process. As a result, each bead is coated with DNA that has the same full barcode, but no two beads have the same partial barcode combination. Finally, a common capping sequence is added to all beads. which includes a unique molecular identifier (UMI) region of 12bp of random sequence in every DNA strand, to disambiguate identical transcripts and avoid PCR amplification biases.

Mid: Schematic of the Primer Exchange Reaction. First, a PER hairpin with a complementary toehold sequence (green) hybridizes with the oligo to be extended. In the buffer of the extension reaction, only A, T and C nucleotides are present, without G. The hairpin will be extended for about 13 base pairs until a repeat sequence of GGG. After this partial extension, branch migration happens, and the hairpin eventually falls off, being free to catalyze another extension.

Bottom: Schematic of the split-pool barcoding. After n extension rounds with k barcodes in each round, the resulting sample can have up to k^n barcodes.

My main contribution to this project was the development and validation of the experimental iPER hydrogel bead production protocol. The three main steps of the protocol are briefly described below. I have not participated in the subsequent experiments aiming at validating using iPER beads for scRNA-seq.

iPER bead production steps

The iPER bead production protocol can be divided into three main steps:

Step 1) Production of hydrogel beads (HB).

Using a microfluidic droplet generator device, a monodisperse HB in oil emulsion is generated. The HB are about 60um in diameter and use a redox-cleavable gel that has an acrydite modified DNA primer crosslinked with it to serve as the initial toehold for PER. The emulsion is incubated overnight for complete polymerization of the HB, and then the HB are transferred to an aqueous buffer for barcoding. The HB manufacturing process was adapted from the inDrops protocol.

Instead of using a standard acrylamide/bis acrylamide gel, we replace the bisacrylamide with N,N-bis(acryloyl)cystamine, which has a disulfide bond and allows the gel to be dissolved using a reducing agent such as DTT. To release the oligos attached to the bead we incorporated a dU (deoxyuridine) base, which allows us to cleave the DNA with the USER enzyme.

Step 2) Combinatorial barcoding using split pool and PER.

Barcode extension in an efficient and cheap manner is one of the key challenges in manufacturing barcoded hydrogel beads. To combinatorially obtain a large number of barcodes, a split-pool strategy is used. On each step, the sample is split amongst each of the k labeling reactions. It is then recombined, mixed and split again into k labeling reactions. After n extension rounds, the resulting sample can have up to k^n barcodes. For example, by using 72 unique barcodes over 4 rounds, 26.8 million unique sequences can be obtained.

Step 3) Capping of beads with the desired capture sequence.

To cap the barcoded beads a standard DNA extension reaction is used to add arbitrary capping sequences. After capping, the beads are enzymatically treated to remove incomplete barcodes. Finally, they are washed in a denaturing buffer to remove the second strand, leaving only single stranded DNA on them.

5 Single cell studies database

This chapter is related to the published study:

A curated database reveals trends in single-cell transcriptomics

Valentine Svensson, Eduardo da Veiga Beltrame, Lior Pachter

Database, 2020

doi: 10.1093/database/baaa073

Database available at: nxn.se/single-cell-studies/

Code to reproduce paper analysis with up-to-date data: github.com/vals/single-cell-studies

Author contributions: *V.S. conceived and started the project. E.B. assisted with data collection. E.B. and V.S. analyzed the data. E.B., V.S. and L.P. and wrote the manuscript. Carlos Talavera-López provided helpful feedback on the manuscript.*

Overview

Single cell RNA sequencing has exploded in popularity since the first streamlined high throughput methods were introduced around 2015. Currently, dozens of studies with new experimental data being published or made available as preprints every month. To facilitate discovery with published single-cell transcriptomics data, we assembled a near exhaustive, manually curated database of single-cell transcriptomics studies with key information: descriptions of the type of data and technologies used, along with descriptors of the biological systems studied.

The database focuses on tracking single cell transcriptomics studies, rather than primary data, for which there already are many initiatives. The compilation of such a database required us to read and manually curate large numbers of publications, which we indexed according to publication and authors. It allows researchers interested in specific tissues to rapidly identify relevant studies. By virtue of providing a comprehensive overview of the field, the database can highlight understudied tissues and facilitate citation of previous work when performing follow-up experiments. The database tracks metadata applicable to most studies, such as the number of cell types identified, and protocols used. Below we showcase up-to-date figures with the analysis we provide in the paper.[1].

[1] The analysis can be launched with one click directly on Google colab notebooks with this link:`https://colab.research.google.com/github/vals/single-cell-studies/blob/master/Example%20analysis.ipynb`

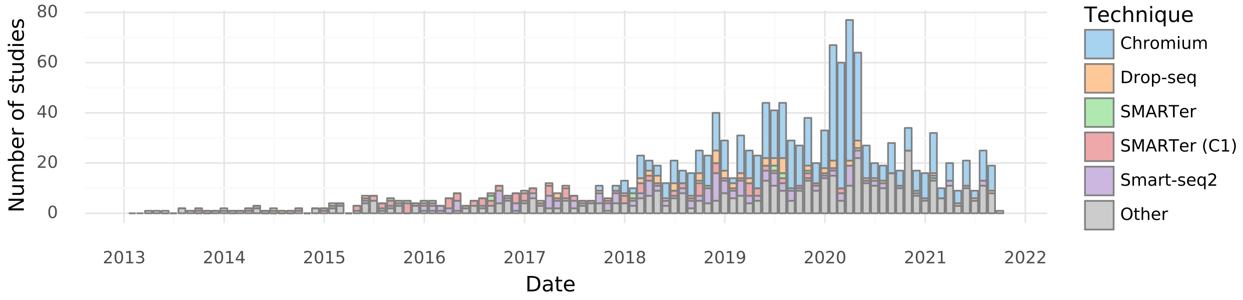


Figure 5.1: Number of studies reported over time, stratified by most popular technologies

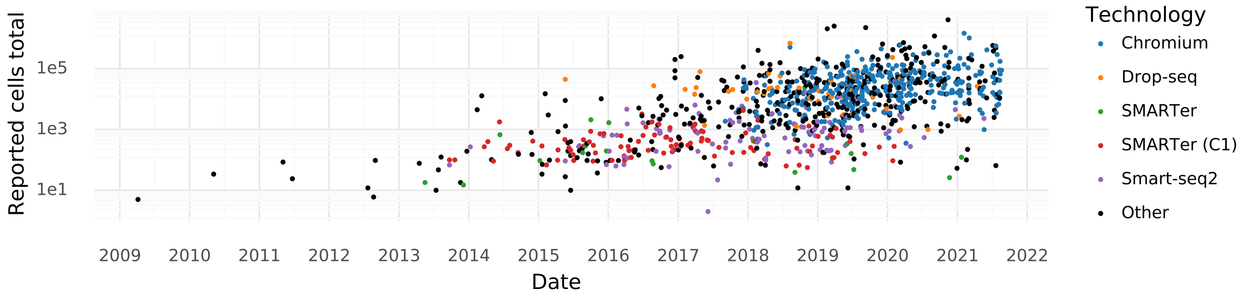


Figure 5.2: Cells reported per study over time. It shows the total number of reported cells in each study, colored by most popular technologies.

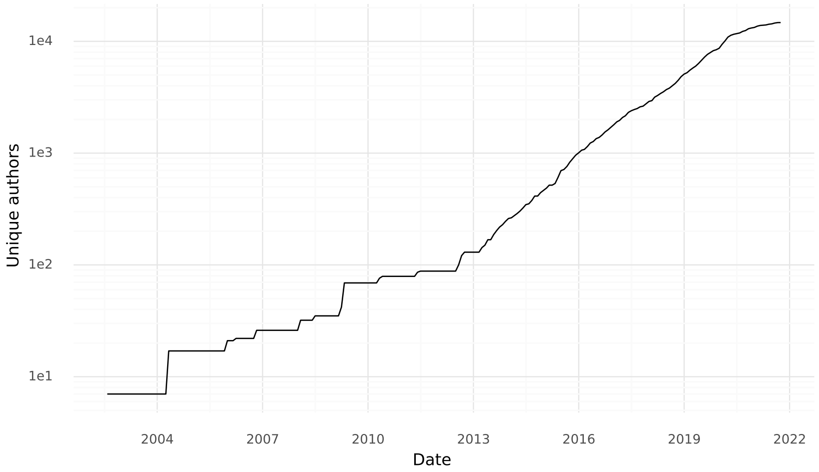


Figure 5.3: Unique authors in the database over time.

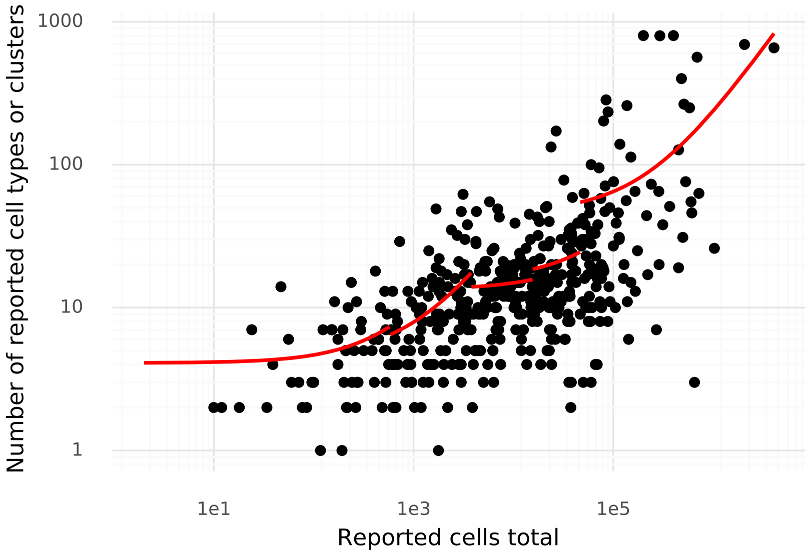


Figure 5.4: Number of total distinct cell types or clusters reported, compared with total cells reported. This suggests a trend where a new cluster or cell type will be defined for every few thousand cells sequenced.

6 kallisto | bustools

This chapter is related to the published study:

Modular, efficient and constant-memory single-cell RNA-seq preprocessing

Páll Melsted, A. Sina Booeshaghi, Lauren Liu , Fan Gao, Lambda Lu, Joseph Min, Eduardo da Veiga Beltrame , Kristján Eldjárn Hjörleifsson , Jase Gehring, and Lior Pachter

Nature Biotechnology. 2021 April 1.

doi: 10.1038/s41587-021-00870-2

Author contributions: *P.M., A.S.B., L.L. (Liu), and L.P. developed the algorithms for bustools and P.M., A.SB., and L.L. (Liu) wrote the software. A.S.B. conceived of and performed the UMI and barcode calculations motivating the algorithms. F.G. implemented and performed the benchmarking procedure, and curated indices for the datasets. A.S.B. and E.B. designed and produced the comparisons between Cell Ranger and kallisto. L.L. (Lu) investigated in detail the performance of different workflows on the 10k mouse neuron data and produced the analysis of that dataset. A.S.B. designed the RNA velocity workflow and performed the RNA velocity analyses. K.H. developed and investigated the effect of reference transcriptome sequences for pseudoalignment. JG interpreted results and helped to supervise the research. A.S.B planned, organized and made figures. A.S.B., E.B., P.M. and L.P. planned the manuscript. A.S.B. and L.P. wrote the manuscript.*

The benchmarking and reasoning behind the workflow is also described in a series of blog posts by Lior Pachter:

1. `https://liorpachter.wordpress.com/2019/06/21/near-optimal-single-cell-rna-seq-pre-processing/`.
2. `https://liorpachter.wordpress.com/2019/06/21/single-cell-rna-seq-for-dummies/`.
3. `https://liorpachter.wordpress.com/2019/06/21/how-to-solve-an-np-complete-problem-in-linear-time/`.
4. `https://liorpachter.wordpress.com/2019/06/24/rotating-the-knee-plot-and-related-yoga/`.
5. `https://liorpachter.wordpress.com/2019/07/01/high-velocity-rna-velocity/`.

Overview of the kallisto | bustools workflow

Rapid growth of different scRNA-seq methods, multiple separate computational pipelines to process the FASTQ files and produce the gene count matrix were developed, but usually only focused on a single method. The kallisto | bustools workflow was developed as an extension of the popular and efficient kallisto program that Harold Pimentel, Páll Melsted and Lior Pachter had developed for bulk RNA sequencing[1].

In version 0.45 kallisto introduced the capacity to process single cell RNA-seq data and output the result in the BUS[2] (Barcode-UMI-Set) file format. By using the BUS format the workflow offers modularity and flexibility, and a large part of the effort going into this project was the creation of documentation and benchmarking.

The BUS file can be processed with BUStools, a suite of tools for working with BUS files. The BUS file format retains barcode and UMI sequence information while discarding sequence information. It keeps only the mapping of equivalence classes to sets of transcripts. This reduces final file size while making possible the development of modular and technology-specific downstream workflows.

The workflow is modular because after each operation an intermediary file can be produced with the BUS format that was conceived to store single cell RNA sequencing data. This confers flexibility, as each step can be individually tweaked, and allows creating new workflows for different use cases using the same tools. This was demonstrated by creating workflows to process single-nucleus RNA-sequencing data, to perform RNA velocity, and to process multiplexed samples.

The kallisto | bustools workflow is extensively documented in the website kallistobus.tools. To demonstrate the modularity and flexibility of workflows using BUS files there are several example notebooks in Python and R with tutorials on for downstream processing and analysis with datasets generated from multiple technologies. Having clear documentation and tutorials is often is an underappreciated aspect of the success and adoption of software projects, and we strove to create extensive tutorials for the kallisto | bustools workflow.

Additionally we performed extensive benchmarking of the workflow, assessing both it's speed and performance, but also how the results fared in comparison with other pre-processing tools. When preprint was first made available, no other computational workflow for processing scRNA-seq data had published any kind of benchmarking against other methods, a remarkably common occurrence in the scRNA-seq field, and fortunately something that is changing.

[1] Nicolas L. Bray, Harold Pimentel, Páll Melsted, and Lior Pachter. Near-optimal probabilistic RNA-seq quantification. *Nature Biotechnology*, 34(5), May 2016. doi: 10.1038/nbt.3519

[2] Páll Melsted, Vasilis Ntranos, and Lior Pachter. The barcode, UMI, set format and BUStools. *Bioinformatics*, 35(21):4472–4473, November 2019. doi: 10.1093/bioinformatics/btz279

How pseudoalignment works

The kallisto RNA-seq quantification program uses pseudoalignment (meaning exact matches) to map reads to a reference. It produces a list of transcripts that are compatible with each read while avoiding alignment of individual bases. Fast and accurate pseudoalignments of reads to a transcriptome is obtained using fast hashing of k-mers together with the transcriptome de Bruijn graph.

An index is constructed by creating the Transcriptome de Bruijn Graph, where nodes are k-mers, each transcript corresponds to a path and the path cover of the transcriptome induces a k-compatibility class for each k-mer.

Conceptually, the k-mers of a read are hashed (black nodes) to find the k-compatibility class of a read. Skipping uses the information stored in the graph to skip k-mers that are redundant due to having the same k-compatibility class. The k-compatibility class of the read is determined by taking the intersection of the k-compatibility classes of its constituent k-mers

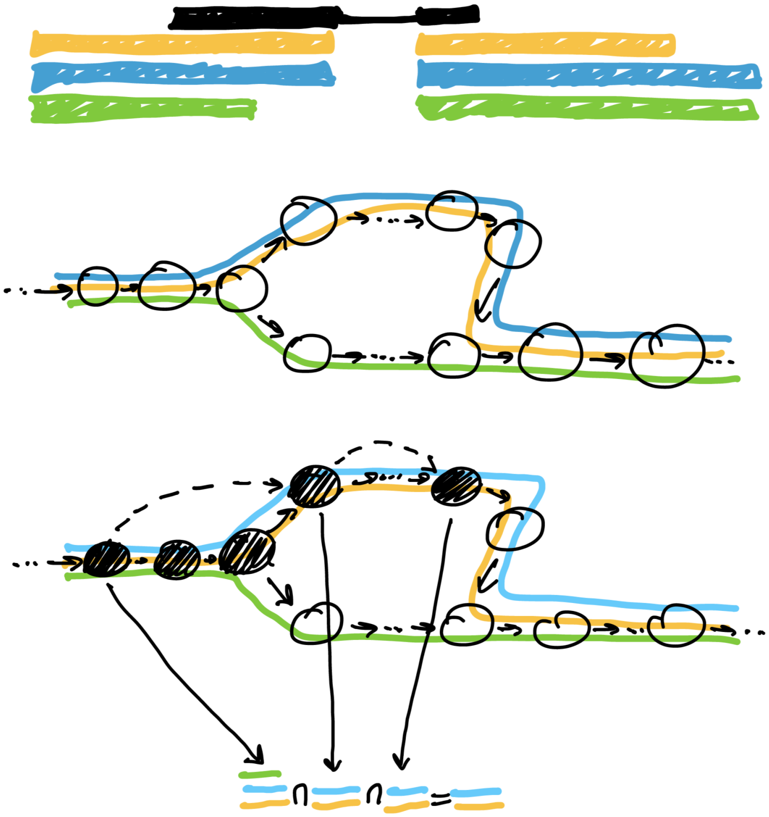


Figure 6.1: **Top:** An example of a read (in black) and three overlapping transcripts with exonic regions shown. **Middle:** Each transcript corresponds to a path and the path cover of the transcriptome k-mers, short strings that match a transcript (represented by empty circles). **Bottom:** The theoretical k-mers for a given gene that are compatible with the k-mers present in a read (filled circles) give the possible set of transcripts that a read could belong to (blue and yellow here).

Benchmarking

The kallisto | bustools was primarily benchmarked against CellRanger[3] (the most popular scRNA-seq processing workflow, an effective popular gold standard) using a selection of 20 datasets of 10x v2 and v3 chemistry that included all the most commonly surveyed organisms with scRNA-seq: mouse, human, *Drosophila*, rat, zebrafish, *C. elegan*s and *A. thaliana*. The frequently seen situation is that people will test their method on a few datasets that are of interest to what they are working on, and if it works apparently well, it will be described as if applicable universally. This is true across the board: from pre-processing tools such as aligners, to (especially) downstream analysis pipelines.

At the time the benchmarking performed in this project was the most extensive with regards to pre-processing tools. To benchmark processing times for kallisto bus, several datasets generated with multiple technologies were processed with kallisto bus and at least one other existing pipelines, including CellRanger, Salmon Alevin , and the original CEL-Seq[4], Drop-Seq[5] and inDrops[6] workflow. Datasets processed are listed below, together with accession number. For most comparison the 20 datasets were processed with four tools: kallisto bus, salmon alevin[7], CellRanger and STAR[8].

The figures below outline some interesting benchmarking visualizations that did not make it to the final publication.

We found that the kallisto | bustools workflow was extremely fast and used considerably less memory than STAR and CellRanger, both of which perform full alignment, which more computationally intensive. Typical memory usaged ranged between 4GB to 12GB, with processing times usually being under one hour in a desktop computer, enabling the processing of scRNA-seq datasets without access to computing clusters, or re-processing with different parameters. The project described in the next chapter leveraged the speed of the kallisto |bustools workflow to reprocess the same dataset dozens or hundreds of times.

[3] Grace X. Y. Zheng, Jessica M. Terry, Phillip Belgrader, Paul Ryvkin, Zachary W. Bent, Ryan Wilson, Solongo B. Ziraldo, Tobias D. Wheeler, Geoff P. McDermott, Junjie Zhu, Mark T. Gregory, Joe Shuga, Luz Montesclaros, Jason G. Underwood, Donald A. Masquelier, Stefanie Y. Nishimura, Michael Schnall-Levin, Paul W. Wyatt, Christopher M. Hindson, Rajiv Bharadwaj, Alexander Wong, Kevin D. Ness, Lan W. Beppu, H. Joachim Deeg, Christopher McFarland, Keith R. Loeb, William J. Valente, Nolan G. Ericson, Emily A. Stevens, Jerald P. Radich, Tarjei S. Mikkelsen, Benjamin J. Hindson, and Jason H. Bielas. Massively parallel digital transcriptional profiling of single cells. *Nature Communications*, 8(1), January 2017. doi: 10.1038/ncomms14049

[4] Dominic Grün, Anna Lyubimova, Lennart Kester, Kay Wiebrands, Onur Basak, Nobuo Sasaki, Hans Clevers, and Alexander van Oudenaarden. Single-cell messenger RNA sequencing reveals rare intestinal cell types. *Nature*, 525(7568), September 2015. doi: 10.1038/nature14966

[5] Evan Z. Macosko, Anindita Basu, Rahul Satija, James Nemesh, Karthik Shekhar, Melissa Goldman, Itay Tirosh, Allison R. Bialas, Nolan Kamitaki, Emily M. Martersteck, John J. Trombetta, David A. Weitz, Joshua R. Sanes, Alex K. Shalek, Aviv Regev, and Steven A. McCarroll. Highly Parallel Genome-wide Expression Profiling of Individual Cells Using Nanoliter Droplets. *Cell*, 161(5):1202–1214, May 2015. doi: 10.1016/j.cell.2015.05.00

[6] Allon M. Klein, Linas Mazutis, Ilke Akartuna, Naren Tallapragada, Adrian Veres, Victor Li, Leonid Peshkin, David A. Weitz, and Marc W. Kirschner. Droplet Barcoding for Single-Cell Transcriptomics Applied to Embryonic Stem Cells. *Cell*, 161(5):1187–1201, May 2015. doi:10.1016/j.cell.2015.04.044

[7] Avi Srivastava, Laraib Malik, Tom Smith, Ian Sudbery, and Rob Patro. Alevin efficiently estimates accurate gene abundances from dscRNA-seq data. *Genome Biology*, 20(1):65, March 2019. doi: 10.1186/s13059-019-1670-y

[8] Alexander Dobin, Carrie A. Davis, Felix Schlesinger, Jorg Drenkow, Chris Zaleski, Sonali Jha, Philippe Batut, Mark Chaisson, and Thomas R. Gingeras. STAR: ultrafast universal RNA-seq aligner. *Bioinformatics*, 29(1):15–21, January 2013. doi: 10.1093/bioinformatics/bts635

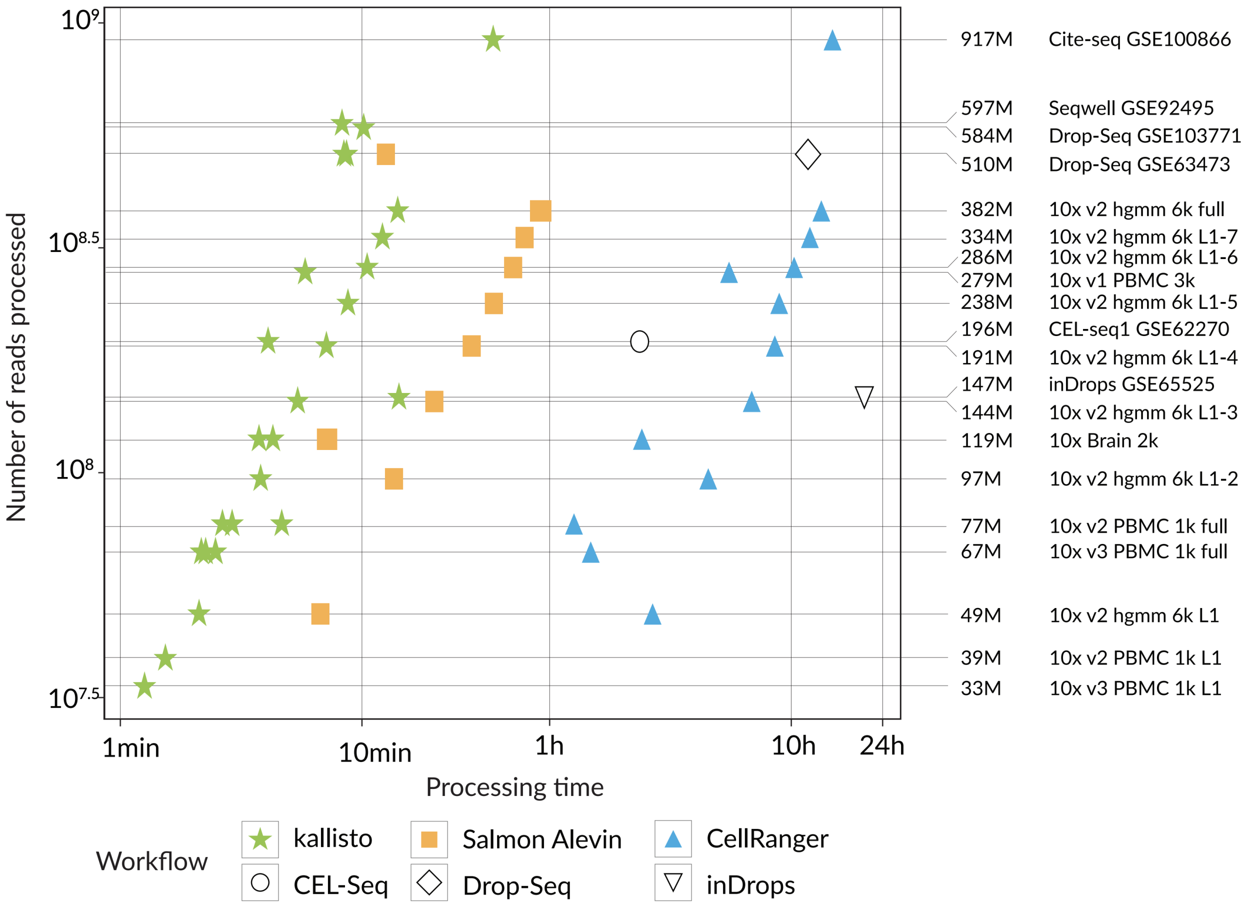


Figure 6.2: Processing times for all 20 datasets benchmarked.

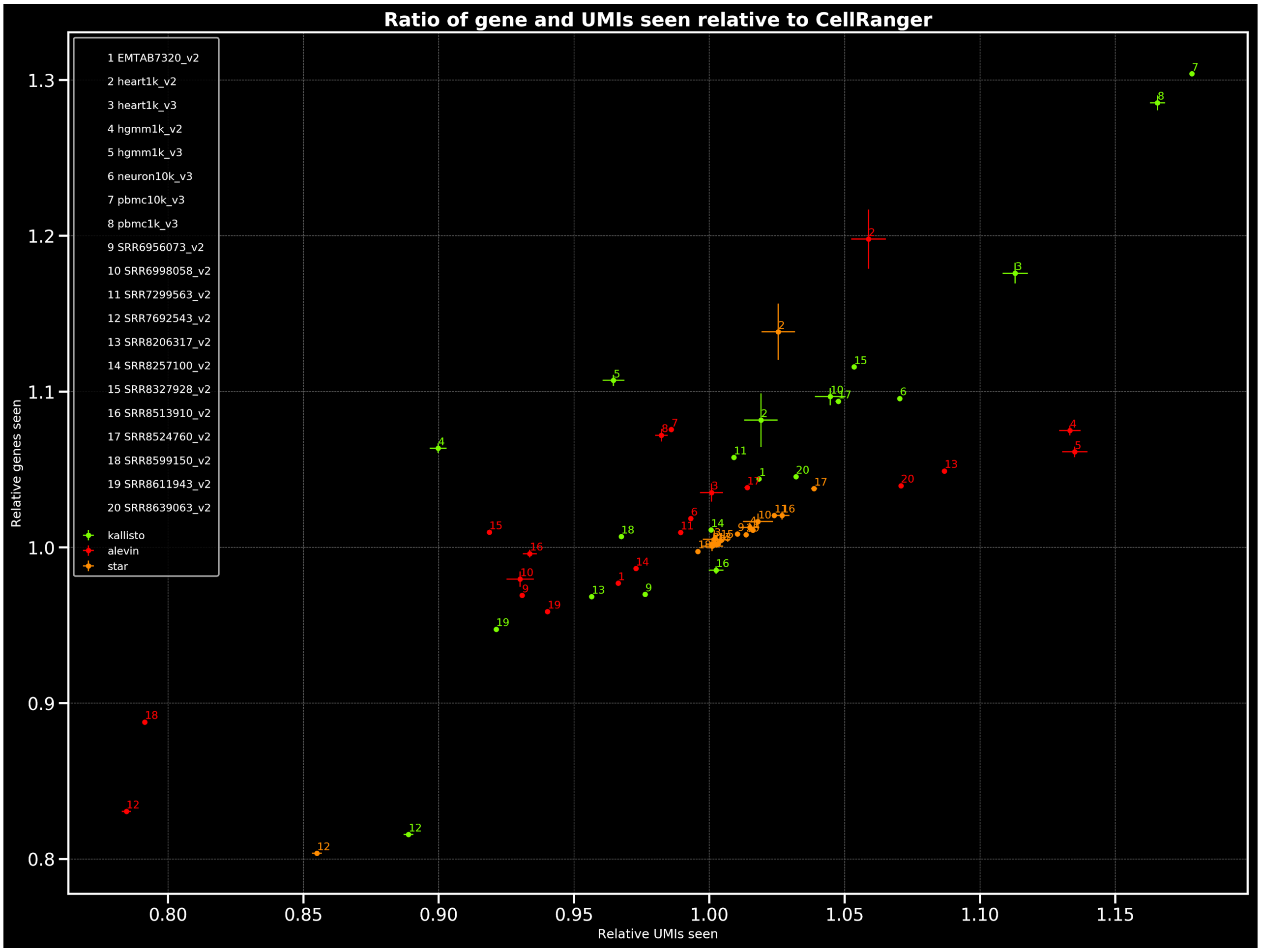


Figure 6.3: A comparison of the number of genes and UMIs seen in each dataset relative to CellRanger, taken to be the reference due to it's popularity. Bars denote one standard deviation.

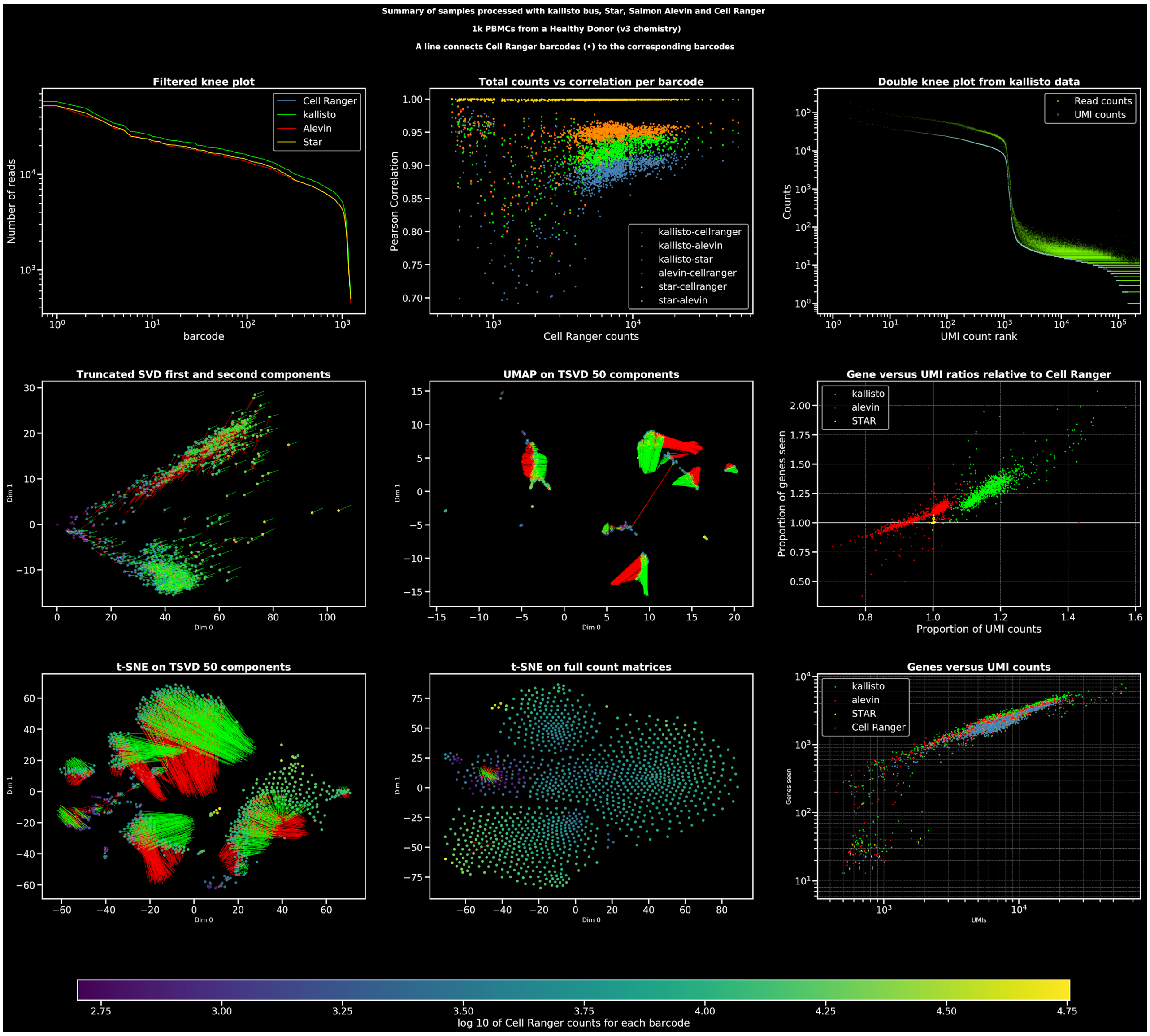


Figure 6.4: A panel of comparisons examining different metrics across the four methods. CellRanger is taken to be the reference, and in the t-SNE, SVG, and UMAP embeddings, a line connects the CellRanger point to it's counterpart from a different method, colored in the same scheme as the first panel.

7 *Quantifying the tradeoff between cells and depth*

This chapter is related to the preprint:

Quantifying the tradeoff between sequencing depth and cell number in single-cell RNA-seq

Valentine Svensson, Eduardo da Veiga Beltrame, and Lior Pachter.

bioRxiv 2019

doi: 10.1101/762773

Author contributions: *V.S. designed the evaluation metric and performed statistical analysis. E.V.B. performed data processing and subsampling. V.S., E.V.B., and L.P. interpreted results and wrote the manuscript.*

Code for reproducing the preprint:

`https://github.com/pachterlab/SBP_2019/`.

Code repository with more revent developments:

`https://github.com/Munfred/seqdepth`

Overview

In this work, described in the preprint, we introduced the concept of autoencoder based power analysis. We developed an approach to quantifying the impact of sequencing depth and cell number on the estimation of a multivariate generative model for gene expression that is based on the analysis of the scVI reconstruction error of the held-out validation data, which we refer to as the validation error. We call this workflow seqdepth.

We took advantage of the fact that scVI can easily be used to evaluate performance on held-out unseen validation data using the comparable and quantitative measure of log likelihood: the probability of seeing the data given the trained model (referred to as reconstruction error). An outline of the seqdepth workflow is shown the figure below.

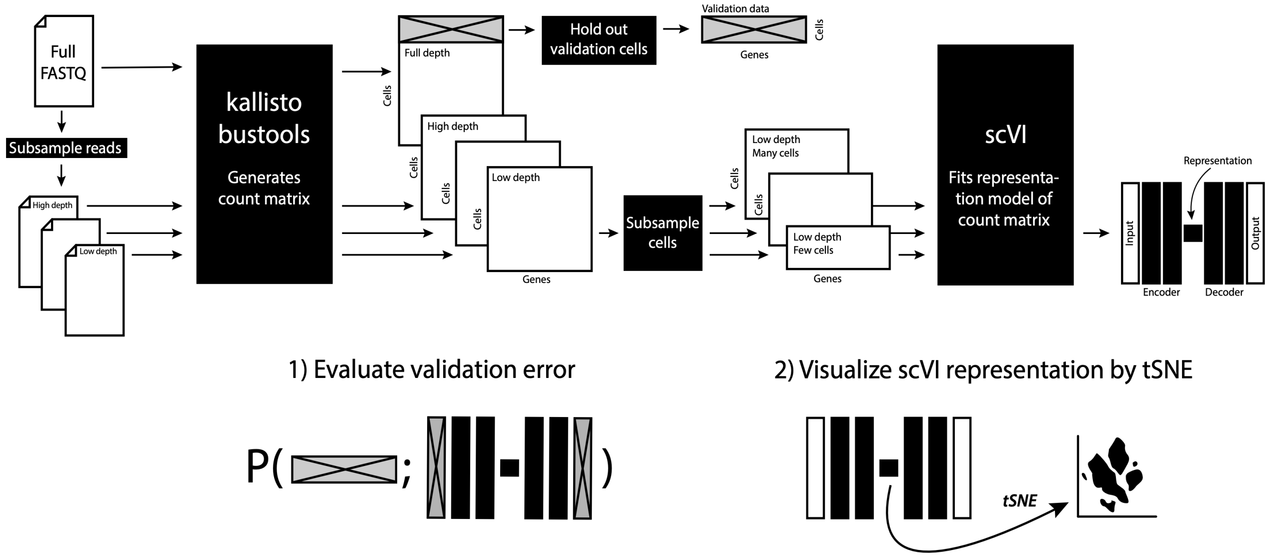


Figure 7.1: Outline of the workflow for subsampling reads and cells, fitting models with scVI, evaluating validation error, and visualization.

In initial results described in the preprint for three 10xv3 datasets, we found that at shallow depths, the marginal benefit of deeper sequencing per cell significantly outweighs the benefit of increased cell numbers. Above about 15,000 reads per cell the benefit of increased sequencing depth was found to be equivalent to assaying more cells.

Future directions

The seqdepth workflow enables subsampling a scRNA-seq dataset and obtaining a quantitative metric for how much can be learned from it (the validation error). Subsampling can be performed across either reads, which yields varying numbers of UMIs, or cells.

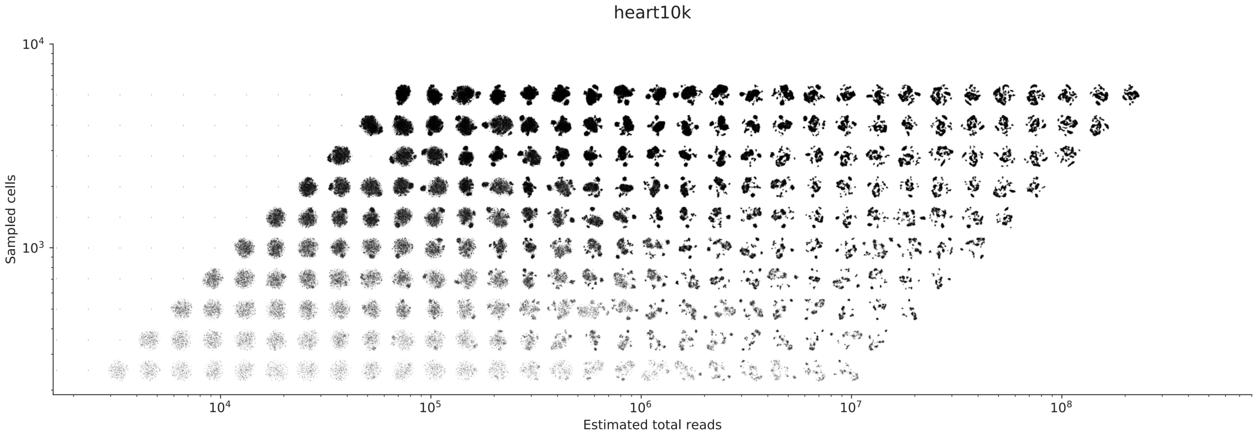


Figure 7.2: A mosaic of t-SNEs showing the results of subsampling a dataset with about 1000 heart cells across cells or depth, training an scVI model, then embedding the latent space in a 2D t-SNNE representation.

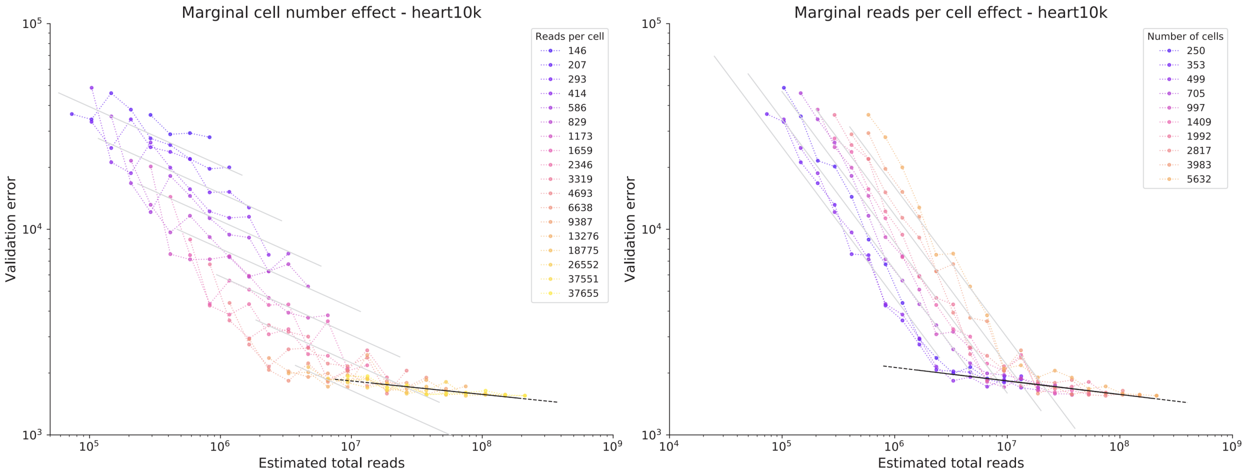


Figure 7.3: scVI validation errors plotted with points subsampled across reads or cells, showing a linear fit before and after 15,000 reads per cell.

Our preliminary results suggest diminishing results with increasing cells or depth, and that in particular the improvement from increasing depth tapers off more clearly than increasisng cells. This is frequently referred to a sequencing depth saturation.

We would like to have one or a few summary statistics that we can derive from each dataset using the seqdepth workflow that would allow us to compare different datasets in terms of how much can be learned as a function of the sequencing budget, and whether different datasets have different saturation points. This would enable comparing these statistics in terms of technology and biological systems.

To do that, we decided to use a simple 2D piecewise linear model with a breakpoint that only depends on the number of UMIs[1]. The error slope for number of cells before and after is allowed to vary independently. This model enables learning the following summary statistics: (i) A breakpoint, in terms of UMIs, at which each dataset "saturates" (ii) How much the error improves by increasing reads or UMIs before and after the breakpoint.

2D piecewise model description

The 2D piecewise model described below has been implemented in Stan and is available at `https://github.com/Munfred/seqdepth/blob/derp/transformed_piecewise_stan_model.stan`. The implemented Stan model was tested on input data where the error was sampled 30 times on a grid of $nUMI$ by $ncell$, values, which are spaced $\sqrt{2}$ apart. Input variables $nUMI$, $ncell$ and $error$ were $\log 2$ transformed and standardized, which made Stan much more stable.

The model is a 2D piecewise linear model with a breakpoint that only depends on $nUMI$. The $error$ slope for $ncell$ before and after is allowed to vary independently. This is illustrated in the picture below. The model is decribed as follows.

$I \sim \mathcal{N}(0,1)$	Intercept, average validation error at breakpoint.
$C_b \sim \mathcal{N}(0,1)$	Cell slope before breakpoint.
$C_a \sim \mathcal{N}(0,1)$	Cell slope breakpoint.
$U_b \sim \mathcal{N}(0,1)$	UMI slope before breakpoint.
$U_a \sim \mathcal{N}(0,1)$	UMI slope after breakpoint.
$B \sim \mathcal{N}(0,1)$	Breakpoint at which saturation begins.
$V_b \sim \mathcal{N}(0,1)$	Residual error before the breakpoint.
$V_a \sim \mathcal{N}(0,1)$	Residual error after the breakpoint.

Conditional mean for calculating the breakpoint: Let U be the number of UMIs per cells, and C the number of cells:

$$M = \begin{cases} I + U_b. \cdot (U - B) + C_b. \cdot C, & \textbf{if } U < B \\ I + U_a. \cdot (U - B) + C_a. \cdot C, & \textbf{if } U > B \end{cases}$$

[1] We looked into making a model with a breakpoint that depends on both cells and UMIs, but it was much harder to fit on Stan, and does not yield much improvement or insight.

Then the model for the error E is reduced to a breakpoint linear regression:

$$E = \begin{cases} E \sim \mathcal{N}(M, V_b), & \textbf{if } U < B \\ E \sim \mathcal{N}(M, v_a), & \textbf{if } U > B \end{cases}$$

A conceptual illustration of a 2D breakpoint model whose breakpoint only depends on the number of UMIs is to imagine it as fitting two planes on the data that can have varying slopes along the x and y axis (cells and UMIs) that are discontinuous at the boundary, that is, U does not need to have the same error E slope for the number of cells C before and after the breakpoint.

8 WormBase single cell tools

This chapter is related to the preprint:

Single cell tools for WormBase
Eduardo da Veiga Beltrame, Valerio Arnaboldi, Paul W. Sternberg
bioRxiv 2021
doi: 10.1101/2021.07.04.451030

Author contributions: *E.B. conceived and designed the tools, implemented scdefg and provided help with the development of wormcells-viz. V.A. implemented wormcells-viz and provided help with the development of scdefg. All authors wrote the manuscript.*

Code repository for scdefg: github.com/WormBase/scdefg

Code repository for wormcells-viz: github.com/WormBase/wormcells-viz

WormBase convention for anndata wrangling:
github.com/WormBase/anndata-wrangling

WormBase single cell tools webpage: single-cell.wormbase.org

Introduction

The number of single cell RNA sequencing (scRNA-seq) publications has exploded in recent years, with over 1200 studies currently available and over 350 new studies in 2020 alone. This wealth of data presents new challenges and opportunities on how to integrate, query, and display results in ways that are useful and easy to use for scientists.

Over 85% of scRNA-seq studies use human or mouse samples, and the volume of data generated by these studies is so high that their integration and unified management represents a formidable engineering challenge. However for other model organisms such as *C. elegans*, for which there are only on the order of a dozen scRNA-seq studies in the literature , data integration and maintenance of tools covering most of the published data is manageable by a single individual or research group with simpler tools.

WormBase is a member of the Alliance of Genome Resources (alliancegenome.org), a consortium of model organism databases that encompasses zebrafish, *Drosophila melanogaster*, mouse, rat and yeast. In this work we have also curated the available *C. elegans* data and made it available for the community. For other model organism databases it is also feasible to manually curate all the relevant public datasets. Once the data is curated, integrating such massive aggregated datasets with scvi-tools becomes straightforward. By leveraging the tools presented here it is possible to offer users an interface to query and compare data from several studies in a way that is quick and useful but without the need to write code.

Organism	**Studies**
Mouse	514
Human	423
Human & Mouse	92
Zebrafish	25
Drosophila	12
Rat	10
C. elegans	7
Yeast	3
A. thaliana	3
Chicken	3
P. falciparium	3

Table 8.1: Number of scRNAseq studies for the most popular organisms. Over 50 other organisms have at least one study.

Once the gene count matrix is filtered and distinct cell types are labeled, the task of enduring interest is to perform differential expression (DE) between arbitrary groupings of cells. Because there are frequently dozens to hundreds of labeled cell types, and possibly multiple experimental conditions, it is impractical to pre-compute the DE results for all possible groupings of interests. As more data becomes available, it becomes possible to integrate and compare the new data with old data, and generate new groupings of interest. And additional data or better software enables improved re-annotation of cell types.

For example, in 2017 Cao et al.[1] published two scRNA-seq experiments with *C. elegans* L2 larvae. Then in 2019 Packer et al.[2] published a *C. elegans* developmental trajectory, and re-annotated the Cao dataset. This means that a scRNA-seq dataset may continue being of interest long after publication, and it's value may even increase with time, with better annotations and additional data to compare it to. Therefore, it is necessary to develop interfaces that allow scientists to easily query these datasets to perform DE on groups of interest.

One of the advantages of using scVI for performing data integration is that the unsupervised model benefits from seeing additional training data. It does not matter that most of the data might be compared in the analysis, such as when only a few cell types are of interest. Thus it is possible to train the model once, on all datasets that could possibly be of interest, and then repeatedly query the trained model for DE between groupings of interest.

To enable rapid iteration and data exploration, it is ideal to have tools that enable visual data exploration, without the need for writing code to perform each operation. In this spirit we developed two apps: scdefg and wormcells-viz, and deployed them at WormBase using *C. elegans* scRNA-seq data at single-cell.wormbase.org.

- scdefg is a simple application that provides a graphical interface for performing differential expression (DE) on scRNA-seq data and enables the user to make arbitrary groupings of data based on available annotations.

- wormcells-viz is a visualization tool that takes in processed data and enable query and visualization of heatmaps, dotplots, ridgeline gene abundance histograms, and swarmplots

We expect these tools to be useful for both individual research groups investigating new datasets as well as for model organisms databases such as those part of the Alliance of Genome Resources. These tools are centered around the anndata file format, a standard file format for scRNA-seq data, and scvi-tools.

[1] Junyue Cao, Jonathan S. Packer, Vijay Ramani, Darren A. Cusanovich, Chau Huynh, Riza Daza, Xiaojie Qiu, Choli Lee, Scott N. Furlan, Frank J. Steemers, Andrew Adey, Robert H. Waterston, Cole Trapnell, and Jay Shendure. Comprehensive single-cell transcriptional profiling of a multicellular organism. *Science*, 357(6352):661–667, August 2017. doi: 10.1126/science.aam8940

[2] Jonathan S. Packer, Qin Zhu, Chau Huynh, Priya Sivaramakrishnan, Elicia Preston, Hannah Dueck, Derek Stefanik, Kai Tan, Cole Trapnell, Junhyong Kim, Robert H. Waterston, and John I. Murray. A lineage-resolved molecular atlas of C. elegans embryogenesis at single-cell resolution. *Science*, 365(6459), September 2019. doi: 10.1126/science.aax1971

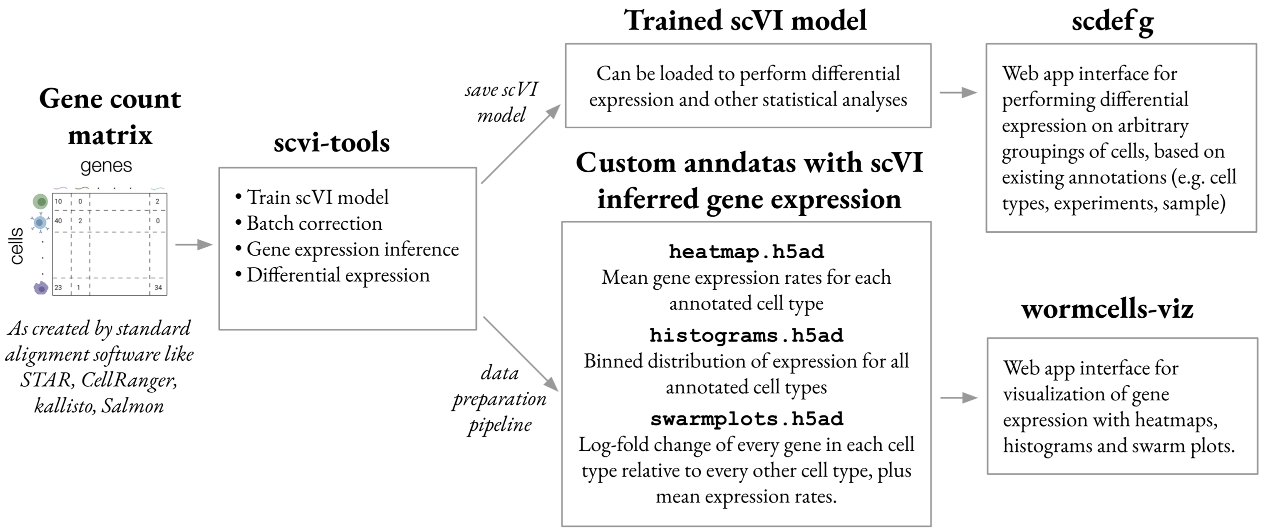

Gene count matrix
genes
cells
10 0 2
40 2 0
23 1 34
As created by standard alignment software like STAR, CellRanger, kallisto, Salmon
scvi-tools
• Train scVI model
• Batch correction
• Gene expression inference
• Differential expression
save scVI model
data preparation pipeline
Trained scVI model
Can be loaded to perform differential expression and other statistical analyses
Custom anndatas with scVI inferred gene expression
heatmap.h5ad
Mean gene expression rates for each annotated cell type
histograms.h5ad
Binned distribution of expression for all annotated cell types
swarmplots.h5ad
Log-fold change of every gene in each cell type relative to every other cell type, plus mean expression rates.
scdefg
Web app interface for performing differential expression on arbitrary groupings of cells, based on existing annotations (e.g. cell types, experiments, sample)
wormcells-viz
Web app interface for visualization of gene expression with heatmaps, histograms and swarm plots.

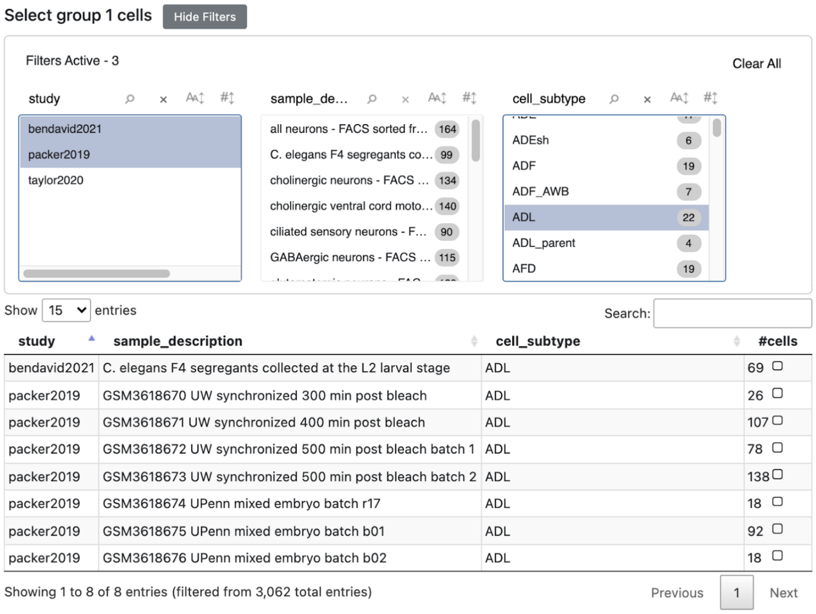
Figure 8.1: Overview of the process to go from gene count matrix to deployment of the apps. Note that scdefg only requires as input the trained scVI model as saved by scvi-tools, while wormcells-viz requires using our pipeline to create the custom input anndatas,which are saved as.h5ad files.

scdefg

The scdefg app provides a single web page with an interface for performing DE on two groups of cells that can be selected according to the existing annotations in the data. For example, the user can select a group according to a combination of cell type, sample, tissue and experimental group. DE is computed using the scVI model from scvi-tools, which enables quick computation even when using only CPUs. The results are displayed in the form of an interactive volcano plot (log fold change vs p-value) and MA plot (log fold change vs mean expression) that display gene descriptions upon mouseover, and sortable tabular results that can be downloaded in csv and Excel format. The app is written in Python using Flask and Plotly, and can be launched from the command line by specifying the path to a trained scVI model, plus the data labels by which cell groups may be stratified (e.g. cell type, experiment, sample). We have deployed the app on a cloud instance with only 8GB RAM and 2 vCPUs and observed this configuration is sufficient for handling a few concurrent users with results being returned in about 15 seconds.

wormcells-viz

The wormcells-viz app provides interactive and responsive visualizations of heatmaps, gene expression histograms, and swarmplots. It is written in Javascript and Python and uses React (reactjs.org/) and D3 (d3js.org). Deploying the app requires having pre-computed gene expression values stored in three custom anndata files as described in



Select group 1 cells

Filters Active - 3

study	sample_de…		cell_subtype	
bendavid2021	all neurons - FACS sorted fr…	164	ADEsh	6
packer2019	C. elegans F4 segregants co…	99	ADF	19
taylor2020	cholinergic neurons - FACS …	134	ADF_AWB	7
	cholinergic ventral cord moto…	140	ADL	22
	ciliated sensory neurons - F…	90	ADL_parent	4
	GABAergic neurons - FACS …	115	AFD	19

Show 15 entries

Search:

study	sample_description	cell_subtype	#cells
bendavid2021	C. elegans F4 segregants collected at the L2 larval stage	ADL	69 □
packer2019	GSM3618670 UW synchronized 300 min post bleach	ADL	26 □
packer2019	GSM3618671 UW synchronized 400 min post bleach	ADL	107 □
packer2019	GSM3618672 UW synchronized 500 min post bleach batch 1	ADL	78 □
packer2019	GSM3618673 UW synchronized 500 min post bleach batch 2	ADL	138 □
packer2019	GSM3618674 UPenn mixed embryo batch r17	ADL	18 □
packer2019	GSM3618675 UPenn mixed embryo batch b01	ADL	92 □
packer2019	GSM3618676 UPenn mixed embryo batch b02	ADL	18 □

Showing 1 to 8 of 8 entries (filtered from 3,062 total entries)

Figure 8.2: The interface for selecting cells on scdefg

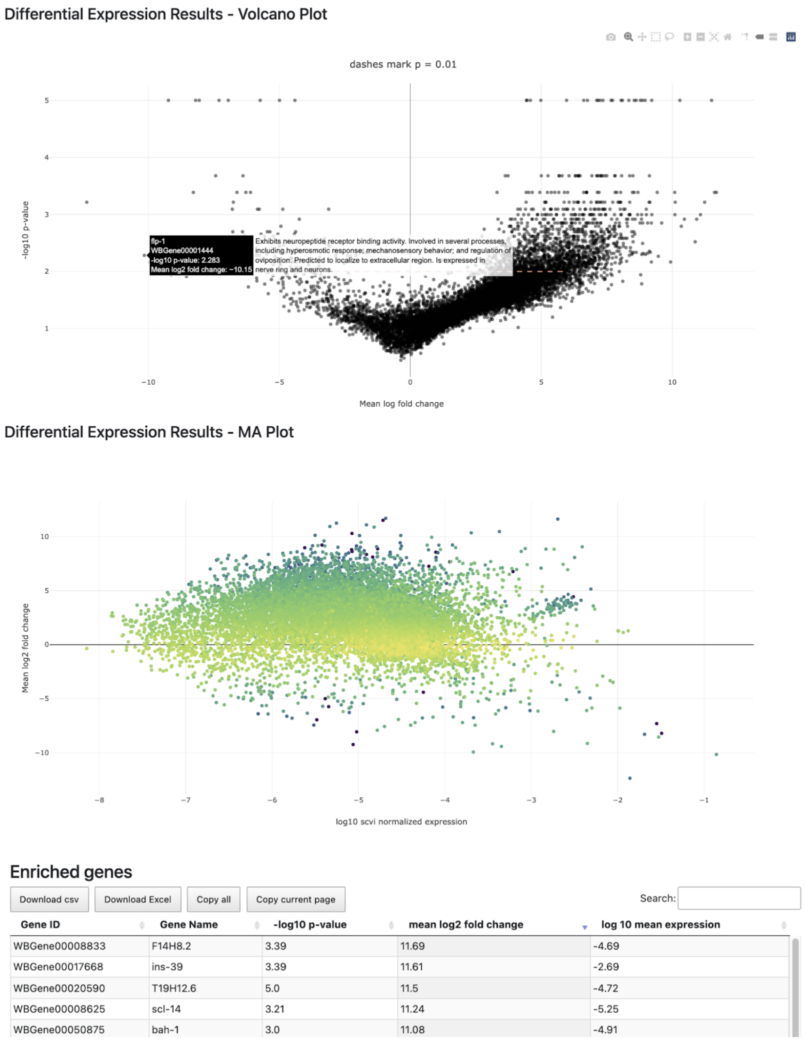
the supplements. Using one anndata file for each visualization type keeps the codebase modular and simplifies adding more visualizations in the future. In the wormcells-viz documentation we provide a pipeline and tutorial to compute these expression values using scVI with any scRNA-seq dataset. The pipeline could be adapted for using other scRNA-seq analysis frameworks, but our recommendation is to use scvi-tools. The following visualizations are currently implemented.

Heatmap: Visualization of mean gene expression in each group annotated in the data. The expression rates can be shown as either a traditional heatmap, or as a monochrome dotplot.

Gene expression histogram: Histograms of the gene expression rates for a given gene across all cell types in the data. The histogram bin counts are computed from the scVI inferred expression rates for each cell.

Swarmplot: This is a new visualization strategy to facilitate candidate marker gene identification. For a given cell type, swarmplots visualize the relative expression of a set of genes across all cells annotated in a dataset. The Y axis displays the set of selected genes, and the X axis displays the log fold change in gene expression between the cell type of interest and all other cell types. This is computed by doing pairwise differential expression of each annotated cell type vs. the cell type of interest.

Below the conventions of the format expected by wormcells-viz for an anndata object are described.



Differential Expression Results - Volcano Plot

flp-1
WBGene00001444
-log10 p-value: 2.283
Mean log2 fold change: −10.15

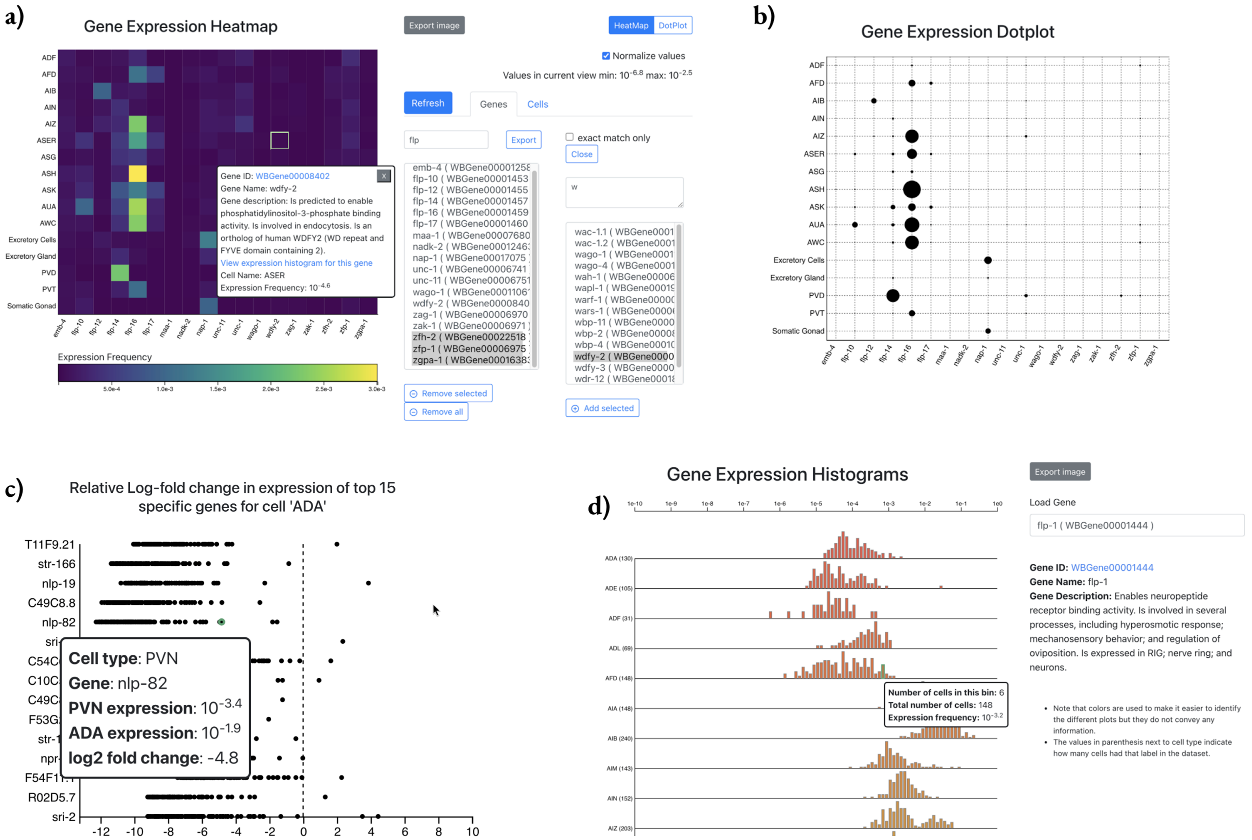
Exhibits neuropeptide receptor binding activity. Involved in several processes, including hyperosmotic response; mechanosensory behavior; and regulation of oviposition. Predicted to localize to extracellular region. Is expressed in nerve ring and neurons.

Differential Expression Results - MA Plot

Enriched genes

Gene ID	Gene Name	-log10 p-value	mean log2 fold change	log 10 mean expression
WBGene00008833	F14H8.2	3.39	11.69	-4.69
WBGene00017668	ins-39	3.39	11.61	-2.69
WBGene00020590	T19H12.6	5.0	11.5	-4.72
WBGene00008625	scl-14	3.21	11.24	-5.25
WBGene00050875	bah-1	3.0	11.08	-4.91

Figure 8.3: The results view of scdefg, showing a volcano plot, MA plot, and tabular results of differential expression.



Gene ID: WBGene00008402
Gene Name: wdfy-2
Gene description: Is predicted to enable phosphatidylinositol-3-phosphate binding activity. Is involved in endocytosis. Is an ortholog of human WDFY2 (WD repeat and FYVE domain containing 2).
View expression histogram for this gene
Cell Name: ASER
Expression Frequency: $10^{-4.6}$

☑ Normalize values

Values in current view min: $10^{-6.8}$ max: $10^{-2.5}$

flp

emb-4 (WBGene0000125
flp-10 (WBGene00001453
flp-12 (WBGene00001455
flp-14 (WBGene00001457
flp-16 (WBGene00001459
flp-17 (WBGene00001460
maa-1 (WBGene00007680
nadk-2 (WBGene0001246
nap-1 (WBGene00017075
unc-1 (WBGene00006741
unc-11 (WBGene00006751
wago-1 (WBGene0001106
wdfy-2 (WBGene0000840
zag-1 (WBGene00006970
zak-1 (WBGene00006971)
zfh-2 (WBGene00022518
zfp-1 (WBGene00006975
zgpa-1 (WBGene0001638

☐ exact match only

w

wac-1.1 (WBGene0001
wac-1.2 (WBGene0001
wago-1 (WBGene0001
wago-4 (WBGene0001
wah-1 (WBGene00006
wapl-1 (WBGene0001
warf-1 (WBGene0000
wars-1 (WBGene0000
wbp-11 (WBGene0000
wbp-2 (WBGene0000
wbp-4 (WBGene0001
wdfy-2 (WBGene0000
wdfy-3 (WBGene0000
wdr-12 (WBGene0001

Cell type: PVN
Gene: nlp-82
PVN expression: $10^{-3.4}$
ADA expression: $10^{-1.9}$
log2 fold change: -4.8

Number of cells in this bin: 6
Total number of cells: 148
Expression frequency: $10^{-3.2}$

flp-1 (WBGene00001444)

Gene ID: WBGene00001444
Gene Name: flp-1
Gene Description: Enables neuropeptide receptor binding activity. Is involved in several processes, including hyperosmotic response; mechanosensory behavior; and regulation of oviposition. Is expressed in RIG; nerve ring; and neurons.

- Note that colors are used to make it easier to identify the different plots but they do not convey any information.
- The values in parenthesis next to cell type indicate how many cells had that label in the dataset.

Figure 8.4: The visualizations in wormcells-viz. a) Heatmap view. b) Dotplot view, showing same information as heatmap. c) Swarmplot. d) Gene expression histogram showing scVI normalized expression rates.

Gene expression histograms anndata formatting

These are used for plotting histograms of the scVI inferred expression rates for a given gene across all cell types in the data. Each anndata layer stores the expression values for all cell types and a given gene. The histogram bin counts are computed from the scVI normalized expression values, which are generated by a trained scVI model using the method `scvi.model.SCVI.get_normalized_expression`. The anndata `obs` contains the cell types and `var` contains the histogram bins, the genes are stored in layers with the keys being the gene ID. We store the genes in the layers because each view in the *wormcells-viz* app show the histograms for a single gene, so this makes accessing the data simpler. The histogram bin counts are computed from the $\log_{10}$ of the scVI expression rate. Each bin contains the number of cells in the dataset that were inferred to have an expression rate in the bin interval. There should be 100 bins with values between $(-10, 0)$, representing expression rates from 10^{-10} to 10^{0}. The data array is of shape $n_{celltypes} \times n_{bins} \times n_{genes}$. The `adata` properties should be:

- `adata.obs` := Dataframe with cell types in index.
- `adata.var` := Dataframe with the bin intervals in index.
- `adata.X` := Not used.
- `adata.layers[gene_id]` := Each layer key is a gene ID, and contains a matrix with the binned expression rate counts for all cell types.
- `adata.uns['about']` := String with dataset information.

Heatmap anndata formatting

These are used for plotting a heatmap of the average expression rates in each cell type, for a given selection of cell types and genes. The input data is a matrix of cell types and gene expression rates that contains the $\log_{10}$ scVI expression rate values. Cell types are in `adata.obs` and genes in `adata.var`. The data array is of shape $n_{celltypes} \times n_{genes}$. The `adata` properties should be:

- `adata.obs` := Dataframe with cell types in index.
- `adata.var` := Dataframe with gene IDs in index.
- `adata.X` := Matrix with $\log_{10}$ values of scVI normalized expression for each cell type and each gene.
- `adata.uns['about']` := String with dataset information.

Swarmplots anndata formatting

These are used for plotting relative expression of a set of genes across all cells annotated in the dataset. The Y axis displays the set of selected genes, and X axis displays the log fold change of expression of that gene on all cell types relative to the cell of interest. The fold change is computed by doing pairwise differential expression of each annotated cell type vs the cell type of interest. For a given cell type, genes can be sorted by p-value, log fold change or mean expression rate. This part of the data is an array of shape $n_{celltypes} \times n_{genes} \times n_{celltypes}$. The cell types are repeated along two dimensions, because this data contains the results of pairwise DE comparisons among each cell type in the data.

Plus $n_{celltypes}$ matrices shaped like $n_{celltypes} \times n_{genes}$, because each unstructured layer `adata.uns[celltype]` contains a dataframe with global differential expression results for that cell type.

Finally, the unstructured layer `anndata.uns['heatmap']` contains a matrix with $\log_{10}$ scVI expression rates heatmap data (same data as used for plotting the heatmap), with genes in the index and cell types in the columns. This data is used to display the expression of each cell type on mouseover. The `adata` object properties should be:

- `adata.obs` := Dataframe with cell types in index.
- `adata.var` := Dataframe with gene IDs in index.
- `adata.X` := Not used.
- `adata.layers[cell_type]` := Mean log fold change for a given cell type for all genes.
- `adata.uns[cell_type]` :=The differential expression tables of the corresponding cell type vs all other cells. This can be used for ordering the genes by p-value, log fold change, and expression rate.
- `adata.uns['heatmap']` := Dataframe with genes in index and cell types in columns containing the $\log_{10}$ of the scVI expression frequency for each cell type
- `adata.uns['about']` := String with dataset information.

Motivations for using scvi-tools

There are currently hundreds of software tools and pipelines developed for scRNA-seq data. The scvi-tools framework offers several models for single cell omics data, and for scRNA-seq in particular offers the scVI model, which is bayesian hierarchical generative model that leverages variational autoencoders to enable robust statistical analysis. It is built with PyTorch and has been extensively validated. The following considerations led to our choice of using scvi-tools for driving scRNA-seq analysis at WormBase.

Scalability: scvi-tools models readily scale to datasets with millions of cells. Using a GPU even large models can be trained in a few hours. Training needs to be done only once, and new datasets can be integrated to the data without having to re-train the entire model.

Consistent development and contributors: The scvi-tools codebase was first introduced in 2017, and published in 2018, with consistent updates and improvements since. It now boasts a mature and professional API and codebase that follows industry best practices, and has over 37 unique contributors and 56 releases.

Extensible framework for analysis: Because the generative model of the data can be modified to reflect our assumptions about underlying processes, the framework can be extended to model other aspects of scRNA-seq data. Currently, extensions include cell type classification and label transfer across batches, modelling single cell protein measurements, single cell chromatin accessibility assays, gene imputation in spatial data, and using a linear decoder to allow for interpretation of the learned latent space.

WormBase data deployments

At the moment, the majority of scRNA-seq data is generated using 10X Genomics Chromium technology. This is also true for *C. elegans* scRNA-seq data. For the time being WormBase will focus development efforts on scRNA-seq tools on 10X Genomics data. Two considerations drive this:

First, data integration of different batches with scvi-tools is more robust when there is more data, and when the technology and biological system of each batch is the same or similar. Attempting to integrate a small number of cells from unique technologies and unique biological systems can make it impossible to discern biological differences from technical artifacts.

Second, the 10X Genomics data has a widely used, validated and commercially supported data pre-processing workflow, from FASTQ files to gene count matrices. This can enable WormBase to uniformly reprocess the FASTQ files in a single pipeline in the future.

WormBase standard wrangled anndatas

There are multiple file formats that are frequently used to store the gene count matrix with scRNA-seq data. When the processed data is made available in GEO or other repositories, authors typically will deposit the file format used by the application they were working with. This results in new users of this data almost always having to manually re-wrangle this data, to convert it to their desired format and also to have standard named fields for the annotations. This is fine when done for one or a few datasets, but is a significant barrier for performing any kind of larger re-analysis, as the vast majority of time ends up being spent in laborious and inglorious data wrangling.

The Anndata file format (stored in .h5ad files) was published in 2018 as a generic class for handling annotated data matrices, with a focus on scRNA-seq data and Python support for machine learning, and with integration with the popular SCANPY analysis framework[3]. Anndata is an efficient storage format because it uses HDF5 compression, and has come to be the standard format for manipulating scRNA-seq data in Python, as well as providing extensive support in R and interoperability with the popular Seurat package .

Anndata's popularity and uses continues to grow, with many packages standardizing their data maniulation around it. Examples include scvi-tools (scvi-tools.org), the Chan Zuckerberg cellxgene platform (chanzuckerberg.github.io/cellxgene) and the COVID-19 cell atlas initiative which standardized data distribution around anndata (covid19cellatlas.org).

Anndata objects have a main matrix, which could be a sparse or dense matrix, is stored in the .X property. Observation annotation (each line corresponding to a cell or barcode) are stored in a pandas dataframe in the .obs property. Variable annotations (corresponding to a gene, or other feature such as proteins) are stored in the .var property. Additional mappings for each observation (of dimensions number of observations x arbitrary number), such as the coordinates for a low dimensional embedding of PCA, t-SNE or UMAP can be stored in an extra layer property named .obsm, and accessed via a key name. The anndata object also supports any number of additional variables of arbitrary format in the .uns field, accessed via a key name. This enables extreme flexibility for storing study metadata and using anndata for new use cases.

Owing to the advantages of anndata and its popularity, WormBase adopted a convention for manually wrangling published *C. elegans* scRNA-seq data into anndata files with standard field names, to streamline their reuse in code pipelines. The convention used by WormBase when wrangling data into anndata objects is described at: github.com/WormBase/anndata-wrangling

Briefly, standard field names are lower case, short, descriptive, and using valid Python variable names. The goal is to standardize the naming convention for frequently used fields so that code and

[3] F. Alexander Wolf, Philipp Angerer, and Fabian J. Theis. SCANPY: large-scale single-cell gene expression data analysis. *Genome Biology*, 19(1), February 2018. doi: 10.1186/s13059-017-1382-0

pipelines may be immediately reused without changing variable names or any alterations. The .h5ad file should only contain genes and cells with at least one count. These guidelines are intentionally simple so that they may be easily adhered to.

var name	**Description**	**Example value**	**Optionality**
index	WormBase gene ID	WBGene00010957	Required
gene_id	WormBase gene ID	WBGene00010957	Required
gene_name	WormBase gene name	nduo-6	Required
gene_description	WormBase short gene description.	Predicted to have NADH dehydrogenase activity.	Optional

Table 8.2: WormBase naming guidelines for the anndata var annotation names

obs name	**Description**	**Example value**	**Optionality**
index	The batch name joined with cell barcode witha + char	F4_1+TGTAACGGTTAGCTAC-1	Required
study	A unique shorthand for the study that published the data, ideally in the style all lower case.	bendavid2021	Required
sample_batch	The run that produced the corresponding barcode. Typically **sample_batch** and **sample** will be the same, but with multiplexing one batch can have multiple samples	F4_1	Required
sample	The name of the biological sample that is in this batch	L2 larvae batch 4	Required
sample_description	Description of the sample. This is mandatory because otherwise it would be easy to confuse two samples from their short name.	F4_1	Required
barcode	The cell barcode	AAACCCAAGATCGCTT-1	Required
cell_type	The cell type annotation provided by the authors. The value should be the string **unlabeled** if not available.	Neuronal	Required
cell_subtype	The lowest level of cell type annotation if provided by the authors. If only one level of cell type label was provided, it should be repeated in both **cell_type** and **cell_subtype**. This value should be the string **unlabeled**	ASJ	Required

Table 8.3: WormBase naming guidelines for the anndata obs annotation names

Author and year	**Ref**	**Accession**	**CaltechData DOI**	**Description**
Hashimshony 2012	[14]	SRP014672	Not wrangled	This was one of the pioneering works in scRNA-seq and introduced the CEL-Seq technique. They reported 96 *C. elegans* cells.
Tintori 2016	[33]	GSE77944	Not wrangled	They surveyed the *C. elegans* embryo through the 16-cell stage and reported 216 cells. They made a custom visualizer at tintori.bio.unc.edu.
Cao 2017	[4]	GSE98561	10.22002/D1.2000	The first high throughput scRNA-seq study on *C. elegans*, they introduced the sci-RNA-seq technique and surveyed over 50,000 cells from whole organism L2 larvae.
Packer 2019	[25]	GSE126954	10.22002/D1.1945	They performed a comprehensive survey of the *C. elegans* embryo developmental trajectory with over 86,000 cells with 10X Genomics v2 chemistry. They made a custom visualizer at cello.shinyapps.io/celegans
Taylor 2021	[32]	GSE136049	10.22002/D1.1977	As part of the CeNGEN project (cengen.org) they FACS sorted L4 larvae neurons and surveyed over 101,000 cells using 10X Genomics v2 and v3 chemistry. They report 65,000 neurons across all neuron types. They made a custom visualizer at cengen.shinyapps.io/CengenApp.
Ben-David 2021	[2]	PRJNA658829	10.22002/D1.1972	They surveyed over 55,000 cells of L2 larvae using 10X Genomics v2 chemistry.

Table 8.4: Summary of *C. elegans* single cell RNA sequencing datasets. High throughput data has been wrangled following the WormBase standard anndata convention and deposited at CaltechData.

9 *Finding marker genes from scRNA-seq data*

A manuscript related to this project is still in preparation, describing experimentally validated results for new markers for several other *C. elegans* neurons. The experimental work was done by Mark Guangde Zhang, Olivia Xuan Wan, Stephanie Nava, Wilber Palma and Shala Gharib.

This workflow is available as Jupyter Notebooks that can be run using Google Colab at github.com/Munfred/worm-markers

Introduction

In this project a computational workflow leveraging scVI to identify potential *C. elegans* marker genes from scRNA-seq data was developed and experimentally validated. In principle, there are only two features that make a "good" marker gene for a given cell type: having high specificity to the cell type of interest and having high expression. We reasoned that as long as the cell types of interest were correctly identified in the data, it should be straightforward to identify potential marker genes by using the scVI differential expression (DE) feature to identify cell type specific genes, and the scVI expression frequency to identify high expression genes.

Because scVI is a Bayesian framework we obtain a posterior predictive distribution when performing hypothesis testing for differential expression tests. We refer to the peak of the posterior predictive distribution as the posterior predictive p-value or simply p-value. It fulfills the same role of p-values in frequentist statistics, helping us assess how robust a result is.

By repeatedly sampling each cell, scVI allows the inference of transcript "frequencies", corresponding to the fractions of transcripts sampled in a given cell that belong to each gene. This is akin to the normalized gene expression that is used in other scRNAseq processing frameworks. However, it is an inference instead of an empirically estimated value obtained by for example dividing the number of transcripts sequenced in each cell by the total. We refer to it as the expression frequency of a gene or simply expression of a gene.

Workflow Description

i) Train the scVI model to integrate the datasets of interest.

scVI creates a latent representation of each cell that is independent of cell size or sequencing depth (the number of transcripts seen in that cell) and of batch (which experimental run a sample came from). Thus, cells that are closer in the latent representation should be more biologically similar (that is, express the same genes in similar quantities). The latent representation uses the batch as a conditional variable, and this enables batch correction when performing differential expression.

ii) Perform DE on the cells of interest to select top genes by DE probability Once the scVI model has been trained, the second step is to select the cells of interest and perform DE using the scVI change mode. To visualize specificity and expression rate on the target cells we can make a scatter plot of the posterior predictive p-value vs. expression frequency for each gene.

iii) Visualize the relative expression across all tissues to select marker candidates using swarmplots. Once the p-values for each gene are computed, the genes with the lowest p-values can be selected for visualizing their expression in every cell type using a swarmplot as introduced in the previous chapter. The swarmplot is a scatter plot of the expression frequency on each cell type normalized against the expression in the target cell, repeated for each gene. Then the expression on the cell type of interest is one, and for a highly specific gene it should be close to zero in all other cell types. Inspection of this plot quickly reveals whether there are clear marker gene candidates.

Results

Using this strategy we proceeded to identify marker candidates for the ASG neuron leveraging the CeNGEN dataset[1]. A swarmplot for the top 25 differentially expressed genes enriched in ASG is shown in Figure 9.1. From inspection of an interactive swarmplot, similar to the ones implemented in WormBase as described in chapter 8, the gene Y41C4A.6 was selected for experimental validation because it showed very high expression.

The interactive swarmplots are available as part of the and implemented in the WormBase single cell tools are shown in Figure 9.2.

[1] Seth R. Taylor, Gabriel Santpere, Alexis Weinreb, Alec Barrett, Molly B. Reilly, Chuan Xu, Erdem Varol, Panos Oikonomou, Lori Glenwinkel, Rebecca McWhirter, Abigail Poff, Manasa Basavaraju, Ibnul Rafi, Eviatar Yemini, Steven J. Cook, Alexander Abrams, Berta Vidal, Cyril Cros, Saeed Tavazoie, Nenad Sestan, Marc Hammarlund, Oliver Hobert, and David M. Miller. Molecular topography of an entire nervous system. *Cell*, 184(16), August 2021. doi: 10.1016/j.cell.2021.06.023

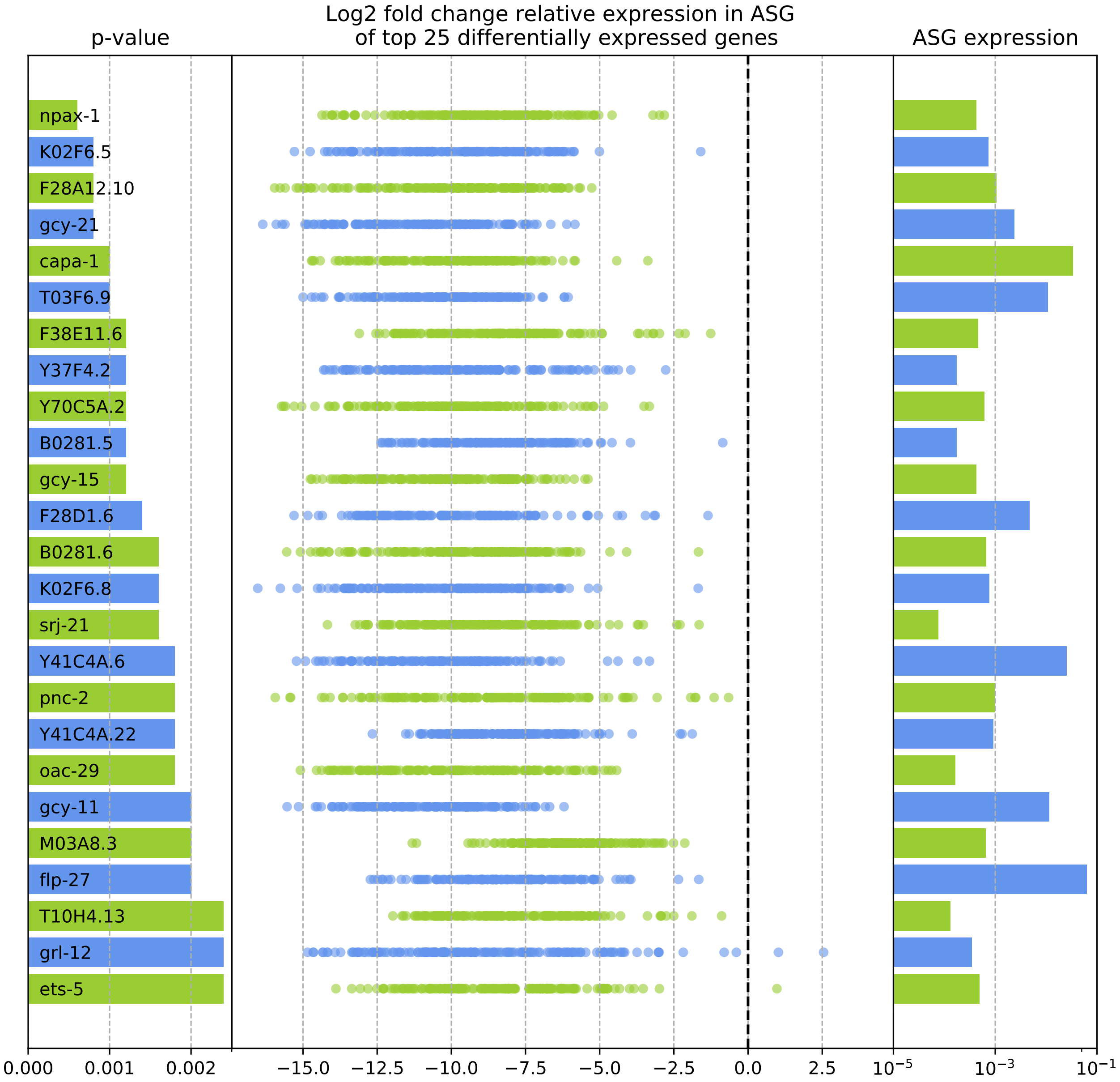
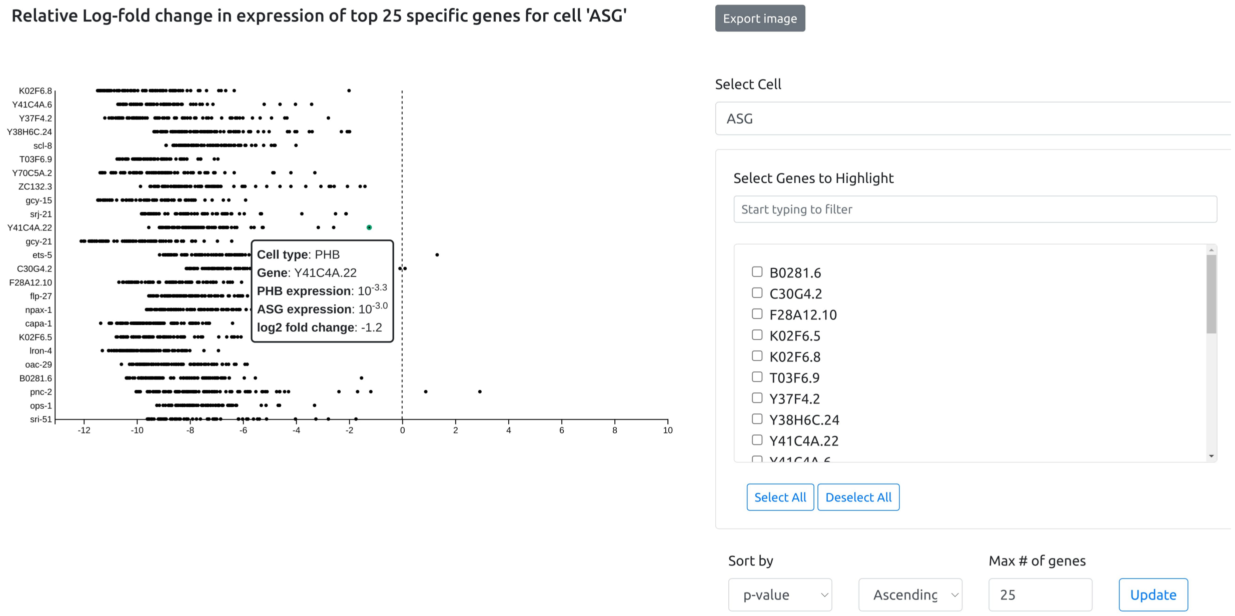


Figure 9.1: Swarmplot of top 25 differentially expressed genes in ASG neuron. From a similar plot the gene Y41C4A.6 was selected for experimental validation



Relative Log-fold change in expression of top 25 specific genes for cell 'ASG'

Cell type: PHB
Gene: Y41C4A.22
PHB expression: $10^{-3.3}$
ASG expression: $10^{-3.0}$
log2 fold change: -1.2

Select Cell: ASG

Select Genes to Highlight

Sort by: p-value

Ascending

Max # of genes: 25

Figure 9.2: The ASG interactive swamplot visualization in the WormBase single cell tools.

10 Epilogue

This thesis presented 8 projects that I worked on during my graduate studies at Caltech. Their primary unifying theme is that they are all about single cell RNA sequencing. However, a strong, unspoken, underlying theme is that they all have a bent towards sharing. They are tools, resources and methods. Things that are not an end in themselves, but where the hope is that they will be useful to others.

Science is, at the end of the day, a social enterprise. And I find that I do my best work not alone, but with others, and with my friends: someone to bounce ideas, try random things, motivate you to keep going, and teach you things. Working with friends that know things that I don't is not only more fun, it is far more productive.

Caltech is a very very very special place to do creative collaborative work, and I feel very fortunate to have joined this wonderful community. Single cell RNA sequencing too, as a field, is very much about collaboration and sharing. The field has a strong emphasis on sharing code, making data available, and making methods reproducible. This kind of culture is not a given across scientific fields, and it should be treasured and fostered.

As I go on to life after grad school, I can only say that I continue to deeply feel this impetus to go on sharing, working with my friends, being friends with the people I work with, and having fun while doing it.

You can follow the next chapters of this adventure on my website, munfred.com.

Cheers,

Eduardo

Bibliography

[1] Simon Anders and Wolfgang Huber. Differential expression analysis for sequence count data. *Nature Precedings*, March 2010. doi:10.1038/npre.2010.4282.1.

[2] Eyal Ben-David, James Boocock, Longhua Guo, Stefan Zdraljevic, Joshua S. Bloom, and Leonid Kruglyak. Whole-organism eQTL mapping at cellular resolution with single-cell sequencing, March 2021. doi: 10.7554/eLife.65857.

[3] Nicolas L. Bray, Harold Pimentel, Páll Melsted, and Lior Pachter. Near-optimal probabilistic RNA-seq quantification. *Nature Biotechnology*, 34(5), May 2016. doi: 10.1038/nbt.3519.

[4] Junyue Cao, Jonathan S. Packer, Vijay Ramani, Darren A. Cusanovich, Chau Huynh, Riza Daza, Xiaojie Qiu, Choli Lee, Scott N. Furlan, Frank J. Steemers, Andrew Adey, Robert H. Waterston, Cole Trapnell, and Jay Shendure. Comprehensive single-cell transcriptional profiling of a multicellular organism. *Science*, 357(6352):661–667, August 2017. doi: 10.1126/science.aam8940.

[5] Andre Maia Chagas, Lucia L. Prieto-Godino, Aristides B. Arrenberg, and Tom Baden. The €100 lab: A 3D-printable open-source platform for fluorescence microscopy, optogenetics, and accurate temperature control during behaviour of zebrafish, Drosophila, and Caenorhabditis elegans. *PLOS Biology*, 15(7), July 2017. doi: 10.1371/journal.pbio.2002702.

[6] Alexander Dobin, Carrie A. Davis, Felix Schlesinger, Jorg Drenkow, Chris Zaleski, Sonali Jha, Philippe Batut, Mark Chaisson, and Thomas R. Gingeras. STAR: ultrafast universal RNA-seq aligner. *Bioinformatics*, 29(1):15–21, January 2013. doi: 10.1093/bioinformatics/bts635.

[7] Carl Doersch. Tutorial on Variational Autoencoders. January 2021. arXiv: 1606.05908.

[8] Gökcen Eraslan, Lukas M. Simon, Maria Mircea, Nikola S. Mueller, and Fabian J. Theis. Single cell RNA-seq denoising

using a deep count autoencoder. page 300681, April 2018. doi:10.1101/300681.

[9] C. J. Forman, H. Tomes, B. Mbobo, R. J. Burman, M. Jacobs, T. Baden, and J. V. Raimondo. Openspritzer: an open hardware pressure ejection system for reliably delivering picolitre volumes. *Scientific Reports*, 7(1), May 2017. doi: 10.1038/s41598-017-02301-2.

[10] Adam Gayoso, Romain Lopez, Galen Xing, Pierre Boyeau, Katherine Wu, Michael Jayasuriya, Edouard Melhman, Maxime Langevin, Yining Liu, Jules Samaran, Gabriel Misrachi, Achille Nazaret, Oscar Clivio, Chenling Xu, Tal Ashuach, Mohammad Lotfollahi, Valentine Svensson, Eduardo da Veiga Beltrame, Carlos Talavera-López, Lior Pachter, Fabian J. Theis, Aaron Streets, Michael I. Jordan, Jeffrey Regier, and Nir Yosef. scvi-tools: a library for deep probabilistic analysis of single-cell omics data. page 2021.04.28.441833, April 2021. doi:10.1101/2021.04.28.441833.

[11] Alicia Gibb, Steven Adabie, and Ed Baafi. *Building open source hardware DIY manufacturing for hackers and makers*. Addison-Wesley, Upper Saddle River, NJ, 2015. OCLC: 904585844.

[12] Christopher H. Grønbech, Maximillian F. Vording, Pascal N. Timshel, Capser K. Sønderby, Tune H. Pers, and Ole Winther. scVAE: Variational auto-encoders for single-cell gene expression datas. page 318295, May 2018. doi:10.1101/318295.

[13] Dominic Grün, Anna Lyubimova, Lennart Kester, Kay Wiebrands, Onur Basak, Nobuo Sasaki, Hans Clevers, and Alexander van Oudenaarden. Single-cell messenger RNA sequencing reveals rare intestinal cell types. *Nature*, 525(7568), September 2015. doi: 10.1038/nature14966.

[14] Tamar Hashimshony, Florian Wagner, Noa Sher, and Itai Yanai. CEL-Seq: single-cell RNA-Seq by multiplexed linear amplification. *Cell Reports*, 2(3), September 2012. doi: 10.1016/j.celrep.2012.08.003.

[15] Jörg D. Hoheisel. Microarray technology: beyond transcript profiling and genotype analysis. *Nature Reviews Genetics*, 7(3):200–210, March 2006. doi:10.1038/nrg1809.

[16] Rhys Jones, Patrick Haufe, Edward Sells, Pejman Iravani, Vik Olliver, Chris Palmer, and Adrian Bowyer. RepRap – the replicating rapid prototyper. *Robotica*, 29(1), January 2011. doi: 10.1017/S026357471000069X.

[17] Diederik P. Kingma and Max Welling. Auto-Encoding Variational Bayes. May 2014. arXiv: 1312.6114.

[18] Jocelyn Y. Kishi, Thomas E. Schaus, Nikhil Gopalkrishnan, Feng Xuan, and Peng Yin. Programmable autonomous synthesis of single-stranded DNA. *Nature Chemistry*, 10(2), February 2018. doi:10.1038/nchem.2872.

[19] Allon M. Klein, Linas Mazutis, Ilke Akartuna, Naren Tallapragada, Adrian Veres, Victor Li, Leonid Peshkin, David A. Weitz, and Marc W. Kirschner. Droplet Barcoding for Single-Cell Transcriptomics Applied to Embryonic Stem Cells. *Cell*, 161(5):1187–1201, May 2015. doi:10.1016/j.cell.2015.04.044.

[20] Romain Lopez, Jeffrey Regier, Michael Cole, Michael Jordan, and Nir Yosef. A deep generative model for gene expression profiles from single-cell RNA sequencing. January 2018. arXiv: 1709.02082.

[21] Aaron Lun. Overcoming systematic errors caused by log-transformation of normalized single-cell RNA sequencing data. page 404962, August 2018. doi:110.1101/404962.

[22] Evan Z. Macosko, Anindita Basu, Rahul Satija, James Nemesh, Karthik Shekhar, Melissa Goldman, Itay Tirosh, Allison R. Bialas, Nolan Kamitaki, Emily M. Martersteck, John J. Trombetta, David A. Weitz, Joshua R. Sanes, Alex K. Shalek, Aviv Regev, and Steven A. McCarroll. Highly Parallel Genome-wide Expression Profiling of Individual Cells Using Nanoliter Droplets. *Cell*, 161(5):1202–1214, May 2015. doi: 10.1016/j.cell.2015.05.00.

[23] Páll Melsted, Vasilis Ntranos, and Lior Pachter. The barcode, UMI, set format and BUStools. *Bioinformatics*, 35(21):4472–4473, November 2019. doi: 10.1093/bioinformatics/btz279.

[24] Isaac Nuñez, Tamara Matute, Roberto Herrera, Juan Keymer, Timothy Marzullo, Timothy Rudge, and Fernán Federici. Low cost and open source multi-fluorescence imaging system for teaching and research in biology and bioengineering. *PLOS ONE*, 12(11), November 2017. doi: 10.1371/journal.pone.0187163.

[25] Jonathan S. Packer, Qin Zhu, Chau Huynh, Priya Sivaramakrishnan, Elicia Preston, Hannah Dueck, Derek Stefanik, Kai Tan, Cole Trapnell, Junhyong Kim, Robert H. Waterston, and John I.

Murray. A lineage-resolved molecular atlas of C. elegans embryogenesis at single-cell resolution. *Science*, 365(6459), September 2019. doi: 10.1126/science.aax1971.

[26] Joshua M. Pearce. Building Research Equipment with Free, Open-Source Hardware. *Science*, 337(6100):1303–1304, September 2012. Publisher: American Association for the Advancement of Science.

[27] Joshua M. Pearce. Cut costs with open-source hardware. *Nature*, 505(7485), January 2014. doi: 10.1038/505618d.

[28] Michael C. Pirrung and Edwin M. Southern. The genesis of microarrays. *Biochemistry and Molecular Biology Education*, 42(2):106–113, 2014. doi:10.1002/bmb.20756.

[29] Avi Srivastava, Laraib Malik, Tom Smith, Ian Sudbery, and Rob Patro. Alevin efficiently estimates accurate gene abundances from dscRNA-seq data. *Genome Biology*, 20(1):65, March 2019. doi: 10.1186/s13059-019-1670-y.

[30] William Stephenson, Laura T. Donlin, Andrew Butler, Cristina Rozo, Bernadette Bracken, Ali Rashidfarrokhi, Susan M. Goodman, Lionel B. Ivashkiv, Vivian P. Bykerk, Dana E. Orange, Robert B. Darnell, Harold P. Swerdlow, and Rahul Satija. Single-cell RNA-seq of rheumatoid arthritis synovial tissue using low-cost microfluidic instrumentation. *Nature Communications*, 9(1), February 2018. doi: 10.1038/s41467-017-02659-x.

[31] Valentine Svensson, Roser Vento-Tormo, and Sarah A. Teichmann. Exponential scaling of single-cell RNA-seq in the past decade. *Nature Protocols*, 13(4):599–604, April 2018. doi:10.1038/nprot.2017.149.

[32] Seth R. Taylor, Gabriel Santpere, Alexis Weinreb, Alec Barrett, Molly B. Reilly, Chuan Xu, Erdem Varol, Panos Oikonomou, Lori Glenwinkel, Rebecca McWhirter, Abigail Poff, Manasa Basavaraju, Ibnul Rafi, Eviatar Yemini, Steven J. Cook, Alexander Abrams, Berta Vidal, Cyril Cros, Saeed Tavazoie, Nenad Sestan, Marc Hammarlund, Oliver Hobert, and David M. Miller. Molecular topography of an entire nervous system. *Cell*, 184(16), August 2021. doi: 10.1016/j.cell.2021.06.023.

[33] Sophia C. Tintori, Erin Osborne Nishimura, Patrick Golden, Jason D. Lieb, and Bob Goldstein. A Transcriptional Lineage of the Early C. elegans Embryo. *Developmental Cell*, 38(4):430–444, August 2016. doi: 10.1016/j.devcel.2016.07.025.

[34] F. William Townes, Stephanie C. Hicks, Martin J. Aryee, and Rafael A. Irizarry. Feature selection and dimension reduction for single-cell RNA-Seq based on a multinomial model. *Genome Biology*, 20(1):295, December 2019. doi:10.1186/s13059-019-1861-6.

[35] Bas Wijnen, Emily J. Hunt, Gerald C. Anzalone, and Joshua M. Pearce. Open-Source Syringe Pump Library. *PLOS ONE*, 9(9), September 2014. doi: 10.1371/journal.pone.0107216.

[36] F. Alexander Wolf, Philipp Angerer, and Fabian J. Theis. SCANPY: large-scale single-cell gene expression data analysis. *Genome Biology*, 19(1), February 2018. doi: 10.1186/s13059-017-1382-0.

[37] Luke Zappia, Belinda Phipson, and Alicia Oshlack. Exploring the single-cell RNA-seq analysis landscape with the scRNA-tools database. *PLOS Computational Biology*, 14(6):e1006245, June 2018. doi:10.1371/journal.pcbi.1006245.

[38] Grace X. Y. Zheng, Jessica M. Terry, Phillip Belgrader, Paul Ryvkin, Zachary W. Bent, Ryan Wilson, Solongo B. Ziraldo, Tobias D. Wheeler, Geoff P. McDermott, Junjie Zhu, Mark T. Gregory, Joe Shuga, Luz Montesclaros, Jason G. Underwood, Donald A. Masquelier, Stefanie Y. Nishimura, Michael Schnall-Levin, Paul W. Wyatt, Christopher M. Hindson, Rajiv Bharadwaj, Alexander Wong, Kevin D. Ness, Lan W. Beppu, H. Joachim Deeg, Christopher McFarland, Keith R. Loeb, William J. Valente, Nolan G. Ericson, Emily A. Stevens, Jerald P. Radich, Tarjei S. Mikkelsen, Benjamin J. Hindson, and Jason H. Bielas. Massively parallel digital transcriptional profiling of single cells. *Nature Communications*, 8(1), January 2017. doi: 10.1038/ncomms14049.