Inaugural dissertation

for

obtaining the doctoral degree

of the

Combined Faculaty of Mathematics, Engineering and Natural Sciences

of the

Ruprecht-Karls-University

Heidelberg

Presented by

M.Sc. Constantin Ahlmann-Eltze
born in Kiel, Germany

Oral examination: 08.09.2023

Development of statistical methods for the analysis of single-cell RNA-seq data

Referees: Prof. Dr. Simon Anders

Prof. Dr. Oliver Stegle

To my family.

Abstract

Single-cell RNA-sequencing profiles the transcriptome of cells from diverse populations. A popular intermediate data format is a large count matrix of genes × cells. This type of data brings several analytical challenges. Here, I present three projects that I worked on during my PhD that address particular aspects of working with such datasets:

1. The large number of cells in the count matrix is a challenge for fitting gamma-Poisson generalized linear models with existing tools. I developed a new `R` package called *glmGamPoi* to address this gap. I optimized the overdispersion estimation procedure to be quick and robust for datasets with many cells and small counts. I compared the performance against two popular tools (*edgeR* and *DESeq2*) and find that my inference is 6× to 13× faster and achieves a higher likelihood for a majority of the genes in four single-cell datasets.

2. The variance of single-cell RNA-seq counts depends on their mean but many existing statistical tools have optimal performance when the variance is uniform. Accordingly, variance-stabilizing transformations are applied to unlock the large number of methods with such an requirement. I compared four approaches to variance-stabilize the data based on the delta method, model residuals, inferred latent expression state or count factor analysis. I describe the theoretical strength and weaknesses, and compare their empirical performance in a benchmark on simulated and real single-cell data. I find that none of the mathematically more sophisticated transformations consistently outperform the simple $\log(y/s + 1)$ transformation.

3. Multi-condition single-cell data offers the opportunity to find differentially expressed genes for individual cell subpopulations. However, the prevalent approach to analyze such data is to start by dividing the cells into discrete populations and then test for differential expression within each group. The results are interpretable but may miss interesting cases by (1) choosing the cluster size too small and lacking power to detect effects or (2) choosing the cluster size too large and obscuring interesting effects apparent on a smaller scale. I developed a new statistical framework for the analysis of multi-condition single-cell data that avoids the premature discretization. The approach performs regression on the latent subspaces occupied by the cells in each condition. The method is implemented as an `R` package called *lemur*.

Zusammenfassung

Einzelzell-RNA-Sequenzierung untersucht die Transkriptome von Zellen aus vielfältigen Populationen. Eine gängiges Datenformat ist eine große Matrix mit nicht-negativen ganzzahligen Einträgen (kurz Zählmatrix) und den Dimensionen Gene × Zellen. Diese Art von neuen Daten bringt aber auch viele Herausforderungen für die Analyse mit sich. Ich werde drei Projekte vorstellen, an denen ich während meiner Promotion gearbeitet habe, die einige der Aspekte mit solchen Daten zu arbeiten, adressieren:

1. Die hohe Anzahl an Zellen stellt eine Herausforderung dar, wenn man gamma-Poisson verallgemeinerte lineare Modelle mit existierenden Methoden fitten möchte. Ich habe ein neues R-Paket namen s *glmGamPoi* entwickelt. Ich habe den Algorithmus um die Überdispersion abzuschätzen optimiert, damit dieser schnell und robust auf Datensätzen mit vielen Zellen und kleinen Zahlen funktioniert. Ich habe die Ergebnisse mit denen von zwei gängigen Methoden (*edgeR* und *DESeq2*) verglichen und festgestellt, dass meine Method 6× bis zu 13× schneller ist und für vier Einzelzell-Datensätzen eine höhere Likelihoods für eine Mehrheit der Gene erzielt.

2. Die Varianz von Einzelzell-RNA-seq Zählwerten ist von ihrem Mittelwert abhängig, aber viele existierende statistischen Methoden liefern die besten Ergebnisse, wenn die Varianz gleichmäßig verteilt ist. Daher werden Varianz-stabilisierende Transformationen auf die Zählmatrix angewendet, um die Performanz solcher generischen Methoden zu verbessern. Ich habe vier verschiedene Ansätze für diese Art von Varianzstabilisierung verglichen: Methoden auf Basis der Delta Methode, von Modell-Residuen, der geschätzten unterliegenden Expressionsaktivität und von Zählmatrix-Faktor-Modellen. Ich beschreibe die theoretischen Stärken und Schwächen der Ansätze und vergleiche die Ergebnisse in einem Benchmark auf simulierten und echten Einzel-Zell-Datensätzen empirisch. Die Ergebnisse zeigen, dass keine der mathematisch komplizierten Methoden konsistent besser ist als die einfache $\log(y/s + 1)$ Transformation.

3. Einzel-Zell Daten, die in mehreren experimentellen Bedingungen gemessen wurden, ermöglichen es differentiell exprimierte Gene in einzelnen Zellsubpopulationen zu finden. Jedoch ist der erste Schritt für existierende Methoden die Zellen in diskrete Gruppen zu unterteilen und dann innerhalb jeder Gruppe nach differentieller Expression zu suchen. Die Ergebnisse sind einfach zu interpretieren, aber dieser Ansatz hat das Risiko interessante Muster zu übersehen, (1) da die Gruppengröße zu klein gewählt wird, wodurch nicht genug Evidenz vorhanden ist, um die Muster zu erkennen, oder (2) die Gruppengröße zu groß gewählt wird, wodurch Muster in kleineren Gruppen übersehen werden. Ich habe ein neues statistisches Framework für die Analyse von Einzel-Zell Daten, die in mehreren experimentellen Bedingungen gemessen wurden, entwickelt, dass diese frühzeitige Diskretisierung vermeidet. Der Ansatz basiert auf einer Regression auf den

unterliegenden Unterräumen in denen die Zellen jeder experimentellen Bedingung sich befinden. Die Methode ist als R-Paket unter dem Namen *lemur* implementiert.

Contents

Acknowledgments

Obtaining a doctoral degree is a momentous life event. And over the last 30 years of my life, many big and small moments–and most importantly, people–have helped me get here.

I am deeply grateful to my family, who, without fail, have always been there for me. My mom tirelessly helped me whenever I struggled at school and bought me my first book on programming, despite its sticker price, without hesitation. My dad, who, since my earliest childhood, has kindled my interest in natural sciences and whose open mindedness and curiosity has always been an inspiration. My sister who, besides constant support, contributed to this thesis the correct phonetic transcription of *glmGamPoi*: dʒi əl əm gam ˈpwɑ.

For this journey through the PhD, Maija has been the best companion that I could imagine. Your unending support, warmth, and ever-present encouragement helped me through the difficult periods of the PhD and made the good times even more enjoyable. And I will be ever grateful for resolving the biggest challenge of my PhD: coming up with the name *LEMUR* for my main PhD project.

The last ten years in Heidelberg would not have been the same if it was not for all my good friends: Anna, Jo, Tobias, Sabrina, Vera, Stefan, Daria, Max, Sara, Anja, and Leonard. There would be countless anecdotes to tell here, but maybe it's best to save those for the next Fotoalbum.

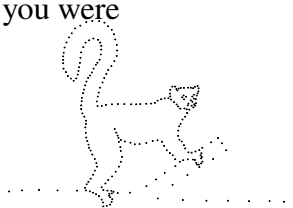
I also want to thank all my colleagues and friends now or previously at EMBL, who made it a joy to come to work every day: my colleagues in the Huber group and all the friends I have made in the last 4 years, particularly Karel and Nadine.

There have been many mentors who influenced my professional interests and helped me succeed. I want to thank my high school teachers, who patiently bore with me when I kept quizzing them *why* something was the way it was. The twelve professors and group leaders who gave me a chance and hosted me for internships and research assistant jobs, which were incredible learning opportunities. Also, I want to thank the people outside Heidelberg who volunteered their time to explain a topic to me or give me feedback on my work. In particular, I want to thank Mike Love, Dmitry Kobak, and Ronny Bergmann.

An important part of developing new methods is to write the software that implements those methods. I am incredibly happy that the R and *Bioconductor* community has always been helpful and welcoming to me. It has been a real joy to be a part of these communities.

I have been lucky to have had an incredible thesis advisory committee: Micheal Boutros, Oliver Stegle, Simon Anders, and Judith Zaugg. They accompanied me through the last four years and gave invaluable feedback to ensure that the projects were as good as they could be. Simon was particularly generous with his time. Our discussions about the state of the single-cell field, what are the relevant problems, and him sitting down with me to explain a mathematical concept in detail have been some of the most valuable learning experiences of my time in Heidelberg.

Lastly, I want to thank Wolfgang, who is just the best boss I could wish for. You believed in my projects and gave me the freedom to pursue my own interests. At the same time, you were



always available when I felt stuck, and without your deep insights, none of my projects would be where they are today.

Research Output

Project	Publication/Preprint	Code availability
glmGamPoi	Bioinformatics (2020)	bioconductor.org/packages/glmGamPoi/
transformGamPoi	Nature Methods (2023b)	bioconductor.org/packages/transformGamPoi/
LEMUR	bioRχiv (2023a)	github.com/const-ae/lemur

1 Introduction

During my PhD, I have (1) developed a tool for fitting gamma-Poisson[1] count models to single-cell RNA-seq data, (2) compared various single-cell preprocessing methods, and (3) developed a new statistical method to find groups of cells that show consistent differential expression without *a priori* assigning the cells to discrete groups. I will explain each of these topics in an individual chapter. Here, I want to give the appropriate context for this work and provide a historical overview of where single-cell RNA-seq originated from and why it is an exciting technology. In addition, I want to review the most important computational innovations that made it possible to gain biological insights from high-throughput transcriptomic data.

1.1 A brief history of transcriptomics

The earliest research article I was able to find discussing the question of differential gene expression is Wilt (1962). It considers the differential gene activity of hemoglobin during embryogenesis of chickens. The term *transcriptome* was coined considerably later by Velculescu et al. (1997) who defined it as the ”set of [expressed] genes”. This is the first paper that surveys all transcripts in an organism (that is, yeast) and kicked off the era of high-throughput transcriptomic characterization of many organisms and experimental conditions.

The word *transcriptome* is a portmanteau of *transcription*, the production of mRNA, and *genome*, the set of all genes in an organism. Unlike the genome, the transcriptome is flexible and responsive to external conditions and thus provides a link between the static genome and the physical characteristics of a cell (Velculescu et al., 1997). Two important physical characteristics of the cell are the proteome (the set of all proteins) and metabolome (the set of all metabolites), so why focus on the intermediary instead of the products that actually do something? Lockhart and Winzeler (2000) answered this question aptly, pointing out ”one reason is simply that protein-based approaches are generally more difficult, less sensitive and have lower throughput than RNA-based ones,” which is still valid 23 years later.

Some of the earliest techniques to study differential expression and the transcriptome were gel based. To this day, Western blots are a quick target confirmation of protein presence; Liang and Pardee (1992) and Welsh et al. (1992) combined Northern blots (detection of cDNA with gels) with PCR amplification of arbitrary mRNAs to compare the set of expressed genes. The method is called *differential display* and was popular throughout the 90s (Liang, 2002). Figure 1A shows an example of a differential display Northern blot comparing the gene expression of heart and kidney cells in mice. Besides reliance on broadly available equipment, one advantage of differential display was that it was not limited to known genes. After the differential display, excision from the gel could be used to sequence a gene. However, identifying differentially expressed genes from differential display was still cumbersome because precise quantification

[1]This is just another name for the negative binomial distribution.

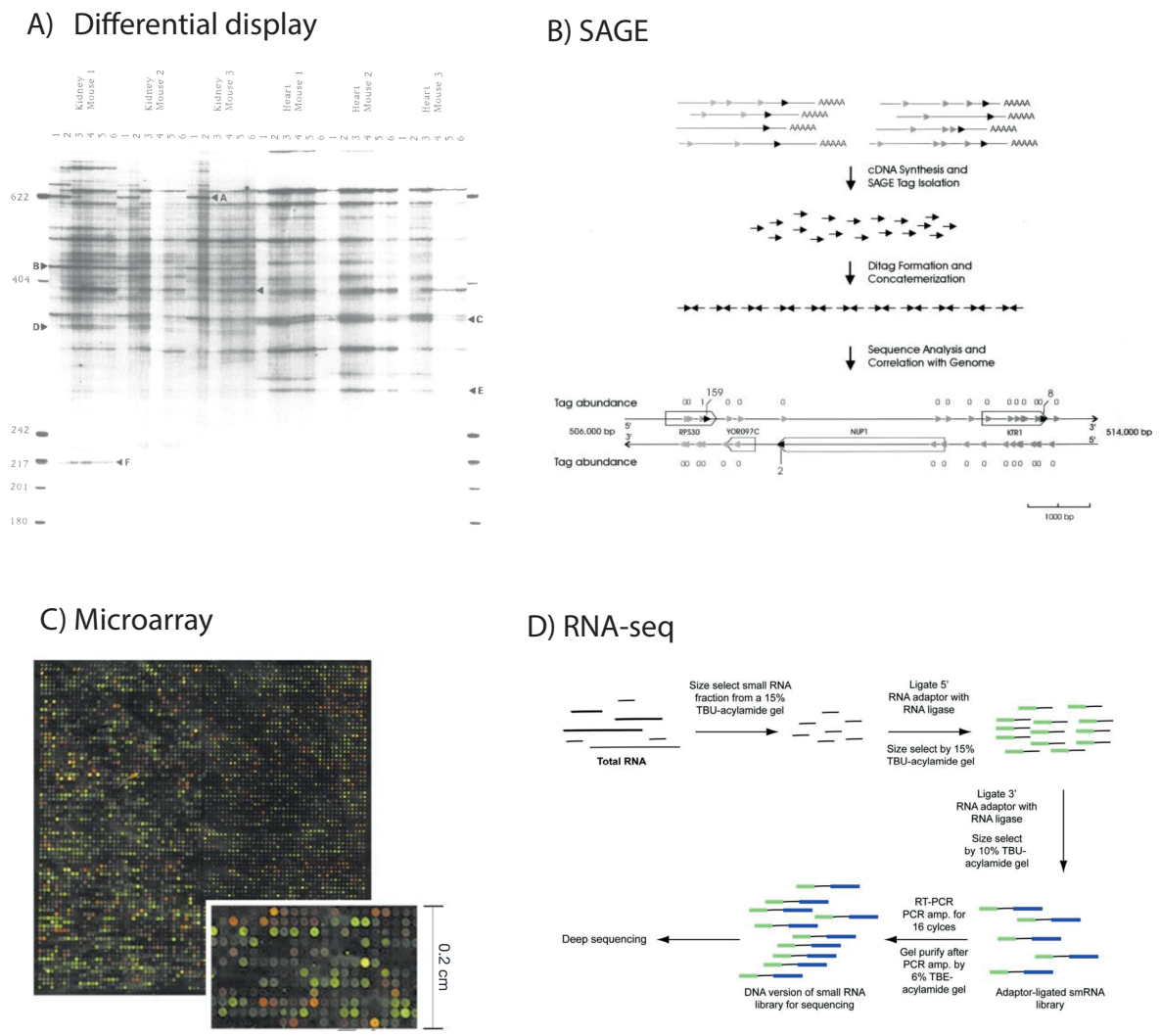


Figure 1 | Overview of popular techniques for studying differential expression in the 90s and 2000s. (A) Northern blot with a differential display of mRNA from mouse kidney and heart. The arrows indicate differentially expressed mRNAs. (B) Schematic of serial analysis of gene expression (SAGE). (C) Image of a two-color microarray. mRNA from two conditions is labeled with a red and green dye, respectively, and the competitive hybridization indicates if a gene is overexpressed in either condition. (D) Schematic of the experimental procedure to conduct an RNA-seq experiment.
The data for panel (A) was generated by Welsh et al. (1992) –reprinted with permission from Nucleic Acid Research. The schematic in panel (B) was produced by Velculescu et al. (1997) and is available as CC-BY-NC-ND. The data for panel (C) is courtesy of J. DeRisi and P. O. Brown, as presented in Lockhart and Winzeler (2000) –reprinted with permission from Springer Nature. The schematic in panel (D) was produced by Lister et al. (2008) –reprinted with permission from Elsevier.

of the expression changes from the gels was difficult and the method relied on mRNA length instead of nucleotide sequence to identify genes.

Two competing techniques developed in the mid-90s were SAGE (serial analysis of gene expression) and microarrays. SAGE was developed by Velculescu et al. (1995) and used in

the landmark study of the yeast transcriptome (Velculescu et al., 1997). It used the growing availability of high-throughput sequencers, creating random 9 or 10 base pair *tags* from all expressed transcripts. These tags are often long enough to be aligned with known gene sequences, and summing up the times a tag matched a gene allowed an absolute quantification of the gene expression (Figure 1B). One limitation of the technique was its low dynamic range which made it less well-suited for differential expression analysis than microarrays (Ding and Cantor, 2004; van Ruissen et al., 2005).

Microarrays use DNA probes immobilized on a surface to measure the hybridization with labeled mRNA. The origins of the technique trace back to the 1960s and 70s (Kafatos et al., 1979; Gillespie and Spiegelman, 1965), when DNA was attached to nitrocellulose membranes and the mRNA was labeled radioactively. However, the throughput of microarrays was limited until glass substrates and fluorescently labeled probes were combined with new ways to deposit the DNA (Figure 1C) (Schena et al., 1995, 1996). These new chips allowed the comparison of ten-thousands of genes. One popular commercial platform for microarrays was Affymetrix (Lockhart et al., 1996). Their chips were used in a landmark study by Golub et al. (1999), who used gene expression signatures to distinguish AML and ALL patients. Microarrays were popular because of their sensitivity, high throughput, and large dynamic range (Ding and Cantor, 2004). However, they are fundamentally limited to a fixed set of DNA probes.

The invention of RNA-seq, developed in 2007 and 2008, overcame these problems (Emrich et al., 2007; Lister et al., 2008). Figure 1D gives an overview of the experimental procedure. It uses short-read sequencing technologies (commercialized, for example, by Illumina), which replaced the less efficient Sanger sequencing used in SAGE. In addition, RNA-seq has made it straightforward to quantify alternative splicing as it only requires counting of reads spanning splice sides (Wang et al., 2009). An important difference compared to microarrays is that RNA-seq does not need to handle background subtraction, which complicates the analysis of microarray data, because the RNA-seq data are integer counts. Finally, the quantification of gene expression levels with RNA-seq is more accurate (using qPCR as a reference) than with microarrays (Nagalakshmi et al., 2008).

In the last 15 years, RNA-seq has become the dominant technology for transcriptomic analysis. And almost since its inception, various extensions have been developed. In addition to the short-read sequencing from Illumina, long-read sequencing approaches by Pacific Biosciences and Oxford Nanopore are popular (Figure 2) (Eid et al., 2009; Garalde et al., 2018). They alleviate the problem of mapping ambiguity and enable better resolution of isoforms. Yet, short-read sequencing is still the most popular technology for transcriptomics because of its lower error rate, higher throughput, and lower costs (Stark et al., 2019).

Only two years after RNA-seq was established, the first sequencing of a single cell was reported (Tang et al., 2009). In the study five oocytes were separated into individual wells and sequenced them using a modified RNA-seq protocol. Figure 3 gives an overview of the different steps in a single-cell experiment and some of the popular technologies. Among these

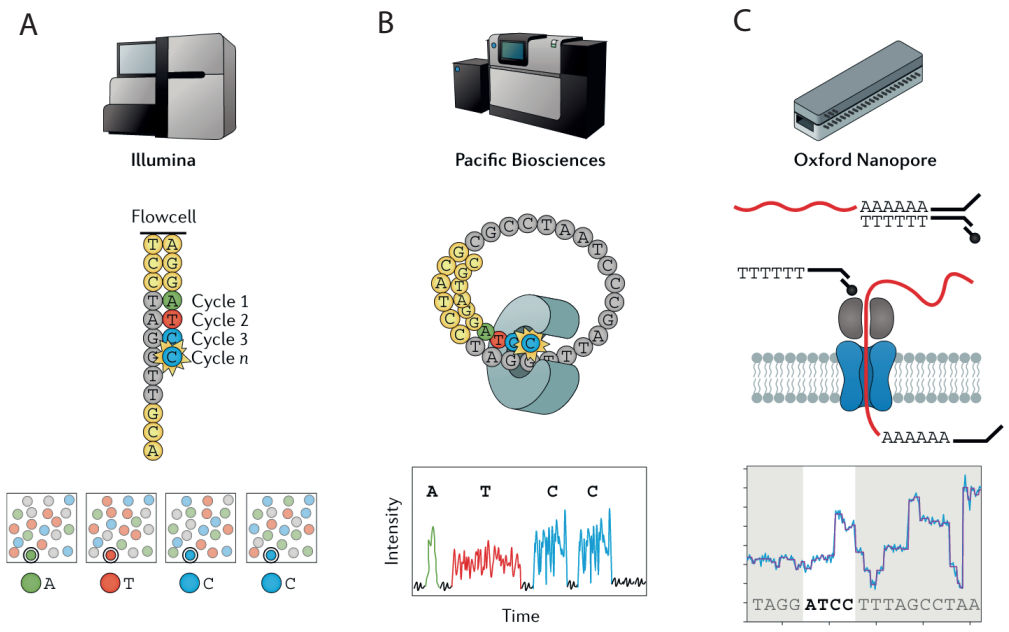


Figure 2 | Overview of RNA-sequencing technologies. (A) Schematic of Illumina short read sequencing: sequencing proceeds base by base, and each newly incorporated nucleotide produces a distinct fluorescent signal. (B) Schematic of Pacific Bioscience long read sequencing: at the bottom of a nano-well, an immobilized polymerase extends fluorescently tagged nucleotides. (C) Schematic of Oxford Nanopore long read sequencing: DNA strands are sucked through a nanopore by a motor protein; the change in current across a membrane is used to detect the nucleotide identity.
The schematic was produced by Stark et al. (2019) –reprinted with permission from Springer Nature.

is the plate-based Smart-seq protocol that sequences full-length transcripts (Picelli et al., 2013, 2014). Besides limited throughput, the main disadvantage of the Smart-seq protocol was the lack of unique molecular identifiers (UMIs) until the introduction of Smart-seq3 three years ago (Hagemann-Jensen et al., 2020). UMIs have proven to be an indispensable tools for solving the problem of inflated counts of some transcripts due to the PCR amplification step (Islam et al., 2014; Stark et al., 2019).

The most popular single-cell sequencing technology is based on droplet-based microfluidics (Macosko et al., 2015; Klein et al., 2015). This technology was commercialized by 10x Genomics, who scaled the system up to tens of thousands of cells (Zheng et al., 2017). Since the initial single-cell studies 14 years ago, the number of cells per study has continuously increased (Figure 4). There exist multiple single-cell studies comprising millions of cells, and the largest study to date profiled 12.4 million single nuclei (Qiu et al., 2023).

The single-cell field is maturing, but there are still several rapidly progressing subfields. There is a lot of excitement around spatially resolved single-cell measurements. Two popular approaches are recording the spatial information during library preparation (e.g., 10x Visium) or using barcoded antibodies that are identified using repeated cycles of fluorescent staining

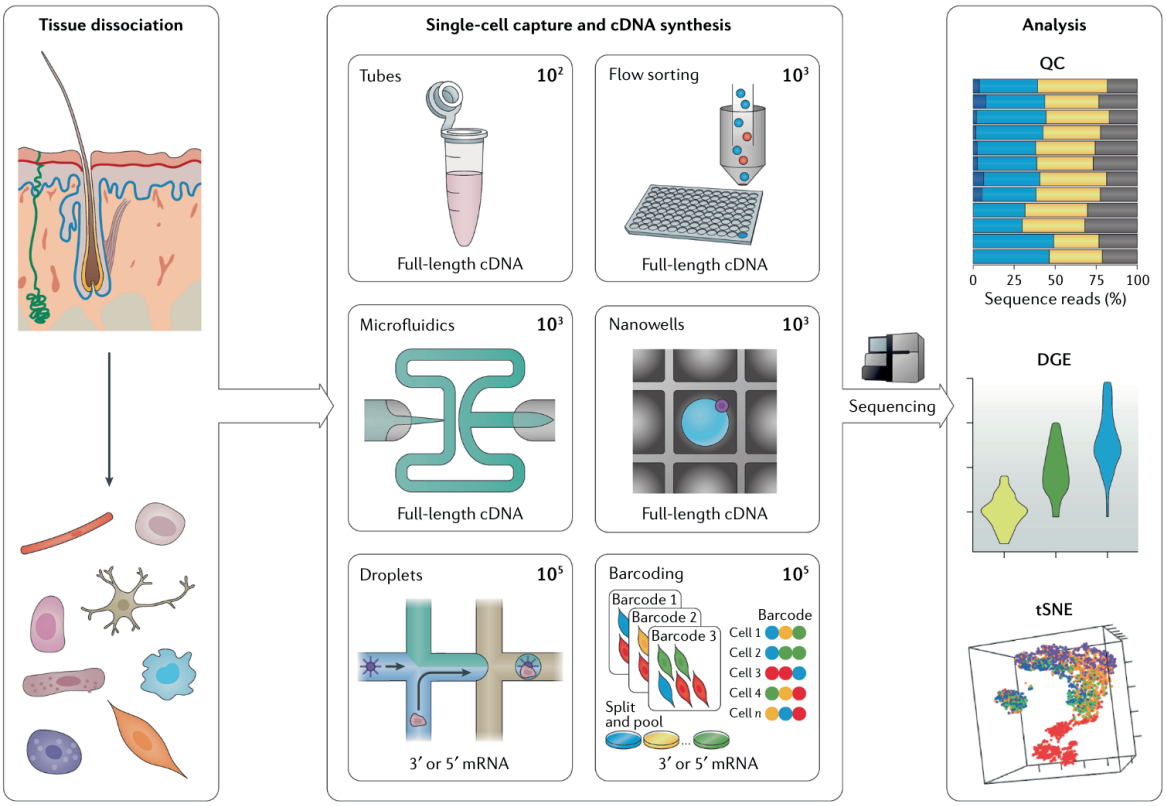


Figure 3 | Overview of single-cell technologies. Steps of a single-cell RNA-seq experiment with an overview of the different single-cell capture approaches and common analytical outputs. The schematic was produced by Stark et al. (2019) –reprinted with permission from Springer Nature.

(CODEX) (Goltsev et al., 2018).

To date, single-cell studies have profiled many different tissues, and large consortia have profiled ten human organs (Regev et al., 2017), the human brain (Siletti et al., 2022), and multiple mouse organs (Schaum et al., 2018). In parallel, there is an increasing interest in comparative single-cell analysis, where multiple conditions are profiled to investigate how an experimental condition, such as drug treatment, changes gene expression across cell types.

Much of the progress in transcriptomic analysis is due to technological advances. In parallel, the birth of high-throughput biological studies and the predominance of new data types (e.g., RNA-seq count data) have spurned many innovations in applied statistics and the development of countless tools. In the next section, I want to give an overview of some of the milestones in the analysis of transcriptomic data.

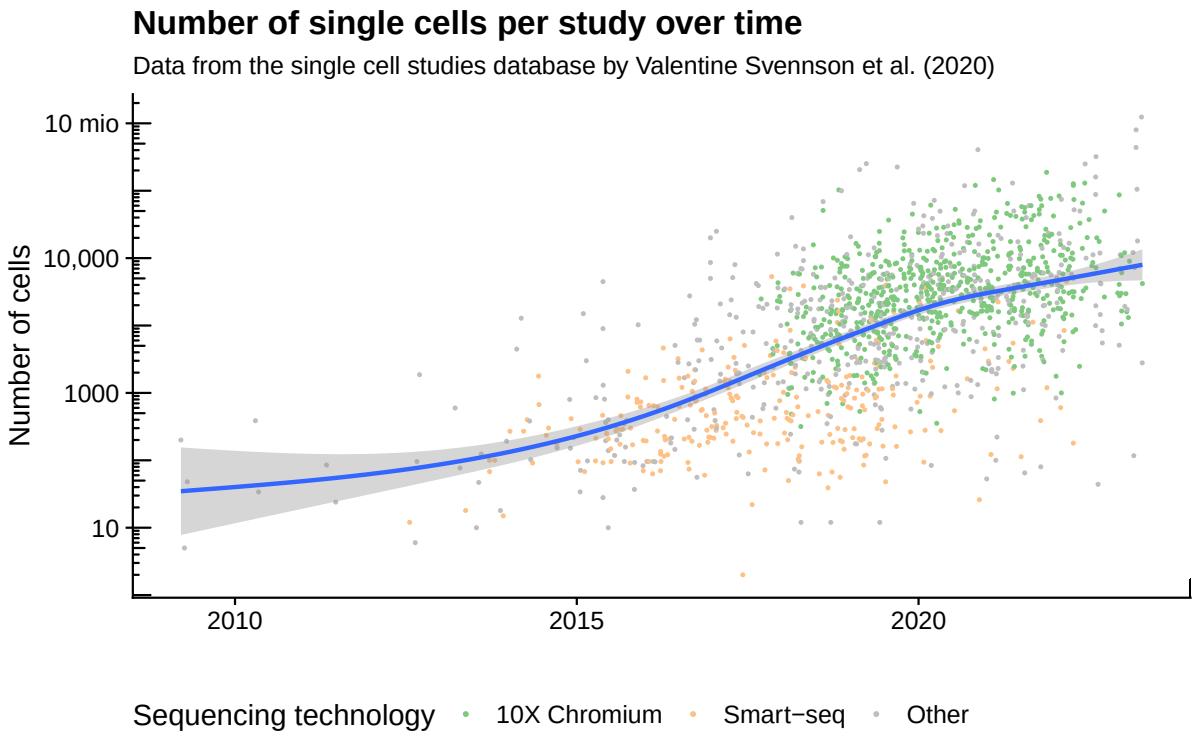


Figure 4 | Number of single cells per study over time. Each point is one study in the database, and the blue line is a generalized additive model smoothing fit. The y-axis is log-transformed. The data for this figure comes from a curated database (Svensson et al., 2020), and the plot was generated by me.

1.2 A brief history of transcriptomics data analysis

On the 26th of November 2001, Robert Gentleman sent the inaugural email to the Bioconductor mailing list[2]. Bioconductor's stated aims were "transparency, pursuit of reproducibility and efficiency of development" (Gentleman et al., 2004). From the beginning, it was connected with the analysis of microarray data which at the time was the method of choice for profiling the transcriptome.

The paper by Gentleman et al. (2004) introduced the Bioconductor project to a wider audience. It demonstrated Bioconductor's capabilities using an Affymetrix microarray data analysis as an example. It showed the *affy* package to load data, *expressionSet* to represent the data in memory, and *limma* to perform inferential statistics. The *affy* package was built to give researchers a more flexible way to work with microarray data than the manufacturer-provided software offered (Gautier et al., 2004). The *expressionSet* was a predecessor to *SummarizedExperiment* and stored a matrix together with additional row and column data (called *featureData* and *phenoData*) (Falcon et al., 2006).

Limma is still one of the most important software packages for analyzing biological high-throughput transcriptomic data. It was originally developed for normalizing and assessing

[2]Within 72 hours, Martin Maechler reminded him that user-contributed code is bundled as *packages* and not *libraries*, which since then has become a recurring theme (`lapply(c(58, 104, 161), fortunes::fortune)`)

differential expression in microarray data (Smyth and Speed, 2003; Smyth, 2004). It is best known for implementing an efficient and statistically sound approach to shrink the variance estimates and thus boosting power to detect differential expression compared with simple linear models (Tusher et al., 2001; Lönnstedt and Speed, 2002; Smyth, 2004).

Originally, microarrays used a two-color system to directly compare the hybridization of two experimental conditions (Schena et al., 1995, 1996). This provided a direct but noisy comparison of their relative expression levels; soon, analysts realized that replicates were essential for reliable differential expression analysis (Yang and Speed, 2002; Allison et al., 2006). This also made single-channel approaches more attractive because the experimental designs for two-color microarrays got increasingly complex (Lipshutz et al., 1999; Yang and Speed, 2002; Kathleen Kerr, 2003).

The variance of a microarray spot intensity measurement depends on the intensity mean. Originally, this was accounted for by using a log transformation (Lockhart et al., 1996). Chen et al. (1997) derived the confidence intervals for the ratio of two microarray measurements for a constant coefficient of variation. This derivation was complicated and not easily generalized to other experimental designs. Thus, Huber et al. (2002) suggested a variance-stabilizing transformation for microarray data based on the delta method, which allowed plugging the transformed values into linear models and use tools such as limma.

The onset of high throughput biological testing also led to reexamination of how multiple testing adjustment was done. Benjamini and Hochberg (1995) had developed the concept of the false discovery rate (FDR). However, only the widespread use of microarrays and quantitative trait loci made the FDR broadly accepted (Benjamini, 2010). In the following years, there were many efforts to improve the multiple testing procedures specifically for microarrays, for example, by working with positive or local false discovery rates (Efron and Tibshirani, 2002; Efron, 2005; Storey, 2002, 2003) or weighting the hypotheses using independent covariates (Bourgon et al., 2010; Ignatiadis et al., 2016).

In 2008, soon after the first RNA-seq studies, Marioni et al. (2008) compared the reproducibility of RNA-seq counts, found that the counts mostly varied within the expected range of Poisson noise, and concluded that replicates might be dispensable for differential expression analysis. However, they only considered technical replicates, and in 2009 and 2010 two packages, edgeR and DESeq, were released that modeled overdispersed counts to account for the additional biological variation between biological replicates. *edgeR* (Robinson et al., 2009) and *DESeq* (Anders and Huber, 2010) remain, to this day, the most popular tools for analyzing bulk RNA-seq data. They were both built around generalized linear models with a gamma-Poisson (another name for the negative binomial) distribution. Both tools are still actively developed and have gained many improvements over the years: McCarthy et al. (2012) introduced the Cox-Reid correction for the overdispersion estimation, Lund et al. (2012) used a quasi-likelihood formulation to improve the estimation of the overdispersion shrinkage, Love et al. (2014) presented *DESeq2* which modernized the programming interface, enabled shrinking of log fold change estimates,

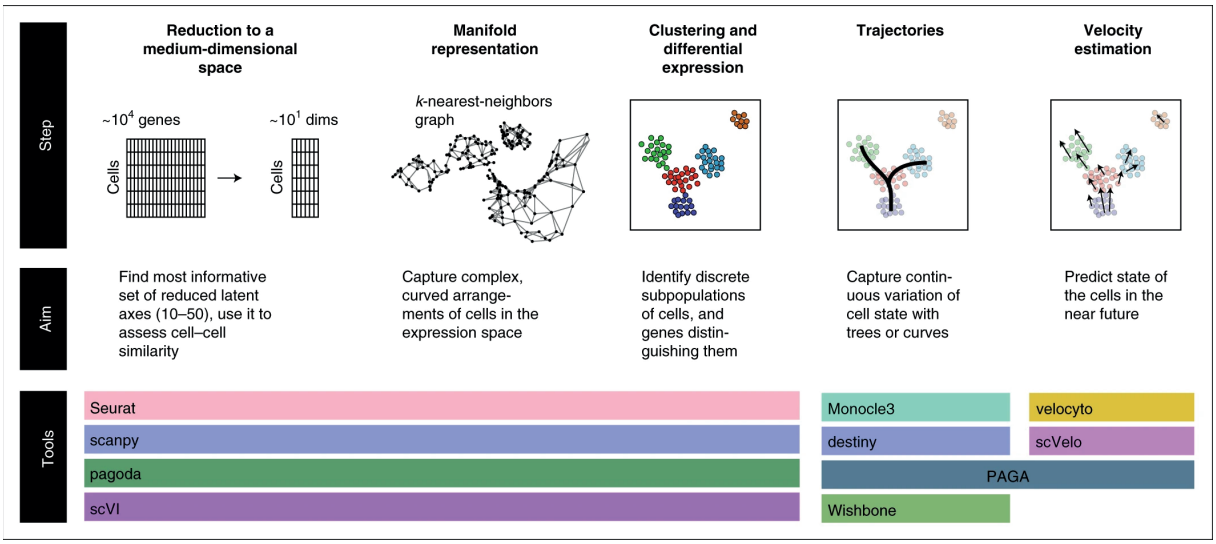


Figure 5 | Overview of a potential computational analysis of a single-cell experiment The schematic was produced by Kharchenko (2021) –reprinted with permission from Springer Nature.

and added independent hypothesis filtering.

The aim of many early single-cell studies was more exploratory than typical bulk RNA-seq experiments. Accordingly, the focus of many newly developed tools for single-cell data was on dimensionality reduction (e.g., with tSNE or UMAP (Van der Maaten and Hinton, 2008; Kobak and Berens, 2019; McInnes et al., 2018; Becht et al., 2019)), data integration (e.g., mutual nearest neighbor and Harmony (Haghverdi et al., 2018; Butler et al., 2018; Korsunsky et al., 2019)), clustering (e.g., Louvain and Leiden (Blondel et al., 2008; Traag et al., 2019)), and computations on neighborhood graphs (e.g., MELD and milo (Burkhardt et al., 2021; Dann et al., 2022)). Figure 5 shows an example analysis workflow and mentions some popular tools.

Whereas many bulk RNA-seq analyses were conducted using `R` and *Bioconductor*, many tools for single-cell analysis are now developed in `Python`. Popular examples are *Scanpy* (Wolf et al., 2018), *scVI* (Lopez et al., 2018), and *Squidpy* (Palla et al., 2022).

When single-cell data was new, there was momentum around developing new tools for differential expression analysis, motivated by the assumption that single-cell counts follow a zero-inflated negative binomial distribution (Kharchenko et al., 2014; Finak et al., 2015). However, Svensson (2020) showed that single-cell data with UMIs are not zero-inflated. Crowell et al. (2020) benchmarked the differential expression performance of these new tools (*MAST*, *scDD*, mixed models with *lme4*) for differential expression analysis and found that they did not outperform using edgeR with pseudo-bulks.

2 glmGamPoi

In this chapter, I will describe how I developed a method to efficiently fit gamma-Poisson generalized linear models on single-cell count data. I published this project with Wolfgang Huber as an application note in Oxford Bioinformatics under the title "glmGamPoi: fitting Gamma-Poisson generalized linear models on single-cell count data" (Ahlmann-Eltze and Huber, 2020) and as an `R` package on Bioconductor. The name of the package is pronounced dʒi əl əm gam ˈpwɑ.

2.1 Generalized linear models

2.1.1 Theory of generalized linear models

Generalized linear models extend the framework of linear models to other data distributions. A key assumption of linear models is the constant variance of the residuals; generalized linear models relax that assumption to any mean-variance relationship described by an exponential dispersion model (EDM) (Dunn and Smyth, 2018). They achieve this by introducing a link function g that connects the linear predictor with the observed values

$$\begin{aligned} \eta &= X\beta \\ Y &\sim \text{EDM}(\mu = g^{-1}(\eta), \phi). \end{aligned} \tag{1}$$

Here, X is a design matrix of size $N \times K$, where N is the number of observations and K is the number of covariates. β are the coefficients found by the generalized linear model fit. η is the linear predictor. Y are the observations, and the domain of Y depends on the exponential dispersion model D (e.g., binary for logistic regression, non-negative counts for Poisson or gamma-Poisson, positive for gamma regression). ϕ is the dispersion parameter, and we assume that it is known.

The density function of the exponential dispersion models is defined as

$$\mathcal{P}_{\text{EDM}}(y; \theta, \phi) = a(y, \phi) \exp\left(\frac{y\theta - \kappa(\theta)}{\phi}\right). \tag{2}$$

θ is the canonical parameter, $\kappa(\theta)$ is called the cumulant function and known, and $a(y, \phi)$ is the normalizing function that ensures that the eq. (2) has an area of 1. See Table 5.1 of Dunn and Smyth (2018) for a table defining θ, κ and ϕ for the most common types of exponential dispersion models.

For illustration, I will go through two relevant examples: the Poisson and gamma-Poisson distributions. The probability density function of the Poisson distribution is

$$\mathcal{P}_{\text{Poisson}}(y; \mu) = \frac{\mu^y \exp(-\mu)}{y!}. \tag{3}$$

To emphasize the relation of eq. (3) and eq. (2), I rewrite eq. (3) as

$$\mathcal{P}_{\text{Poisson}}(y;\mu) = \frac{1}{y!}\exp\left(y\log(\mu) - \mu\right). \tag{4}$$

This demonstrates that $\theta = \log\mu$ is the canonical parameter, the cumulant function is $\kappa(\theta) = \mu$, the normalization function is $a(y,\phi) = 1/y!$, and the dispersion is $\phi = 1$. This proves that the Poisson distribution is an exponential distribution model.

The probability density function of the gamma-Poisson distribution is (Ahlmann-Eltze, 2021)

$$\mathcal{P}_{\text{gamma-Poisson}}(y;\mu,\alpha) = \frac{\Gamma\left(y + 1/\alpha\right)}{\Gamma\left(1/\alpha\right)\Gamma\left(y+1\right)}\left(\frac{\mu\alpha}{1+\mu\alpha}\right)^{y}\left(\frac{1}{1+\mu\alpha}\right)^{1/\alpha}, \tag{5}$$

which I will again rewrite to emphasize the relation to eq. (2)

$$\mathcal{P}_{\text{GP}}(y;\mu,\alpha) = a(y,\alpha)\exp\left(y\log\left(\frac{\mu\alpha}{1+\mu\alpha}\right) - \frac{1}{\alpha}\log\left(1+\mu\alpha\right)\right), \tag{6}$$

where the canonical parameter is $\theta = \log\left(\frac{\mu\alpha}{1+\mu\alpha}\right)$, the cumulant function is[3] $\kappa(\theta) = \frac{1}{\alpha}\log\left(1+\mu\alpha\right)$ and the normalizing function is $a(y,\alpha) = \frac{\Gamma(y+1/\alpha)}{\Gamma(1/\alpha)\Gamma(y+1)}$. Importantly, eq. (6) shows that $\phi = 1$. A common source of confusion is that α is sometimes referred to as *dispersion* which suggests a relation with the dispersion of the exponential dispersion models, even though α and ϕ play distinct roles. For this reason, I will refer to α as the *overdispersion*.

An important property of exponential dispersion models is their mean-variance relationship. A specific choice for the mean-variance relationship uniquely determines the exponential dispersion model class. For example, the mean-variance relationship of

- a Normal distribution is $V(\mu) = 1$,
- a Poisson distribution is $V(\mu) = \mu$,
- and a gamma-Poisson distribution is $V(\mu) = \mu + \alpha\mu^2$.

This property also holds in reverse, that is, the Poisson distribution is the only exponential dispersion model which has a mean-variance relation of $V(\mu) = \mu$.

A powerful generalization using the framework of exponential dispersion models is the introduction of quasi-Poisson and quasi-gamma-Poisson distributions (Lund et al., 2012). They extend the familiar mean-variance relationship by introducing a $\phi \neq 1$. For example, the mean-variance relationship of the quasi-Poisson model is $V(\mu) = \phi\mu$. No closed-form probability density function exists for this mean-variance relationship, but the generic estimators of the coefficients and corresponding standard errors still apply. The concept of quasi-likelihoods is important for the introduction of variance shrinkage estimators.

[3] This definition differs from the result given in Table 5.1 in Dunn and Smyth (2018), who define $\kappa(\theta) = -\log(1-\exp(\theta))$, by a factor of $1/\alpha$.

2.1.2 Inference of the coefficients in generalized linear models

The success of generalized linear models is because a single algorithm for the inference of the coefficients β can be applied to all instances of the exponential dispersion models. The algorithm is based on the theory of maximum likelihood and works by iteratively calculating the coefficients' first and expected second derivatives.

I optimize the coefficients β with respect to the log-likelihood of the distribution

$$\ell(\boldsymbol{\beta}; \boldsymbol{y}) = \sum_{i}^{N} \log \mathcal{P}(y_i; \mu = g^{-1}(X_{i:}\boldsymbol{\beta})). \tag{7}$$

I use the notation $X_{i:}$ to indicate that I extract the i-th row vector from the design matrix X with N rows (i.e. observations) and K columns (i.e.,, the number of covariates).

The vector with the partial derivatives of eq. (20) is called the *score vector*

$$U(\boldsymbol{\beta}) = \frac{\partial \ell(\boldsymbol{\beta}; \boldsymbol{y})}{\partial \boldsymbol{\beta}}. \tag{8}$$

The derivative depends on the log-likelihood of the distribution and the link function (eq. (1)). Specifically, for one coefficient β_k

$$\begin{aligned} U(\beta_k) &= \frac{\partial \ell(\boldsymbol{\beta}; \boldsymbol{y})}{\partial \beta_k} \\ &= \frac{\partial \ell(\boldsymbol{\beta}; \boldsymbol{y})}{\partial \mu} \frac{\mu}{\partial \beta_k} \\ &= \sum_{i}^{N} w_i \frac{\mathrm{d}\eta_i}{\mathrm{d}\mu_i} (y_i - \mu_i) x_{ki}. \end{aligned} \tag{9}$$

In the last step, we use the definition of the cumulant-generating function

$$\begin{aligned} \frac{\mathrm{d}\kappa(\theta)}{\mathrm{d}\theta} &= \mu \\ \frac{\mathrm{d}^2\kappa(\theta)}{\mathrm{d}\theta^2} &= V(\mu). \end{aligned} \tag{10}$$

and store the working weights as

$$w_i = \frac{1}{V(\mu_i)(\mathrm{d}\eta_i/\mathrm{d}\mu_i)^2}. \tag{11}$$

The matrix of second derivatives is called the *observed information matrix*

$$\mathcal{J}_{kk'}(\boldsymbol{\beta}) = -\frac{\partial^2 \ell(\boldsymbol{\beta}; \boldsymbol{y})}{\partial \beta_k \partial \beta_{k'}}. \tag{12}$$

It turns out that for generalized linear models, it is more convenient to work with the expected

information matrix, which is called the *Fisher information matrix*

$$\mathcal{I}(\boldsymbol{\beta}) = -\mathbb{E}\left[\frac{\partial^2 \ell(\boldsymbol{\beta}; \boldsymbol{y})}{\partial \beta_k \partial \beta_{k'}}\right] = \mathbb{E}[\mathcal{J}(\boldsymbol{\beta})]. \tag{13}$$

The Fisher information is a property of the model and independent of the observed values. It is calculated as

$$\mathcal{I} = \frac{1}{\phi} X^T \operatorname{diag}(\boldsymbol{w}) X. \tag{14}$$

To find the optimal $\hat{\boldsymbol{\beta}}$, we set the score function equal to zero $U(\boldsymbol{\beta}) = 0$. The multivariate generalization of the Newton method to find the root of a function ($x^{(t+1)} = x^{(t)} - f(x^{(t)})/f'(x^{(t)})$) is called the Newton-Raphson algorithm

$$\boldsymbol{\beta}^{(t+1)} = \boldsymbol{\beta}^{(t)} + \mathcal{J}(\boldsymbol{\beta}^{(t)})^{-1} U(\boldsymbol{\beta}^{(t)}). \tag{15}$$

As we use the Fisher information matrix $\mathcal{I}$ instead of the observed information $\mathcal{J}$, the procedure is called Fisher scoring and update the parameters by calculating

$$\boldsymbol{\beta}^{(t+1)} = \boldsymbol{\beta}^{(t)} + \mathcal{I}(\boldsymbol{\beta}^{(t)})^{-1} U(\boldsymbol{\beta}^{(t)}). \tag{16}$$

2.1.3 Inference of the coefficients in glmGamPoi

In glmGamPoi, I implemented the Fisher scoring algorithm for gamma-Poisson distributed values with the canonical link function (i.e., $g^{-1}(\eta) = \exp \eta$). Accordingly, the working weights are

$$w_i = \frac{\mu_i}{1 + \mu_i \alpha}. \tag{17}$$

The derivative of the link function is

$$\frac{\mathrm{d}\eta_i}{\mathrm{d}\mu_i} = 1/\mu_i. \tag{18}$$

When I plug these definitions into eq. (9) and rewrite the expression using matrix notations, I find that

$$U(\boldsymbol{\beta}) = \operatorname{diag}(\boldsymbol{w}) \frac{\boldsymbol{y} - \hat{\boldsymbol{\mu}}}{\boldsymbol{\mu}}. \tag{19}$$

To find the inverse of the Fisher information matrix $\mathcal{I}(\boldsymbol{\beta}^{(t)})^{-1}$ in eq. (16) efficiently and reliably, I use a QR decomposition-based linear regression solver. This algorithm is the same that is used by other popular implementations of gamma-Poisson generalized linear models such as DESeq2 and edgeR (Love et al., 2014; Robinson et al., 2009).

In practice, the Fisher scoring algorithm can diverge. I avoid this problem by checking after each step if the proposed step decreases the deviance. If the deviance is not decreased, I assume that the step size is too large and keep halving it until the deviance increases or a maximum

number of iterations is reached.

Unlike edgeR and DESeq2, glmGamPoi supports arbitrary ridge penalties on the coefficients. This is helpful because they can help avoid overfitting, allow us to fit models with design matrices containing colinear columns, and can be used to fit generative additive models (GAMs) for smoothing (an idea by Koen van den Berge). The ridge penalty is provided as a matrix Λ of size $K \times K$ and changes the optimization function eq. (20)

$$\arg\min_{\boldsymbol{\beta}} -\ell(\boldsymbol{\beta}; \boldsymbol{y}) + \boldsymbol{\beta}^T \Lambda \boldsymbol{\beta} \tag{20}$$

Hoerl and Kennard (1970) shows that we can account for the modified optimization function by adding Λ to the Fisher information matrix in eq. (16)

$$\boldsymbol{\beta}^{(t+1)} = \boldsymbol{\beta}^{(t)} + \left(\mathcal{I}(\boldsymbol{\beta}^{(t)}) + \Lambda\right)^{-1} U(\boldsymbol{\beta}^{(t)}). \tag{21}$$

2.1.4 Inference of the overdispersion in generalized linear models

The theory of gamma-Poisson generalized linear models assumes that the overdispersion α is known. In reality, we usually do not know α and need to estimate in parallel to the coefficients. Following the example of DESeq2, I first approximate the overdispersions, fit the coefficients $\boldsymbol{\beta}$, and then use those coefficients to generate the final overdispersion estimates:

1. Approximate the overdispersion based on the mean-variance relationship.

2. Approximate the coefficients for each gene fitting a linear model on the log-normalized counts.

3. Fit the coefficients using the approximated overdispersions by optimizing eq. (20).

4. Fit the overdispersion values using the coefficient estimates. Optionally, shrink the overdispersion values.

5. Fit the coefficients again using the (shrunken) overdispersion estimates by optimizing eq. (20).

In step 1, I approximate the overdispersion values by inverting the mean-variance relation of the gamma-Poisson distribution

$$V(\mu) = \mu + \alpha\mu^2. \tag{22}$$

I calculate the empirical mean and variance of each gene and approximate the overdispersion as

$$\alpha_g = \frac{\mathrm{var}(Y_{g:}) - \mathrm{mean}(Y_{g:})}{\mathrm{mean}(Y_{g:})^2}. \tag{23}$$

In most circumstances, this approach will overestimate the overdispersion because it does not account for the variability explained by the model matrix.

In step 3, I fit the overdispersion by maximizing the log-likelihood of the Gamma-Poisson GLM with respect to the overdispersion

$$\begin{aligned}\arg\max_{\alpha} \ell(\alpha; \boldsymbol{y}, \boldsymbol{\mu}) &= \sum_{c}^{C} \log\left(\frac{\Gamma\left(y_c + 1/\alpha\right)}{\Gamma\left(1/\alpha\right)\Gamma\left(y_c + 1\right)}\left(\frac{\mu_c\alpha}{1+\mu_c\alpha}\right)^{y_c}\left(\frac{1}{1+\mu_c\alpha}\right)^{1/\alpha}\right) \\ &\propto -C\log(1/\alpha) - C\frac{\log(\alpha)}{\alpha} + \sum_{c}\log\Gamma(y_c + 1/\alpha) + \\ &\quad \sum_{c}(-y_c - 1/\alpha)\log(\mu_c + 1/\alpha).\end{aligned} \tag{24}$$

I use the `nlminb` function to optimize eq. (24) to which I provide the first and second derivatives.

A computational bottleneck in the optimization is the repeated calculation of the log-gamma function (and its derivatives, the digamma and trigamma functions). I speed up the process by using the fact that most counts in single-cell data are small integers. I tabulate the counts for each gene and evaluate the log-gamma function only once for each unique integer.

2.1.5 Overdispersion shrinkage

Empirical Bayesian shrinkage increases the power to detect differential expression by sharing information about the overdispersion across genes. The method was pioneered by limma for testing differences in microarray data (Smyth, 2004). Lund et al. (2012) extended the idea to count data using a quasi-gamma-Poisson model with a mean-variance relationship of

$$V(\mu) = \phi(\mu + \alpha\mu^2). \tag{25}$$

I fit a local median regression of the overdispersion estimates per gene depending on the mean gene expression. The fitted values for each gene are called α_{trend}.

I convert the maximum likelihood overdispersion estimates to quasi-likelihood dispersions by positing that the variance is constant

$$\phi = \frac{1 + \mu\alpha_{\text{ML}}}{1 + \mu\alpha_{\text{trend}}}. \tag{26}$$

I infer the parameters of the dispersion prior using maximum likelihood and, lastly, shrink the quasi-likelihood overdispersion estimates by calculating the mean between trended prior and quasi-likelihood overdispersion estimates.

2.1.6 Differential expression testing

I find differentially expressed genes using a quasi-likelihood ratio test following the work of Lund et al. (2012). The motivation for this statistical test comes from the fact that under the null hypothesis for two nested models, the logarithm of a likelihood ratio (or equivalently, the difference of the log-likelihood) of the full and reduced model with a known dispersion ϕ follows a χ^2-distribution

$$L = 2(\ell_{\text{full}} - \ell_{\text{red}}) \sim \chi^2(K_{\text{full}} - K_{\text{red}}). \tag{27}$$

This test corresponds to a z-test for continuously distributed data. The equivalent of a t-test, i.e., where we account for the fact that ϕ was estimated from the data, is the F-test

$$\frac{L_{\phi_{\text{QL}}}}{K_{\text{full}} - K_{\text{red}}} \sim F(K_{\text{full}} - K_{\text{red}}, N - K_{\text{full}}). \tag{28}$$

I use this relation to calculate the p-value for the comparison of the full and reduced model for each gene.

2.2 Implementation

After presenting the theory behind the gamma-Poisson generalized linear models, I will turn to the specific implementation decisions that I made. I designed glmGamPoi with three goals:

1. Higher speed compared with existing implementations.
2. Reliable overdispersion estimates on datasets with many small counts.
3. Handling arbitrarily large datasets.

I achieved the first priority by grouping the evaluation of the log-gamma function by distinct integers. For the second priority, I included lower and upper bounds based on the Taylor series of the first and second derivatives to avoid divergent steps. glmGamPoi supports arbitrarily large single-cell datasets by abstracting the internal memory representation of the count, mean and offset matrices. I used the DelayedArray and beachmat packages (Pages et al., 2022; Lun et al., 2018).

In Figure 6, I show the most important external and internal functions of glmGamPoi and the order in which they are called. The functions with a blue background were implemented in C++ for efficiency.

2.2.1 Integration with the single-cell ecosystem

Besides its use as a standalone package, glmGamPoi has been integrated with two popular packages for the analysis of single-cell transcriptomic data: DESeq2 (Love et al., 2014) and sctransform (Hafemeister and Satija, 2019).

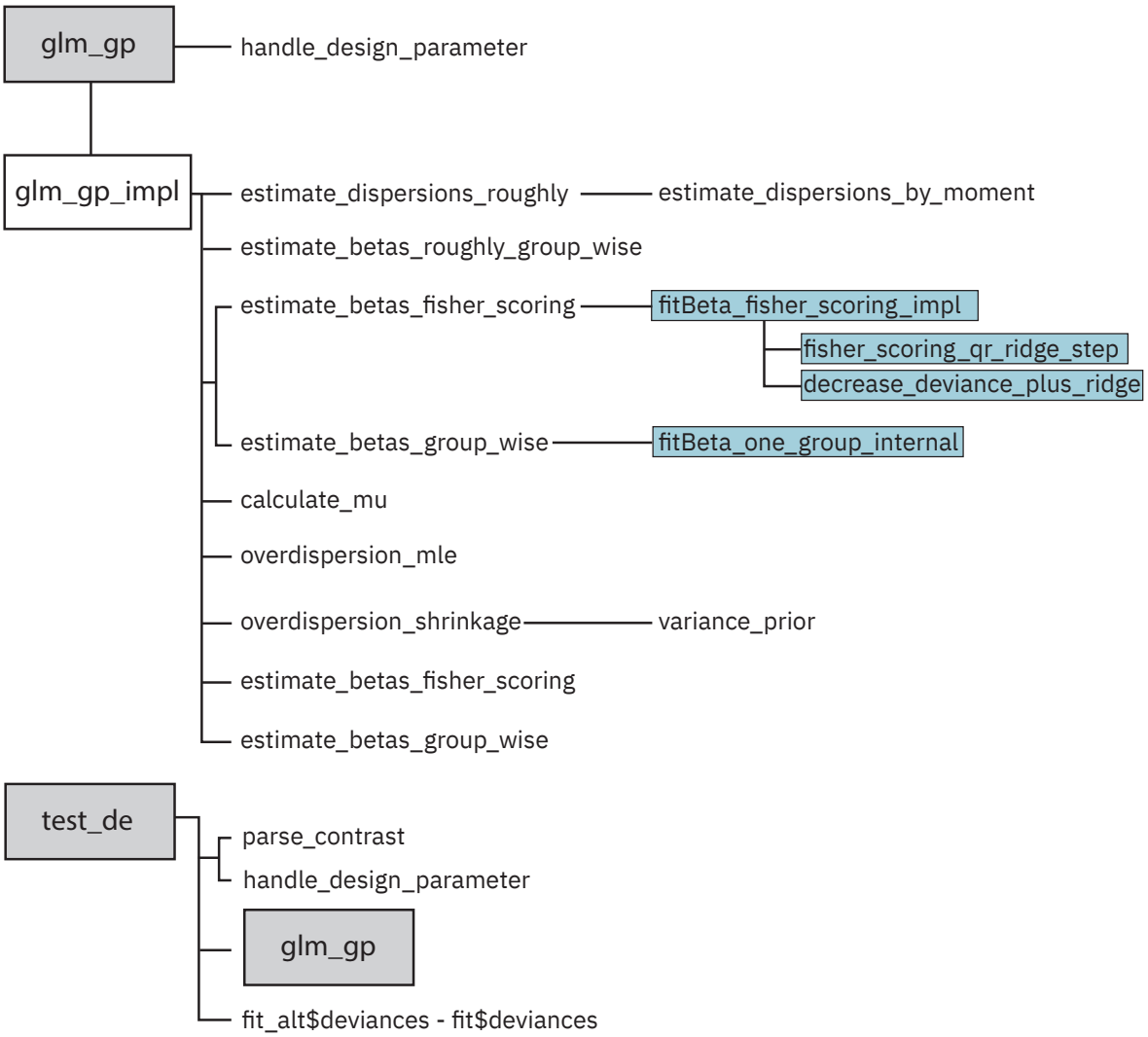


Figure 6 | Overview of the functions in the glmGamPoi package. Call diagram of the most important functions in glmGamPoi. The exported functions are highlighted with a grey background. The functions with the light blue background are implemented in C++ for efficiency.

In DESeq2, glmGamPoi can be used as an alternative inference engine by setting `fitType = "glmGamPoi"` when calling `DESeq`. The second version of sctransform automatically uses glmGamPoi to infer the overdispersion and model coefficients.

2.2.2 Contrasts

Contrasts are vectors multiplied with the coefficients β to produce a single log-fold change estimate between two conditions. To combine differential testing with contrasts with the underlying likelihood ratio tests comparing a full and a reduced model, I need to convert the contrast into a reduced model. I achieved this by projecting the full model matrix onto the null space of the contrast vector. This idea was adapted from the implementation of edgeR's `glmLRT` function.

In version 1.11 (released in April 2023), I added a new way to specify contrasts to glmGamPoi.

Although the theoretical interpretation of the contrast vector is straightforward, in practice, it can be challenging to specify the correct contrast vector because it depends on the ordering of the variables in the original design formula and whether an intercept was included. With my new syntax, the user directly specifies the levels of the condition that they want to compare, and I internally find the corresponding contrast vector. The following is a simple example using the new syntax

```
fit <- glm_gp(sce, design = ~ cell * stim)
# test contrast vector c(0, 0, 0, 1, 1, 0)
test_de(fit, contrast = cond(cell = "Tcells", stim = "ctrl") -
                        cond(cell = "Tcells", stim = "stim"))
```

The `cond` function only works within the contrast argument and expands to a vector as if I had called `model.matrix(~ cell * stim, data.frame(cell = "Tcells", stim = "ctrl"))` and `model.matrix` knew which other levels `cell` and `stim` could be.

I achieved this by storing the `stats::.getXlevels` from the `model.frame` output as an attribute of the design formula. I then restore the factor levels to each `cond` argument before calling `model.matrix`.

2.3 Comparisons with edgeR and DESeq2

2.3.1 Speed

I benchmarked the performance of glmGamPoi against edgeR and DESeq2 on four datasets with 4 000-68 000 cells. glmGamPoi was between 6× to 13× faster than edgeR and DESeq2 if the data was stored in memory (Figure 7). If the data was stored on disk, the runtime of glmGamPoi roughly doubled. If I skipped the overdispersion estimation, the runtime was roughly halved.

In Figure 8A,B, I benchmarked the scaling of glmGamPoi, DESeq2, and edgeR depending on the number of cells and genes. The relative runtime scaled linearly for all three methods with the number of genes. Although the theoretical inference in generalized linear models scales linearly with the number of cells, Figure 8B shows that glmGamPoi scales sub-linearly with the number of cells due to the count tabulation. This explains the good overall performance that I observed in Figure 7.

In Figure 8C, I analyzed the dependence of the overall computation time on the number of covariates. Theoretically, the inference scales cubically with the number of columns in the design matrix because of the matrix inverse in eq. (16). However, in the PBMC4k dataset, I found that the scaling is approximately linear for up to 20 covariates. I did not calculate the dependence of edgeR and DESeq2 on the number of covariates because the overdispersion inference dominated their runtime.

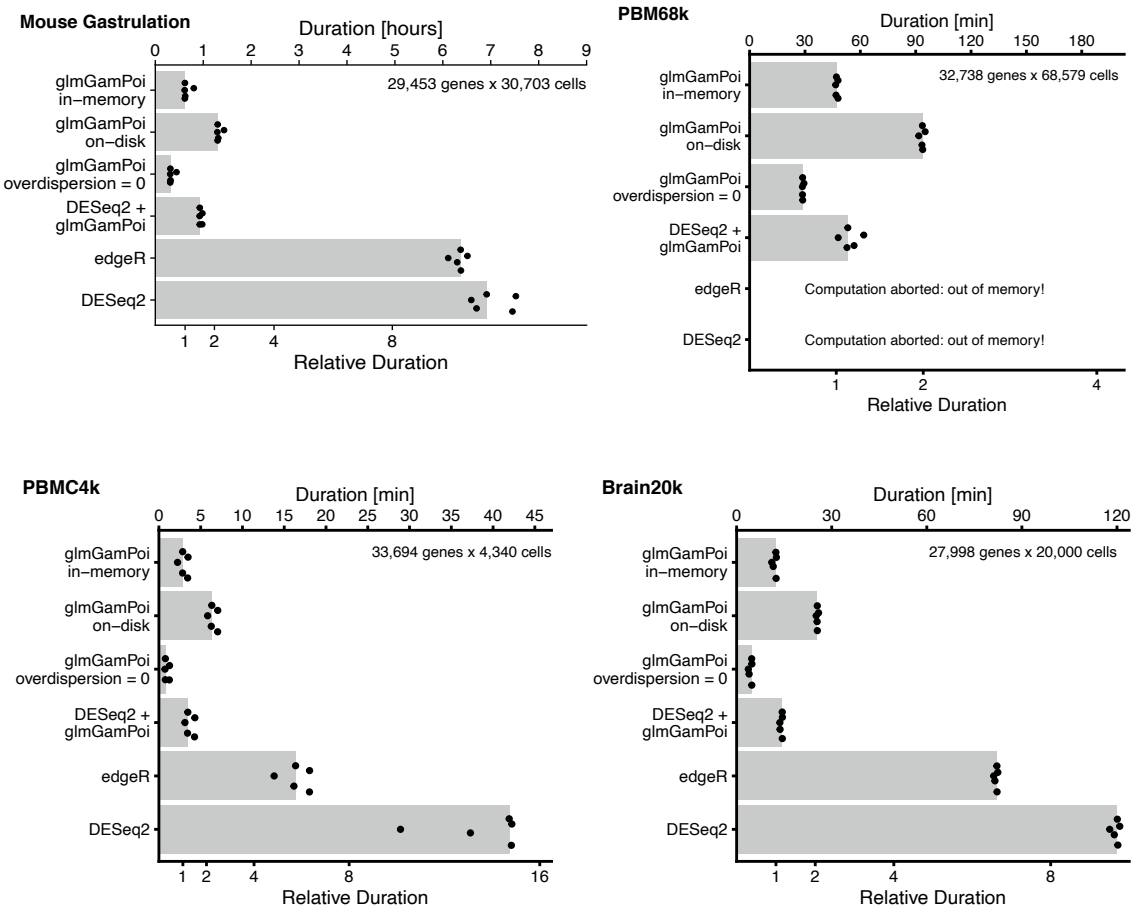


Figure 7 | Calculation comparison of glmGamPoi with edgeR and DESeq2. Barcharts comparing the computation times of glmGamPoi using data stored in memory or on disk against DESeq2 and edgeR. The timings were repeated five times without parallelization. The bottom x-axis scale is relative to the results of *glmGamPoi in-memory*.
The data for this figure was generated by me and adapted from Ahlmann-Eltze and Huber (2020).

2.3.2 Parameter estimates

glmGamPoi's overdispersion estimates have a higher likelihood than the one calculated by edgeR and DESEq2. In Figure 9, I compared the maximum likelihood overdispersion estimates of glmGamPoi against the maximum likelihood estimates of DESeq2 and edgeR on the PBMC4k dataset. All three methods optimized the same Cox-Reid adjusted Gamma-Poisson profile likelihood; thus, their results are directly comparable. For 14 675 and 1 958 genes glmGamPoi's estimates had a higher likelihood than DESeq2 and edgeR, respectively (purple points). Only for 400 and 430 genes DESeq2's and edgeR's overdispersion estimates were higher (orange points).

The coefficients inferred by glmGamPoi, DESeq2, and edgeR were broadly similar (Figure 10A compares the values of the *MouseA* coefficient in the Brain20k dataset). In particular, when using DESeq2 with the glmGamPoi inference engine, I get almost exactly the same coefficients

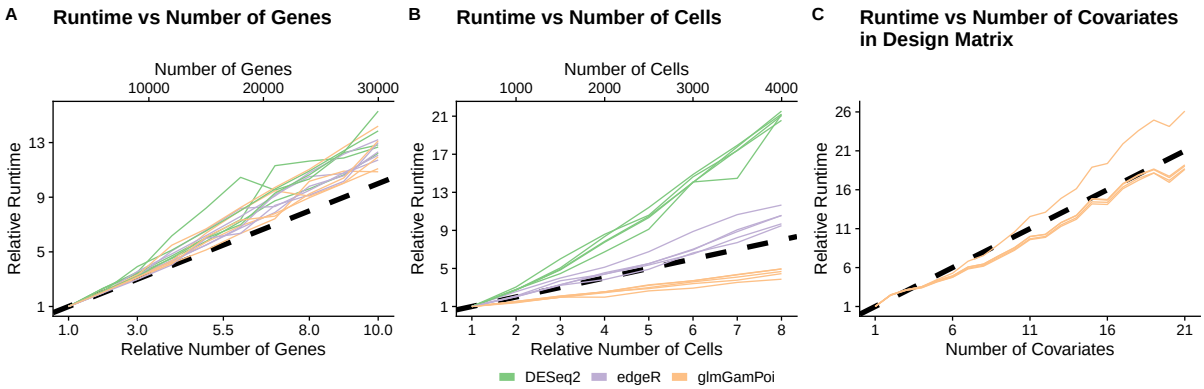


Figure 8 | Dependence of the runtime on the number of cells, genes, and covariates. Line charts showing the runtime of glmGamPoi, DESeq2, and edgeR on (A) the number of genes, (B) the number of cells, and (C) the number of covariates. In (C), I only show the results for glmGamPoi, because the runtime dependence of edgeR and DESeq2 on the number of covariates is obscured by the duration to estimate the overdispersion.
The data for this figure was generated by me and adapted from Ahlmann-Eltze and Huber (2020).

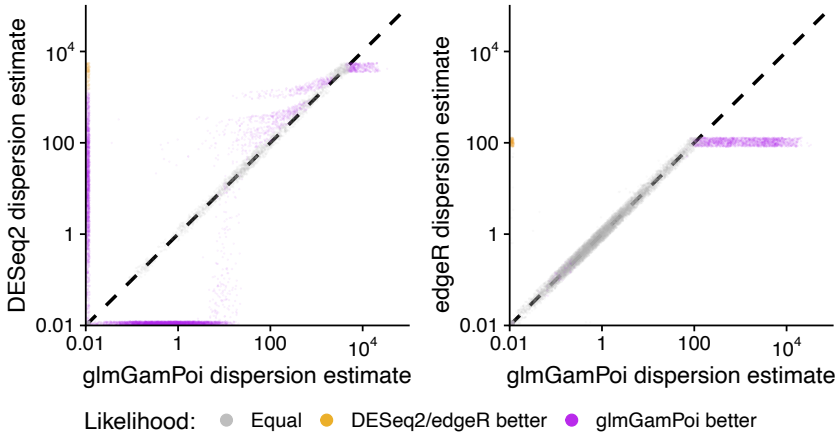


Figure 9 | Comparison of the overdispersion maximum likelihood estimates of glmGamPoi against DESeq2's and edgeR's. Scatter plot of the overdispersion estimates by glmGamPoi against DESeq2 and edgeR. Each point represents one gene, and the color indicates which method achieved the higher likelihood. For grey points, the likelihood was within a range of ±0.001. The dashed diagonal indicates the position if the two methods calculate the same overdispersion value.
The data for this figure was generated by me and adapted from Ahlmann-Eltze and Huber (2020).

as when I ran glmGamPoi as a standalone package. In Figure 10B, I compared the p-values on a log-log plot. The p-values inferred by edgeR and glmGamPoi were very similar across the whole range. In contrast, the DESeq2 infered smaller p-values for several genes. In Figure 11, I show the expression patterns of three genes for which glmGamPoi and DESeq2 inferred similar p-values and three genes for which the p-values diverged. The higher similarity of the results of edgeR and glmGamPoi is because they share the same quasi-likelihood ratio test framework.

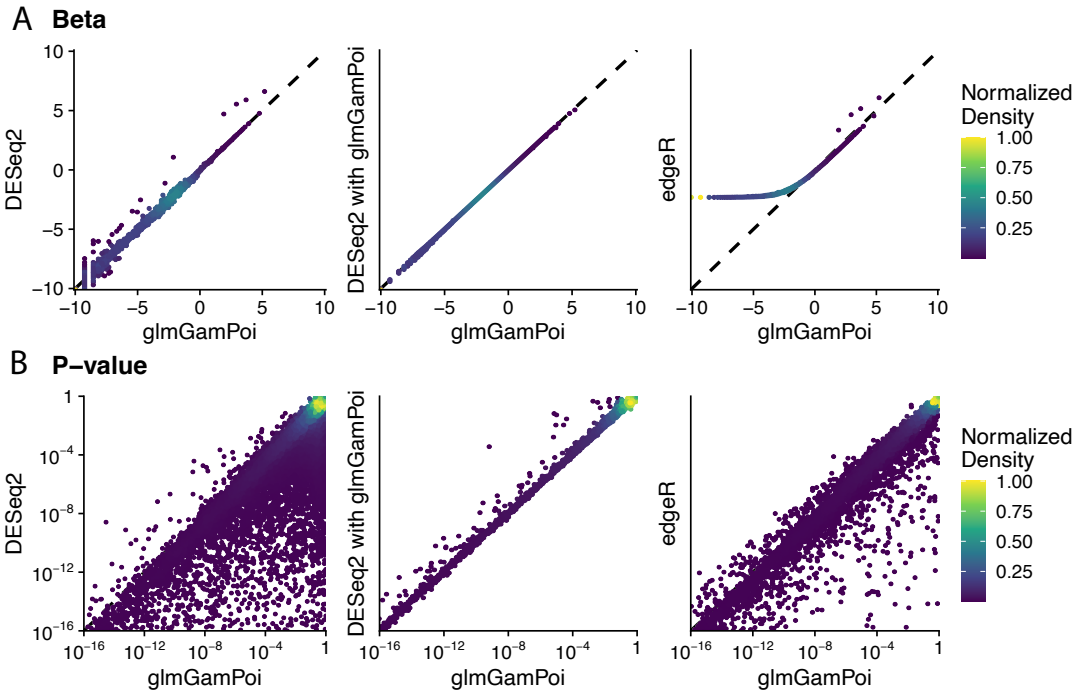


Figure 10 | Comparison of the coefficients and p-values estimated by glmGamPoi against DESeq2's and edgeR's. Scatter plots of the (A) coefficients and (B) p-values estimated by glmGamPoi against DESeq2 and edgeR on the Brain20k dataset. The color shows the normalized density of the points per plot. The dashed diagonal lines indicate the position if the two methods calculated the same overdispersion value.
The data for this figure was generated by me and adapted from Ahlmann-Eltze and Huber (2020).

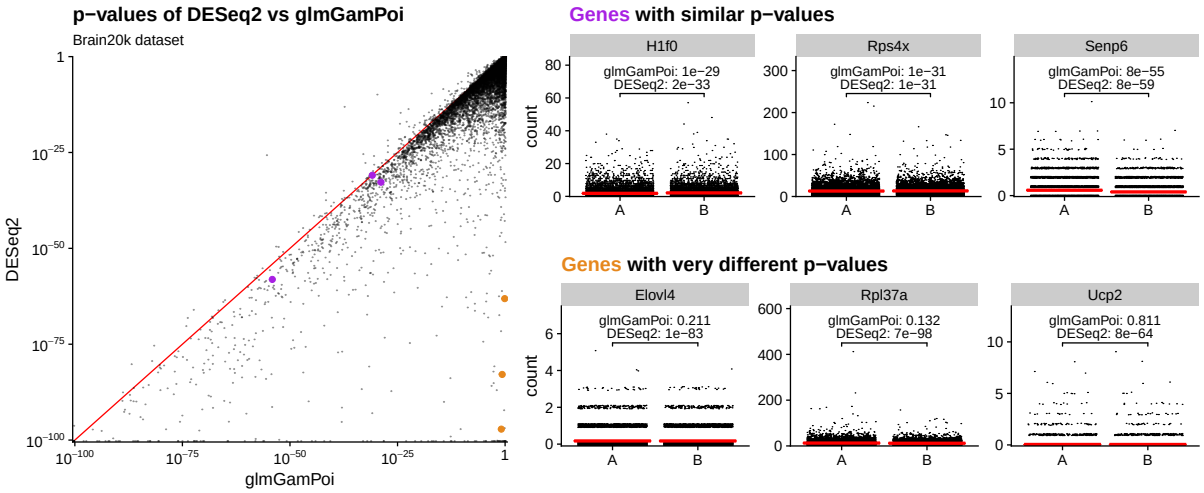


Figure 11 | Analysis of the gene expression values with divergent p-values between DESeq2 and glmGamPoi Comparison of three genes for which glmGamPoi and DESeq2 inferred similar p-values (purple) and three genes for which DESeq2 inferred much smaller p-values than glmGamPoi (orange).
The data for this figure was generated by me and adapted from Ahlmann-Eltze and Huber (2020).

2.4 Discussion

In this chapter, I presented my work on developing a new R package for fitting gamma-Poisson generalized linear models. My package glmGamPoi is faster than the two most popular alternatives DESeq2 and edgeR, which were developed for bulk RNA-sequencing data.

The main innovation of glmGamPoi is the tabulation based-optimization of the overdispersion estimation. Furthermore, glmGamPoi reliably infers the overdispersion estimates on single-cell data with many small counts. In all other aspects, it builds on the successful examples of edgeR and DESeq2. The differential testing framework reimplemented the quasi-likelihood framework by Lund et al. (2012) and pioneered by edgeR. The public functions and arguments of glmGamPoi are inspired by DESeq2, and I recently added some syntactic sugar to simplify the creation of pseudobulk samples and description of contrasts.

The package was well received by the single-cell community and has been integrated into `sctransform`, which is developed under the Seurat umbrella, one of the most successful computational single-cell projects to date. The package is downloaded between 2 000-3 000× per month, and the original publication (Ahlmann-Eltze and Huber, 2020) has been cited between 32 times according to Web of Science and 80 times according to Google Scholar.

The motivation to start this project at the beginning of my PhD was the expectation that count-based models with an accurate likelihood model could improve the inference across many tasks in single-cell analysis. For differential testing in bulk RNA-seq, count-based testing has been hugely successful (as demonstrated by the popularity of DESeq2 and edgeR); for single-cell data, count-based differential expression testing is popular; however, it can only be recommended in combination with pseudobulking (Crowell et al., 2020). I recently added a function to simplify the creation of pseudobulks from `SingleCellExperiment` objects; however, for differential expression testing on pseudobulks, the improved speed of glmGamPoi is of secondary importance. The familiarity of established differential expression testing pipelines using DESeq2 and edgeR makes them appealing alternatives.

Another application of glmGamPoi could be as a building block for an improved dimensionality reduction method similar to *GLM-PCA*by Townes (2019); Townes et al. (2019). In the next chapter, I will describe my comparison of different preprocessing and variance stabilization methods for single-cell data and show that they outperform count-based methods.

3 transformGamPoi

In this chapter, I will present the second project of my PhD: comparing a variety of preprocessing and transformation methods for single-cell data. I analyzed the conceptual properties of four transformation approaches and then measured their respective performance for 23 concrete implementations. In this chapter, I will begin by giving a backaground on the properties of single-cell RNA-seq counts and detail different approaches for variance stabilization (with an emphasis on the delta method). Then I will compare three conceptual differences and, lastly, describe the results of a large scale benchmarking effort to measure the empirical performance of the methods. I, together with Wolfgang Huber, published this project in Nature Methods as an Analysis paper called "Comparison of transformations for single-cell RNA-seq data" (Ahlmann-Eltze and Huber, 2023b).

3.1 Single-cell RNA-seq data properties

The primary output of a single-cell RNA-sequencing experiment is a table of genes × cells where each entry is a non-negative count of how often the mRNA of a gene was detected in a cell. The RNA-sequencing counts are heteroskedastic, which means that the counts vary more for highly expressed genes than for lowly expressed genes.

3.1.1 Why are counts heteroskedastic?

The dependence of the variance on the mean is typical for count data. It occurs whenever the counts are generated as a sum of a series of more or less independent Bernoulli events (whuber, 2014). If a count Y is the sum of the outcome of N independent Bernoulli trials with probability P_i where i is the index variable for the i-th trial, the expected value for Y is the sum

$$\mathbb{E}[Y] = \sum_{i}^{N} P_i. \tag{29}$$

Assuming that the Bernoulli variables P_i are independent means that the variance of Y is the sum of the individual variances ($\mathbb{V}[P_i] = P_i(1 - P_i)$)

$$\mathbb{V}[Y] = \mathbb{V}\left[\sum_{i}^{N} P_i\right] = \sum_{i}^{N} P_i(1 - P_i). \tag{30}$$

In binomial sampling, we assume that all probabilities are equal; accordingly, $\mathbb{E}[Y] = NP$ and $\mathbb{V}[Y] = NP(1 - P)$ which highlights that the mean and variance are directly proportional. The Poisson distribution can be understood as a limit of the binomial sampling when P is small and N is large. In that case, $1 - P$ becomes negligible and the variance is equal to the mean ($\mathbb{V}[Y] \approx NP = \mathbb{E}[Y]$).

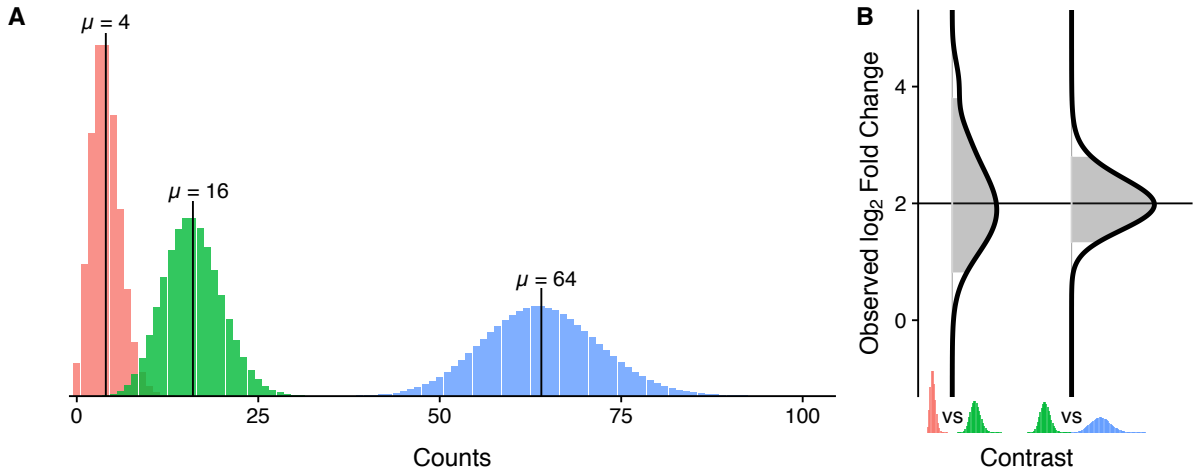


Figure 12 | Schematic of heteroskedastic data. (A) Probability mass function of three Poisson random variables with increasing mean and variance. (B) The log-fold changes between the red and the green compared to the log-fold changes between the green and the blue are the same, but the variance of the green-vs-blue case is smaller. The shaded area shows the range between the 5% and 95% quantile.
The data for this figure was generated by me and adapted from the first version of Ahlmann-Eltze and Huber (2022).

Figure 12A shows an example of three Poisson random variables with increasing mean. As predicted by theory, the spread of the distributions increases with the mean. However, it is important to remember that a higher average expression is beneficial for precise parameter estimates. Figure 12B shows the estimated log-fold change between the red and green compared to the estimated log-fold change between the green and blue distribution. Even though the individual variance is larger for the green and blue distributions, their observed log-fold change is less variable.

This apparently paradoxical effect can be explained by considering the coefficient of variation, that is, the standard deviation divided by the mean, of a Poisson random variable $Y \sim \text{Poisson}(\mu)$

$$c_v = \frac{\sqrt{\mathbb{V}\text{ar}[Y]}}{\mathbb{E}[Y]} = \frac{\sqrt{\mu}}{\mu} = \frac{1}{\sqrt{\mu}}. \tag{31}$$

The coefficient of variation of a Poisson random variable decreases with increasing mean.

If we now consider the variance of a log fold change (as in Figure 12B), we find

$$\mathbb{V}\text{ar}\left[\log \frac{Y_1}{Y_2}\right] = \mathbb{V}\text{ar}[\log Y_1] + \mathbb{V}\text{ar}[\log Y_2]. \tag{32}$$

We use the delta method (which we will explain in detail in section 3.2.2) to approximate the

variance of the log of a Poisson random variable

$$\begin{aligned} \mathbb{V}\text{ar}[g(Y)] &\approx (g'(\mathbb{E}[Y]))^2 \mathbb{V}\text{ar}[Y] \\ \mathbb{V}\text{ar}[\log(Y)] &\approx \frac{1}{\mathbb{E}[Y]^2} \mathbb{V}\text{ar}[Y] \end{aligned} \tag{33}$$

If we now plug equation (33) into equation (32), we find that

$$\begin{aligned} \mathbb{V}\text{ar}\left[\log \frac{Y_1}{Y_2}\right] &\approx \frac{\mathbb{V}\text{ar}[Y_1]}{\mathbb{E}[Y_1]^2} + \frac{\mathbb{V}\text{ar}[Y_2]}{\mathbb{E}[Y_2]^2} \\ &= c_{v1}^2 + c_{v2}^2, \end{aligned} \tag{34}$$

which demonstrates that the log fold change decreases with increasing means of two Poisson random variables.

3.1.2 Examples of single-cell data heteroskedasticity

Figure 13 shows a subset of the count table for the PBMC4k dataset generated using 10x single-cell technology. I represent the counts using a matrix $Y \in \mathbb{Z}_{\geq 0}^{G \times C}$, where G is the number of genes and C is the number of cells. Figure 13A and C demonstrate that the counts for most genes are very small (i.e., mostly zeros). Figure 14 shows the mean-variance relationship of 15 single-cell datasets on a log-log scale. Comparing the empirical distributions of the variance per gene with that expected for the Poisson (purple) and gamma-Poisson distribution (yellow) revealed that highly expressed genes have more variation than expected due to the Poisson distribution. For lowly expressed genes, the simple Poisson model relating mean and variance explains the observed pattern well.

3.2 Variance-stabilizing transformations

In Chapter 2, I presented generalized linear models as a method to analyze data with a non-constant mean-variance relationship. An alternative approach is to transform the values so that the mean-variance relationship is approximately constant. This is called variance stabilization, and the transformations to achieve approximately constant variance are called variance-stabilizing transformations.

3.2.1 Comparison of variance-stabilizing transformations and generalized linear models

Historically, variance-stabilizing transformations have been around since the pioneering work of Maurice Stevenson Bartlett (Bartlett, 1947, 1936), roughly 35 years longer than the concept of generalized linear models (Nelder and Wedderburn, 1972). The appeal of variance-stabilizing transformations comes from their simplicity: after applying a function to the data, one can use any method that assumes constant variance for optimal performance. In fact, fitting linear regression

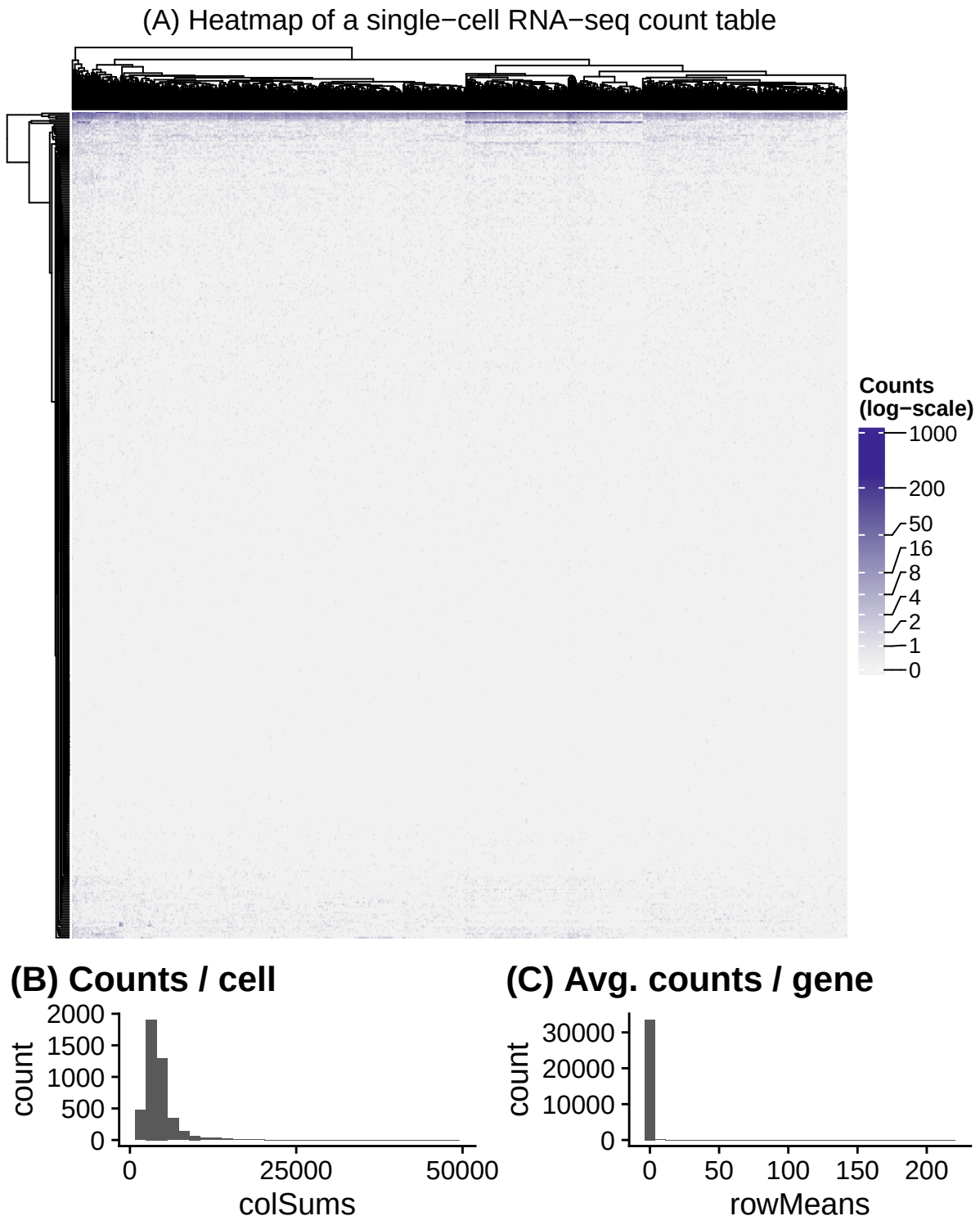


Figure 13 | Structure of a single-cell RNA-sequencing count table. (A) Heatmap of the counts from 1 000 cells and 500 genes of a PBMC dataset. The color scale is logarithmic. (B) Histogram of the total counts per cell for the full PBMC dataset. (C) Histogram of the average counts per gene for the full PBMC dataset.
The data for this figure was generated by me.

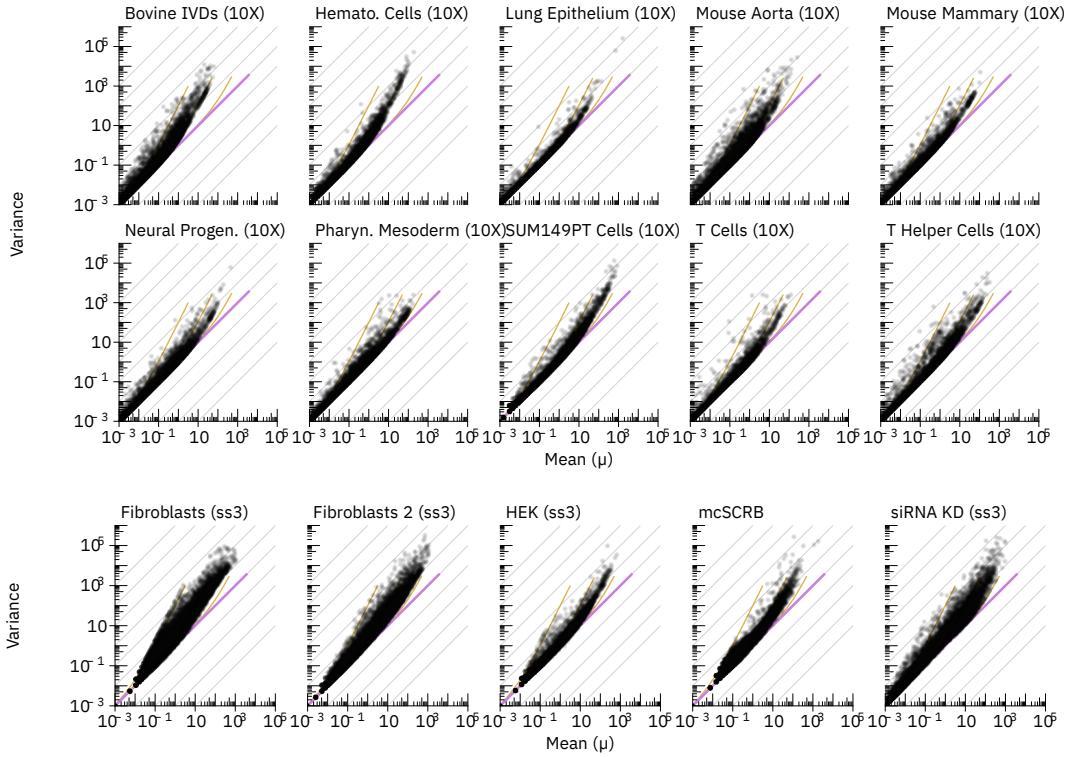


Figure 14 | Plot of the mean-variance relationship for 15 single-cell datasets Scatter plots of the empirical mean-variance relationship per gene for 15 single-cell datasets on a log-log scale. The purple diagonal shows the expected mean-variance relationship for a Poisson random variable. The yellow lines show the mean-variance relationships for gamma-Poisson random variables with different overdispersion values. We limited the cells to those that were within 50% quantile of total counts per cell to avoid heterogeneity due to the sequencing depth.
The data for this figure was generated by me and adapted from Ahlmann-Eltze and Huber (2022).

to data that was variance stabilized with g, approximates fitting a generalized linear model with variance function $V(\mu) = \frac{1}{g'(\mu)^2}$ and link function $g(\mu)$ (Dunn and Smyth, 2018, sec. 5.8).

If the variance-stabilizing transformation is $g(x) = \sqrt{x}$, the variance function is

$$\begin{aligned} V(\mu) &= \frac{1}{g'(\mu)^2} \\ &= \frac{1}{(1/\sqrt{\mu})^2} \\ &= \mu. \end{aligned} \tag{35}$$

We recognize this as the mean-variance relationship of a Poisson random variable; however, the canonical link function[4] of a Poisson generalized linear model is $g(x) = \log(x)$ and not $g(x) = \sqrt{x}$ as implied by the variance-stabilizing transformation.

Generalized linear models have two advantages: First, they decouple the choice of mean-

[4]A link function is considered canonical if $\eta \equiv \theta$

variance relationship (corresponding to a specific choice of exponential dispersion model) and link function (Dunn and Smyth, 2018). Second, generalized linear models allow the direct interpretation of the coefficients (Dunn and Smyth, 2018). For a log-link function

$$\begin{aligned} \log \mu = \eta &= \beta_0 + \beta_1 x \\ 3.2.2 \mu = \exp \eta &= \exp(\beta_0) \exp(\beta_1)^x. \end{aligned} \tag{36}$$

However, a linear model

$$\mathbb{E}[\log y] = \beta_0 + \beta_1 x \tag{37}$$

only approximately models this relationship because

$$\mathbb{E}[\log y] \approx \log \mathbb{E}[y] = \log \mu. \tag{38}$$

For the analysis of RNA-sequencing data, both generalized linear models and fitting linear models after variance-stabilizing transformation are popular. The former is represented by DESeq2 and edgeR (Love et al., 2014; Robinson et al., 2009). The latter approach is implemented by limma and limma-voom (Smyth, 2004; Law et al., 2014).

3.2.2 Construction of variance-stabilizing transformations with the Delta method

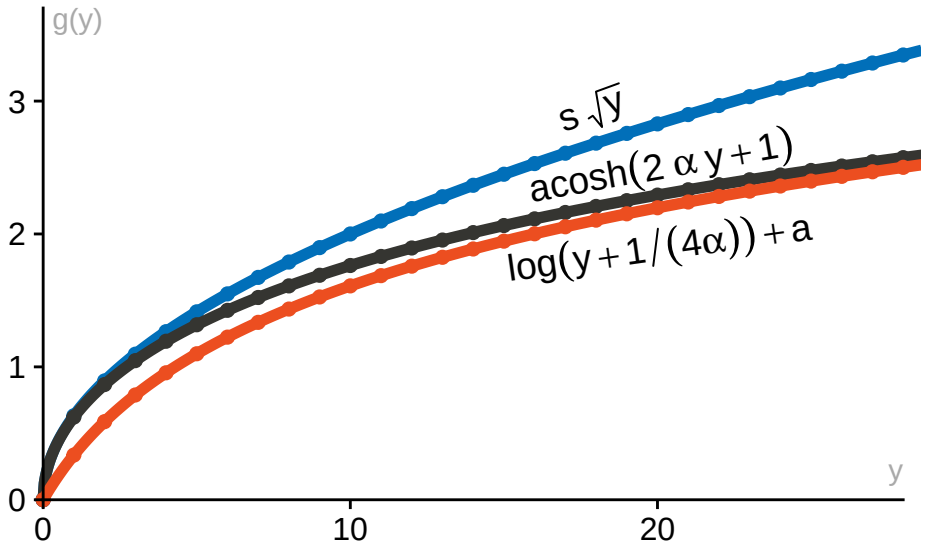


Figure 15 | Graph of three variance-stabilizing transformations The data for this figure was generated by me and adapted from the first version of Ahlmann-Eltze and Huber (2022).

For an observed mean-variance relationship, we can construct a variance-stabilizing transformation using the methods described in Bartlett (1947). Figure 15 shows three variance-stabilizing transformations.

The construction uses a trick called the delta method (Dorfman, 1938). The delta method is a

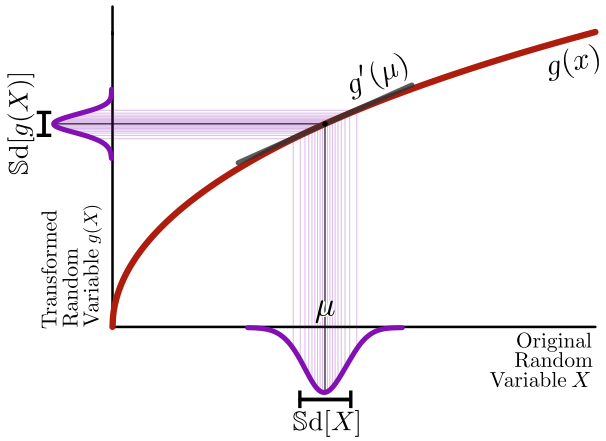


Figure 16 | Illustration of the delta method. The data for this figure was generated by me and adapted from Ahlmann-Eltze and Huber (2022).

way to find the standard deviation of a random variable after applying a transformation.

The standard deviation of a transformed random variable can be approximated by multiplying the original standard deviation by the derivative of the transformation at the mean

$$\mathbb{S}\mathrm{d}[g(Y)] = |g'(\mu)|\mathbb{S}\mathrm{d}[Y]. \tag{39}$$

The derivative at the mean $g'(\mu)$ is a linear approximation of the original transformation g (Fig. 16). And transforming a random variable with a linear function scales the standard deviation linearly.

If we have many random variables $Y_1, Y_2, \ldots$ with a known mean-variance relationship $v(\mu)$, we can derive a variance-stabilizing transformation. We posit that

$$\begin{aligned} \mathbb{V}\mathrm{ar}[g(Y)] &= \text{const.} \\ \mathbb{S}\mathrm{d}[g(Y)] &= \text{const.}. \end{aligned} \tag{40}$$

Now, when we plug eq. (40) into eq. (39)

$$\begin{aligned} \text{const.} &= |g'(\mu)|\mathbb{S}\mathrm{d}[Y] \\ &= |g'(\mu)|\sqrt{v(\mu)}. \end{aligned} \tag{41}$$

Without loss of generality, we will use a constant value of 1 and rewrite the equation as a function of g'

$$g'(\mu) = \frac{1}{\sqrt{v(\mu)}}. \tag{42}$$

If we assume a particular mean-variance relationship, we can solve eq. (42) using integration.

For example, for the mean-variance relationship of a Poisson random variable $v(\mu) = \mu$, we find

$$\begin{aligned} g(\mu) &= \int \frac{1}{\sqrt{v(\mu)}} \\ &= \int \frac{1}{\sqrt{\mu}} \\ &= 2\sqrt{\mu}. \end{aligned} \tag{43}$$

For the mean-variance relationship of a gamma-Poisson random variable $v(\mu) = \mu + \alpha\mu^2$, we find

$$\begin{aligned} g(\mu) &= \int \frac{1}{\sqrt{v(\mu)}} \\ &= \int \frac{1}{\sqrt{\mu + \alpha\mu^2}} \\ &= \frac{2}{\sqrt{\alpha}} \operatorname{asinh}(\sqrt{\alpha\mu}) \\ &= \frac{1}{\sqrt{\alpha}} \operatorname{acosh}(2\alpha\mu + 1). \end{aligned} \tag{44}$$

The last two expressions are identical. Bartlett (1947) used the name asinh transformation, but I will call it the acosh transformation because the expression seems slightly simpler.

Another popular transformation for count data is the shifted logarithm

$$\log(x + c) \tag{45}$$

where c is called a pseudo-count. This transformation is usually not motivated by the delta method, but it is a variance-stabilizing transformation for the mean-variance relation

$$v(\mu) = \mu^2. \tag{46}$$

This is not quite the mean variance relation of a gamma-Poisson random variable ($v(\mu) = \mu+\alpha\mu^2$), but eq. (45) can be used as an approximation for that relation. The inverse hyperbolic cosine (acosh) transformation (eq. (44)) can be rewritten in terms of a logarithm

$$\begin{aligned} g(x) &= \frac{1}{\sqrt{\alpha}} \operatorname{acosh}(2\alpha\mu + 1) \\ &= \frac{1}{\sqrt{\alpha}} \log\left(2\alpha x + \sqrt{(2\alpha x + 1)^2 - 1} + 1\right). \end{aligned} \tag{47}$$

I will find the parameters a, b, and c in

$$h(x) = a + b \log(x + c) \tag{48}$$

such that $h(x) \approx g(x)$. The middle summand in eq. (47) quickly converges to $2\alpha x$ for large x

because of the squaring

$$\lim_{x\to\infty} \frac{\sqrt{(2\alpha x+1)^2-1}}{2\alpha x} = 1. \tag{49}$$

Accordingly, I can rewrite

$$\begin{aligned} g(x) &\approx \frac{1}{\sqrt{\alpha}} \log\left(4\alpha x+1\right) \\ &= \frac{1}{\sqrt{\alpha}} \log\left(x+\frac{1}{4\alpha}\right) + \frac{\log(4\alpha)}{\sqrt{\alpha}} \end{aligned} \tag{50}$$

and find $a = \frac{\log(4\alpha)}{\sqrt{\alpha}}$, $b = \frac{1}{\sqrt{\alpha}}$, and $c = \frac{1}{4\alpha}$. For the variance-stabilizing properties of eq. (45) the offset a and the scaling with b do not matter. Importantly, the relation $c = \frac{1}{4\alpha}$ provides guidance for choosing the pseudo-count that best approximates the variance-stabilizing transformation we derived using the delta method in eq. (44).

3.3 Accounting for varying sequencing depth

In Figure 13B, I gave an example of the varying number of counts per cell in single-cell data. These can have different explanations: some variation is due to stochasticity, but it can also represent biological signal. For example, few observed mRNAs per cell can indicate a dying cell or an empty droplet where only ambient RNA is detected. A large number of UMIs per cell can occur if more than one cell is captured in a droplet (called doublets or multiplets). In addition, transcriptionally active cells contain more mRNA as do in general larger cells.

These variations are often considered nuisance variables and accounted for before fitting the model. One common way to account for the variation in the sequencing depth is to divide the counts by a size factor to bring all to a common scale. The size factor is a scalar that is larger for cells with more UMIs and smaller for cells with fewer UMIs. In addition, it is typically centered around 1 so that dividing the counts by the size factor does not change the scale of the values.

The simplest way to calculate the size factor is to calculate the total counts per cell and rescale them so that they have a mean of 1

$$s_c = \sum_g Y_{gc} \Big/ \left(\frac{1}{C}\sum_{gc'} Y_{gc'}\right). \tag{51}$$

In bulk-RNA sequencing, a more sophisticated method developed by Anders and Huber (2010) was popularized by DESeq

$$s_c = \text{median}_g \frac{Y_{gc}}{\exp\left(\frac{1}{C}\sum_{c'} \log Y_{gc'}\right)}. \tag{52}$$

It is more robust against outliers, but it, unfortunately, does not work well for single-cell data

because of the prevalence of zeros which pose a problem for the denominator, which calculates the mean of the log counts.

3.3.1 Scaling counts by a size factor introduces problems for variance-stabilizing transformation based on the delta method

The content of this section arose from discussions with Simon Anders and was published in the supplementary material of Ahlmann-Eltze and Huber (2023b).

The counts Y are adjusted for varying sequencing depths by dividing them by the size factor before applying a variance-stabilizing transformation $g(Y/s)$. The procedure is straightforward but impacts downstream analysis. Intuitively, the problem is that a count y_1 from a cell with a larger size factor and a count y_2 from a cell with a smaller size factor ($y_1 > y_2$) will be scaled to the same size. A single variance-stabilizing transformation cannot correct for the additional variation in y_1 compared with y_2.

For a more technical analysis of the problem, I will first explain the variance decomposition of a gamma-Poisson random variable Y. We assume, given the true mean Q, that Y is conditionally Poisson distributed

$$Y|Q \sim \text{Poisson}(Q). \tag{53}$$

If we assume that Q is a gamma distributed random variable around mean μ, we find that Y is marginally gamma-Poisson distributed

$$\begin{aligned} Q &\sim \text{gamma}(\mu, \alpha) \\ Y &\sim \text{gamma-Poisson}(\mu, \alpha). \end{aligned} \tag{54}$$

Here, the Poisson variation represents technical noise inherent in the sampling procedure (see Section 3.1.1). The gamma-distributed Q captures additional noise. Using the law of total variation, we can re-derive the mean-variance relationship of a gamma-Poisson random variable

$$\begin{aligned} \mathbb{V}\text{ar}[Y] &= \mathbb{E}[\mathbb{V}\text{ar}[Y|Q]] + \mathbb{V}\text{ar}[\mathbb{E}[Y|Q]] \\ &= \mu + \alpha\mu^2, \end{aligned} \tag{55}$$

where the variance of the conditional Poisson random variable $(Y|Q)$ is μ and the variance of Q is $\alpha\mu^2$.

I will use this framework to study the effect of rescaling the counts with a size factor s. First, I will assume that s is known and include it in the definition of the Poisson noise

$$Y|Q, s \sim \text{Poisson}(sQ). \tag{56}$$

Accordingly, the variance of Y is now

$$\begin{aligned}\mathbb{V}\mathrm{ar}[Y] &= \mathbb{E}[\mathbb{V}\mathrm{ar}[Y|Q,s]] + \mathbb{V}\mathrm{ar}[\mathbb{E}[Y|Q,s]] \\ &= s\mu + \alpha s^2\mu^2 \\ &= \mu' + \alpha\mu'^2,\end{aligned} \tag{57}$$

where $\mu' = s\mu$.

Now, if I divide the count Y by s

$$Z = Y/s \tag{58}$$

the variance of Z is

$$\begin{aligned}\mathbb{V}\mathrm{ar}[Z] &= \frac{1}{s^2}\mathbb{V}\mathrm{ar}[Y] \\ &= \frac{1}{s^2}\left(s\mu + \alpha s^2\mu^2\right) \\ &= \frac{1}{s}\mu + \alpha\mu^2.\end{aligned} \tag{59}$$

The dependence of the eq. (59) on s and the difference to eq. (57) causes problems, as I will show in Section 3.5.1.

3.3.2 Other scaling factors

In the previous section, I introduced scaling the counts by dividing them by size factors which are normalized around 1 so that the scale of the counts is unchanged (eq. (51)). Sometimes the denominator $L = \frac{1}{C}\sum_{gc} y_{gc}$ is changed to a fixed value such as $L = 10\,000$ (in Seurat). Others have even suggested to use $L = 10^6$ calling the results *counts per million* (CPM)(Luecken and Theis, 2019). Counts per ten-thousand or counts per million are used in combination with the simple $\log(y/L+1)$ transformation. Even though the choice of L may appear arbitrary, it impacts the variance-stabilizing quality of the transformation. Changing the scaling factor L is equivalent to changing the pseudo-count. If we assume a $\frac{1}{C}\sum_{gc} y_{gc} \approx 5000$ for a typical droplet-based single-cell dataset, we find

$$\begin{aligned}\log\left(\frac{y}{(\sum_c y_{:c})/10^6} + 1\right) &= \log\left(200\left(\frac{y}{(\sum_c y_{:c})/5000} + \frac{1}{200}\right)\right) \\ &= \log\left(\frac{y}{(\sum_c y_{:c})/5000} + 0.005\right) + \log(200) \\ &\propto \log\left(y/s_c + 0.005\right).\end{aligned} \tag{60}$$

Accordingly, using the relation established in eq. (50), using $L = 10^6$ is the same as using a pseudo-count of $y_0 = 0.005$ or positing an overdispersion of $\alpha = \frac{1}{4y_0} = 50$ which is roughly 100× larger than empirically observed values. Seurat's normalization factor of $L = 10\,000$ is the same as an overdispersion of $\alpha = 0.5$.

3.4 Other approaches for transforming single-cell counts

Besides the delta method based-variance-stabilizing transformations that I introduced in Section 3.2, there are two other approaches to handle the heteroskedastic counts in single-cell RNA-seq data: model residuals and inferred latent expression activity values. Lastly, another approach is to skip the normalization altogether and apply statistical models that account for the data idiosyncrasies.

3.4.1 Model residuals

In Section 3.2.1, I discussed the relationship between generalized linear models and variance-stabilizing transformations based on the delta method. Instead of identifying the function that produces approximately variance stabilized values, Hafemeister and Satija (2019) suggested using Pearson residuals from a gamma-Poisson generalized linear model fit per gene as transformed values.

Pearson residuals of a gamma-Poisson are defined as

$$\begin{aligned} r_{gc} &= \frac{y_{gc} - \hat{\mu}_{gc}}{\sqrt{\hat{v}(\hat{\mu}_{gc})}} \\ &= \frac{y_{gc} - \hat{\mu}_{gc}}{\sqrt{\hat{\mu}_{gc} + \hat{\alpha}_g \hat{\mu}_{gc}^2}}. \end{aligned} \tag{61}$$

The idea is to take the count, center it, and equalize the variance by dividing it by the expected standard deviation for the inferred mean.

The mean and overdispersion for each gene is estimated using

$$\begin{aligned} Y_{gc} &\sim \text{gamma-Poisson}(\mu_g c, \alpha_g) \\ \log(\mu_{gc}) &= \beta_{g,\text{intercept}} + \beta_{g,\text{slope}} \log(s_c). \end{aligned} \tag{62}$$

The size factor s_c is inferred for each cell, and the intercept $\beta_{g,\text{intercept}}$ and the slope $\beta_{g,\text{slope}}$ are fitted for each gene g. Hafemeister and Satija (2019) implemented their model in an R package called *sctransform*. In version 2 of the model, the package uses *glmGamPoi* to fit the model.

One challenge for the Pearson residuals (eq. (61)) is that the denominator can become very small for genes that are zero in most cells. Hafemeister and Satija (2019) suggested to clip the range of r to $\left[-\sqrt{C}, +\sqrt{C}\right]$.

Lause et al. (2021) argued that the model (62) is too flexible and suggested fixing $\beta_{g,\text{slope}} = 1$. In response to the criticism, in version 2 of sctransform fixing $\beta_{g,\text{slope}} = 1$ has become the default (Choudhary and Satija, 2022).

Furthermore, Lause et al. (2021) suggested not to estimate the overdispersion per gene, but to use a fixed value such as $\alpha = 0.01$. The question of how to choose the overdispersion comes down to the level of variation the analyst considers uninteresting.

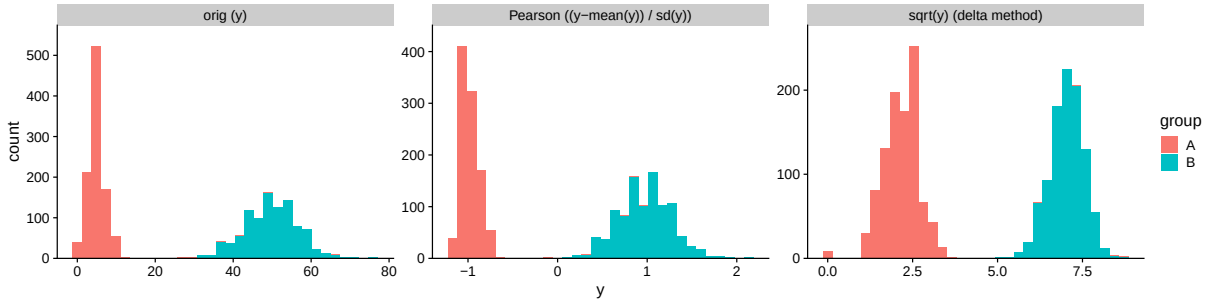


Figure 17 | Demonstration that Pearson residuals are a linear transformation. The data y are generated using R's `rpois` function.
The data for this figure was generated by me and adapted from the peer review discussion of Ahlmann-Eltze and Huber (2023b).

If the analyst considers all variation larger than the noise due to the sampling process from the Poisson distribution interesting, setting $\alpha = 0$ or $\alpha = 0.01$ as suggested by Lause et al. (2021) is the way to go. Alternatively, if the focus is supposed to be on genes that show more variation than most other genes one can use a robust estimation procedure such as the one implemented in sctransform. Lastly, suppose the analyst wants to minimize the influence of individual genes and focus on patterns that are consistent across genes. In that case, they should use each gene's maximum likelihood overdispersion estimate.

Lause et al. (2021) combined the insight that they can fix $\beta_{g,\text{slope}} = 1$ and $\alpha = 0.01$ while retaining most of the model's expressivity, to construct a fast way to approximate the Pearson residuals, which they call *analytic Pearson residuals*. They estimate μ_{gc} as

$$\hat{\mu}_{gc} = \frac{\left(\sum_{g'} y_{g'c}\right)\left(\sum_{c'} y_{gc'}\right)}{\sum_{g'c'} y_{g'c'}}. \tag{63}$$

The formula is derived from the logarithm of the probability density function of a Poisson random variable with size factors. They then go on and use $\hat{\mu}_{gc}$ from eq. (63) to normalize the counts with eq. (61).

One limitation of eq. (61) is that it is a linear transformation of the counts (Figure 17). I developed a non-linear residual-based transformation adapting an idea by Dunn and Smyth (1996) called *randomized quantile residuals*. Figure 18 shows schematically for one hypothetical gene how randomized quantile residuals are constructed. In the first step, I fit a gamma-Poisson model to the counts of that gene. Then, I match the quantiles of the cumulative distribution function (CDF) against the CDF of a standard normal distribution. The mapping is not unique because the CDF of a gamma-Poisson model is discrete. The idea of randomized quantile residuals to resolve this is by randomly drawing a value from the matching standard normal CDF before mapping back to the original scale. The two colored bars demonstrate this mapping for counts $y = 2$ and $y = 21$.

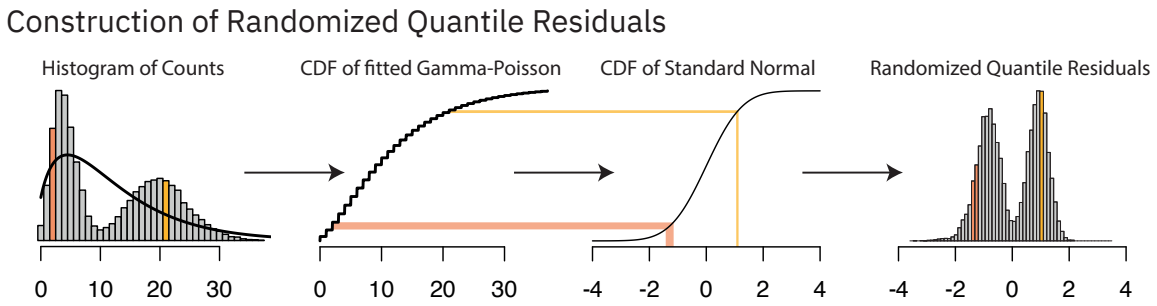


Figure 18 | Schematic representation of how randomized quantile residuals are constructed. The data for this figure was generated by me and adapted from Ahlmann-Eltze and Huber (2022).

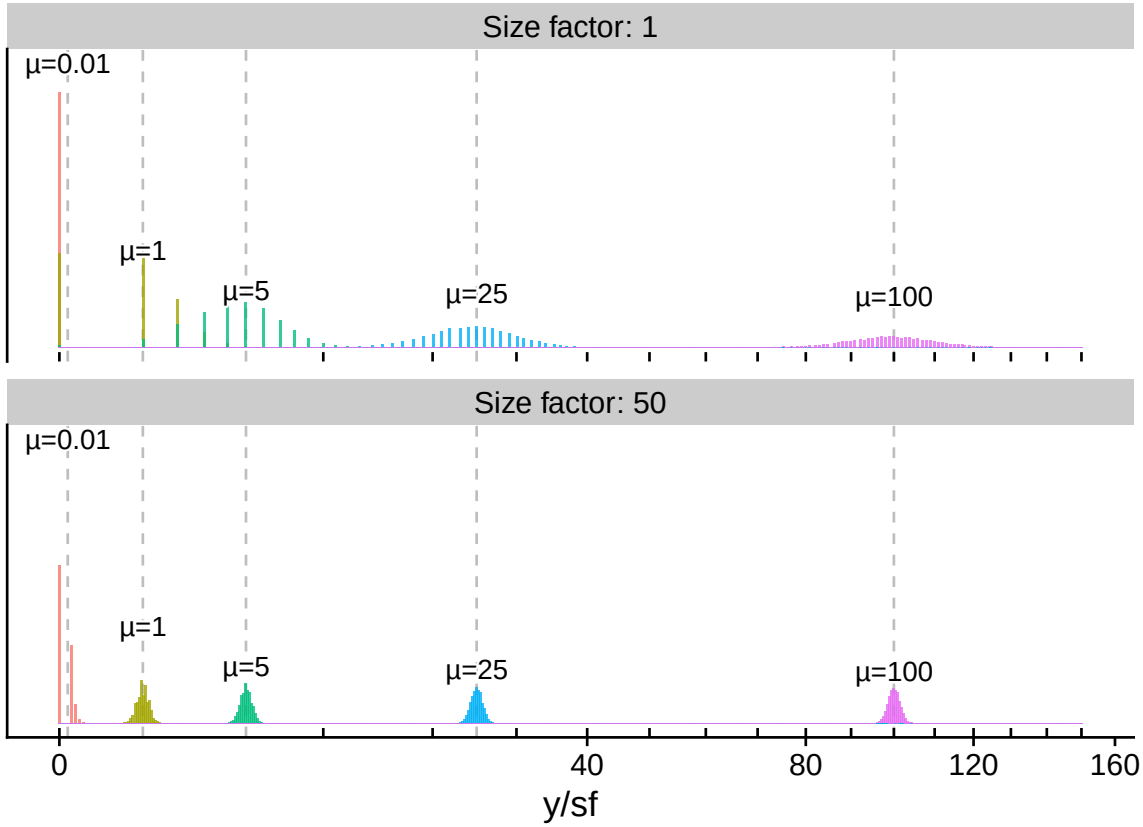


Figure 19 | Schematic representation of how size factors influence the precision of the inferred values. The figure shows histograms for random variables from $Y \sim \text{Poisson}(\mu s)$ for five mean values and two size factors. The values are plotted on a square-root transformed scale for better visualization.

The last panel of Figure 18 shows how the non-linear nature of the randomized quantile residuals equalizes the width of the two peaks.

3.4.2 Latent expression value estimation

A fundamental assumption for the analysis of single-cell count data is that they provide a noise-corrupted image of the true biological processes that are happening in each cell (Figure 19). For example, in generalized linear models, we infer the means of each value conditional on the observed covariates. The idea that we can infer the latent state has been extended from tools like DESeq2 and edgeR to single-cell data transformations.

One prominent example is called *Sanity* which was developed by Breda et al. (2021). Sanity takes as input a matrix of counts and infers a posterior distribution of the latent expression activities which they represent using a matrix of means and standard deviations. The model is fully Bayesian and fits a model that is similar to a variational mean-field approximation for a log-normal Poisson mixture model.

I will call the full model *Sanity Distance* because Breda et al. (2021) provide an additional tool to calculate the cell-by-cell distance matrix from the mean and standard deviation matrices. I also consider ignoring the inferred uncertainty and calling the corresponding procedure *Sanity MAP* (short for *maximum a posteriori*). In their analysis, Breda et al. (2021) find that accounting for the uncertainty improves distance inference. The advantage of Sanity MAP is, though, that it is directly compatible with any existing algorithm that assumes a single matrix of values (for example, approximative k-nearest neighbor inference).

We consider two more latent expression-based transformations: *Dino* and *Normalisr*. Dino fits mixtures of gamma-Poisson mixtures to the data and returns random samples from the posterior (Brown et al., 2021). Normalisr uses a binomial model and returns the minimum squared estimate for each count (Wang, 2021). It was not originally designed as a preprocessing method, but I decided to test it for this purpose as it infers logarithmic latent gene expression values.

3.4.3 Post-transformation processing

Some analysts apply additional modifications after applying a delta method or residual-based transformation. Here, I will consider the z transformation, highly variable gene selection, and their concatenation.

The motivation for z transforming the variance stabilized counts

$$\text{z-trans}(g(y_{gc}/s)) = \frac{g(y_{gc}/s)}{\sqrt{\text{variance}(g(y_{g:}/s))}} \tag{64}$$

is that the z transformation equalizes the variance for each gene. This means that each gene will contribute the same amount to distance calculation.

Highly variable gene (HVG) selection has the opposite motivation. To avoid corrupting the signal by including genes for which the counts are all almost zero and thus have little variance, genes with small variation are excluded from the analysis. In addition, highly variable gene

selection can significantly speed up the processing because it can reduce the size of the data from ten thousands of genes to a few hundred while retaining most of the variation.

Sometimes the data is first subset to the highly variable genes and subsequently transformed with a z transformation.

3.4.4 *No-transformation* analysis of single-cell data

I have presented three approaches that take the count matrix and transform it. The transformed values are not a goal in and of themselves but are an intermediate step for further downstream analysis. There are many possible downstream applications, but I will focus in this comparison on finding the local and global structure of the data. The preferred approach is to reduce the dimensionality of the data using principal component analysis (PCA) and find the k-nearest neighbors (k-NN) of each cell.

The idea of PCA is to take the matrix of transformed values $g(Y)$ and approximate it using a lower-dimensional embedding

$$\underset{R,Z,\beta}{\arg\min} ||g(Y) - (RZ + \gamma_{\text{offset}})||_2^2 \tag{65}$$

where R is an orthonormal matrix of size $G \times P$ and Z is a low-dimensional embedding matrix of size $P \times C$. Here, P is the dimension that is used for the PCA, and γ_{offset} is a vector of length G centering the values for each gene.

Townes (2019) suggested to use the ideas of generalized linear models instead of optimizing eq. (65), minimize the deviance of a Poisson (or gamma-Poisson) model

$$\begin{aligned} Y_{gc} &\sim \text{Poisson}(\exp(M_{gc})s_c) \\ M &= R\,Z + \gamma_{\text{offset}} \end{aligned} \tag{66}$$

where M is an intermediate matrix of size $G \times C$ and s_c is the size factors for each cell.

Townes (2019) and Townes et al. (2019) developed an inference algorithm for the parameters of model (66) and implemented it as an R package called *glmPCA*. Recently, Agostinis et al. (2022) proposed an alternative inference algorithm for model (66) which they optimized for fast parallel processing called *NewWave*.

Both methods produce a low-dimensional embedding matrix Z directly compatible with 2D visualization tools like UMAP and k nearest neighbor graph inference approaches. Thus, I decided to include both methods in my comparison even though they are not technically transformations. I use the umbrella term *count models* for them because they work directly on the counts.

3.4.5 Negative controls

Applying a transformation is not technically necessary. Instead of running, for example, PCA on the transformed counts, one can also run PCA on the raw counts. According to statistical theory, the least squares optimization on heteroskedastic data is unbiased, yet less precise (Wooldridge, 2015).

To demonstrate that the transformations are improving our ability to analyze single-cell RNA-seq data, I included two negative controls. First, I conducted all analyses on the raw counts and second on the raw counts divided by the size factors.

3.5 Conceptual comparison of the transformation approaches

I have introduced four approaches to preprocess and transform single-cell counts: *delta method*-based variance-stabilizing transformation, *generalized linear model residual*-derived transformations, *latent expression* activity inference approaches, and the *count models* which produce a lower dimensional space without transforming. In this section, I will compare the conceptual difference between the approaches.

3.5.1 Confounding effect of size factors on PCA

The sequencing depth per cell varies and we typically want to adjust for this effect, i.e., reduce its impact on downstream analyses. As mentioned in the previous sections, applying variance-stabilizing transformations based on the delta method on counts divided by size factors causes problems (section 3.3.1).

To demonstrate the differences varying size factors have across transformations, I took a single-cell dataset of an ostensibly homogeneous cell line population (Svensson et al., 2017), applied each transformation outlined in the introduction and plotted the location of each cell according to the first two principal components (Figure 20). Coloring the cells by size factor revealed a gradient along the first principal component for the delta method-based variance-stabilizing transformations.

I quantified this effect pattern by calculating the canonical correlation coefficient between the size factors and the first ten principal components (Hotelling, 1936). In Figure 21, I show the correlation coefficient for each method, confirming the pattern I saw in the PCA scatter plot (Figure 20).

As expected, the negative control where I calculated the PCA directly on the raw counts performed worsts; it had an almost perfect correlation between the size factor and principal component one. Interestingly, the second negative control, where I divided the raw counts by the size factors, performed well ($\rho = 0.3$). This demonstrates how heteroskedasticity does not necessarily bias the results of PCA.

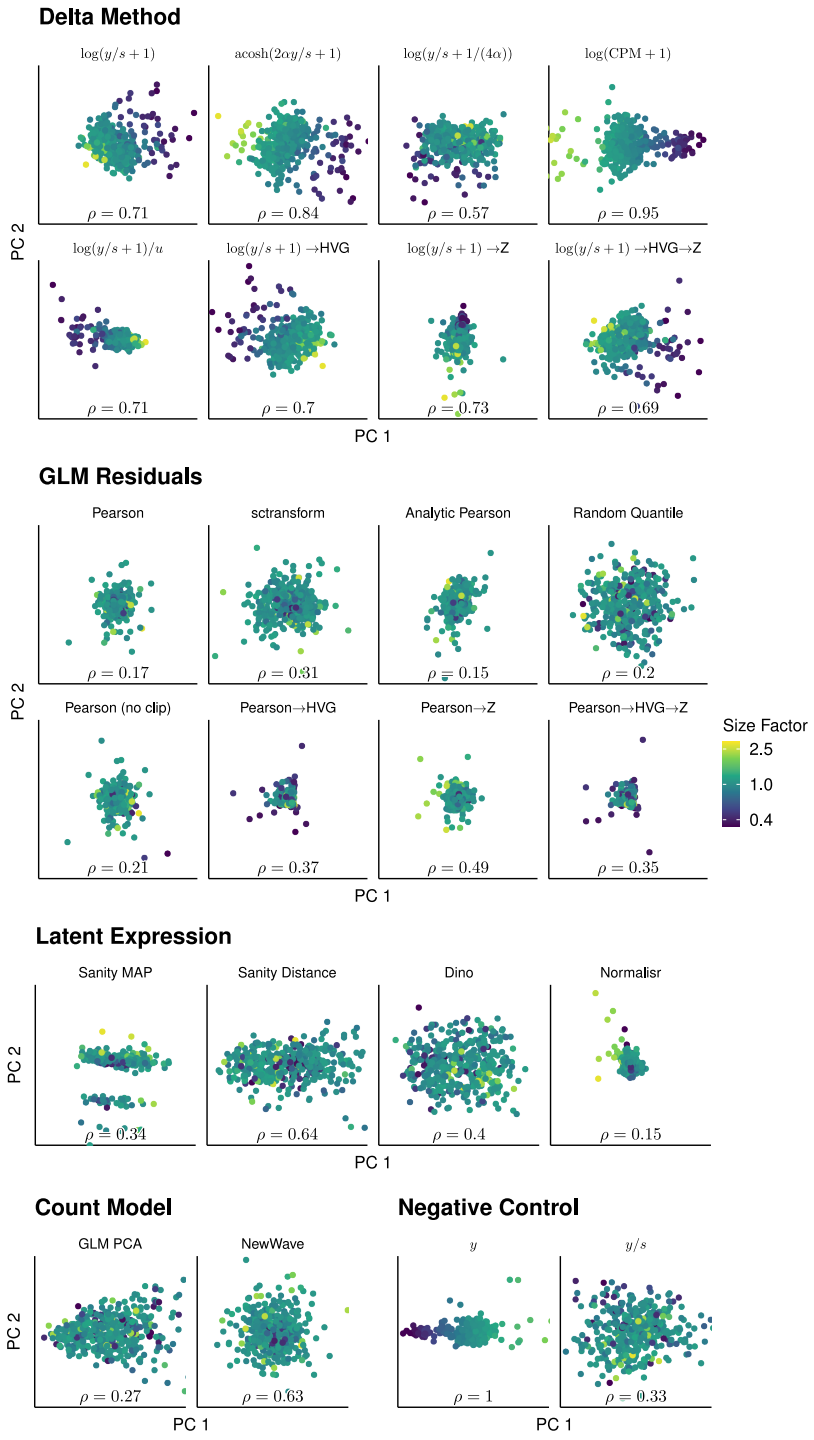


Figure 20 | Scatter plot of the first two principal components after transforming the counts of an ostensibly homogeneous cell line population. Each point is a cell colored by its size factor, which is on a logarithmic scale. The number at the bottom of the plot is the canonical correlation coefficient ρ (Hotelling, 1936) of the size factor and the first ten principal components.
The data for this figure was generated by me and adapted from Ahlmann-Eltze and Huber (2022).

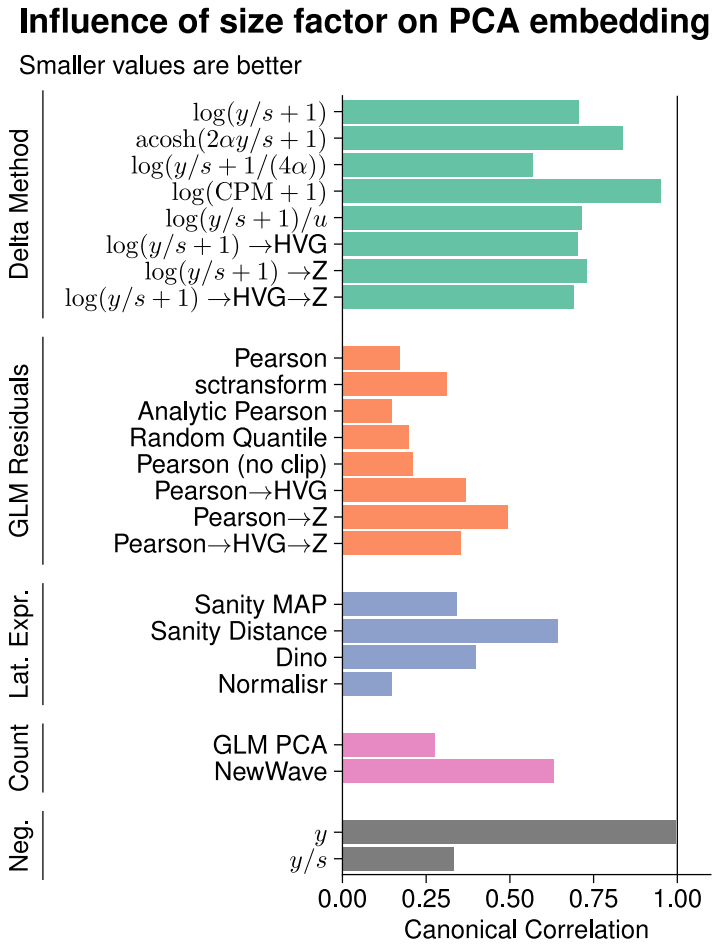


Figure 21 | Bar chart of the canonical correlation coefficient between size factor and principal components. The data for this figure was generated by me and adapted from Ahlmann-Eltze and Huber (2022).

3.5.2 Empirical mean-variance relationship after transformation

One criticism by Hafemeister and Satija (2019) of delta method-based variance-stabilizing transformations is that they fail to stabilize the variance of genes with a small mean. In fact, Warton (2018) showed that it is mathematically impossible for a transformation to stabilize the counts of genes which are mostly zero.

To assess the empirical mean-variance relation after applying each transformation, I applied each transformation to 5 000 genes of a human hematopoietic development dataset (Bulaeva et al., 2020) (Figure 22). As expected, the negative controls showed a very strong dependency of the variance on the mean (Figure 22 bottom row). The delta method-based variance-stabilizing transformations, as predicted by Warton (2018) and observed by Hafemeister and Satija (2019), showed a dependency of the variance on the mean: for genes with a mean of less than roughly 0.1 the variance is close to zero. The pattern for the latent expression-based transformations

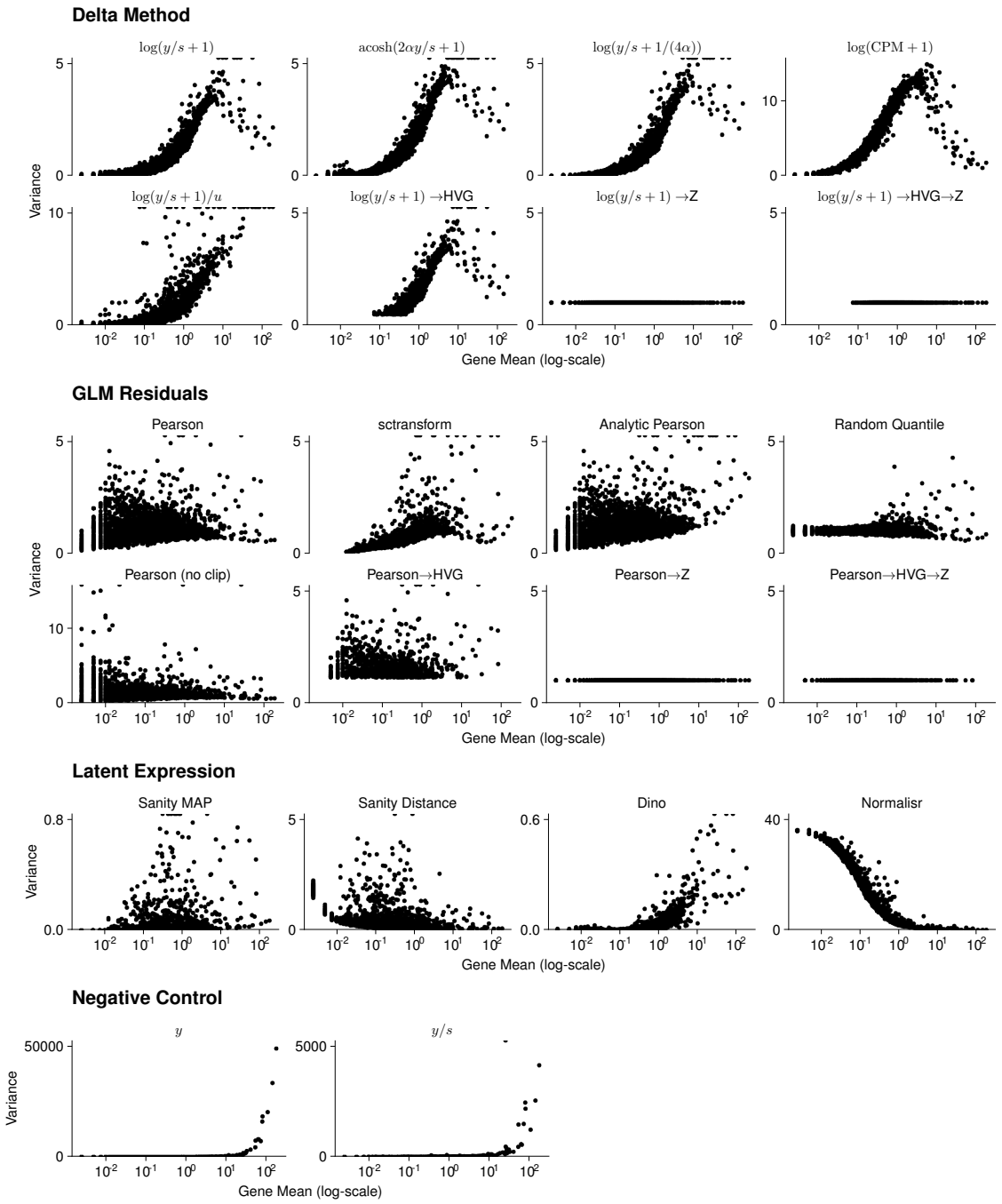


Figure 22 | Scatter plot of empirical mean-variance relationship for 5 000 **genes after applying the transformations.** Each point is a gene, and the x-axis is log-scaled. Note that the y-axis differs for the raw counts, $\log(CPM + 1)$, $\log(y/s + 1)/u$, Pearson (no clip), Sanity MAP, DINO, and Normalisr for aesthetic purposes.
The data for this figure was generated by me and adapted from Ahlmann-Eltze and Huber (2022).

depended on the specific implementation: for Dino, the pattern resembled the delta method-based variance-stabilizing transformations; for Normalisr, it was the opposite, i.e., lowly expressed genes had more variance than highly expressed genes.

The clipping helped to suppress very high variances for lowly expressed genes for the Pearson

residuals. The highly variable gene selection produced the same pattern as the transformation without highly variable gene selection, except that the bottom $4,000$ genes were removed. The z transformation equalized the variance of all genes and thus the results look like a horizontal line in the scatter plot.

The count models *glmPCA* and *NewWave* do not produce transformed data for which I could have calculated the mean and variance, so they are absent from Figure 22.

3.5.3 Stabilization of a gene with a bimodal expression pattern

In Section 3.4.1, I explained that Pearson residuals are linear transformations of the data and thus struggle to adjust the heteroskedasticity within a gene. In Figure 23 shows the histograms of a marker gene of type II pneumocytes which has a bimodal gene expression pattern in the mouse lung epithelium dataset from Angelidis et al. (2019).

As expected, the histogram of the Pearson residuals was a linearly scaled version of the negative control of the counts divided by the size factors. In contrast, my proposed randomized quantile residuals looked more similar to the delta method-based variance stabilized values. The problem of not stabilizing the variance of the larger peak is that this obscures the actual information of the larger counts relative to each other because random variation within the high-expression groups dominates.

Figure 23 also shows that choosing count per million in the logarithm creates a large gap between the zeros and the rest of the *Other cells* population, whereas for $\log(y/s + 1)$ the gap is much smaller.

Figure 24 summarizes the results of Figure 20-23

3.6 Empirical comparison of the transformation approaches

Understanding the conceptual differences between the transformation approaches is most salient when developing new methods or applying them to novel data types. However, for most analysts of existing single-cell data, empirical performance will be the primary interest. The performance of a preprocessing method cannot be judged just by itself because the preprocessing exists to facilitate downstream analyses. Here, I will judge the performance on their ability to recover or uncover the latent local structure of the data manifested in the k nearest neighbor graph.

A major challenge for any benchmark is what to use as the ground truth, i.e., how can I decide that one method performed better than another? I used three separate benchmarks to get a comprehensive picture of the performance, where I traded off the realism of the data against certainty about the ground truth.

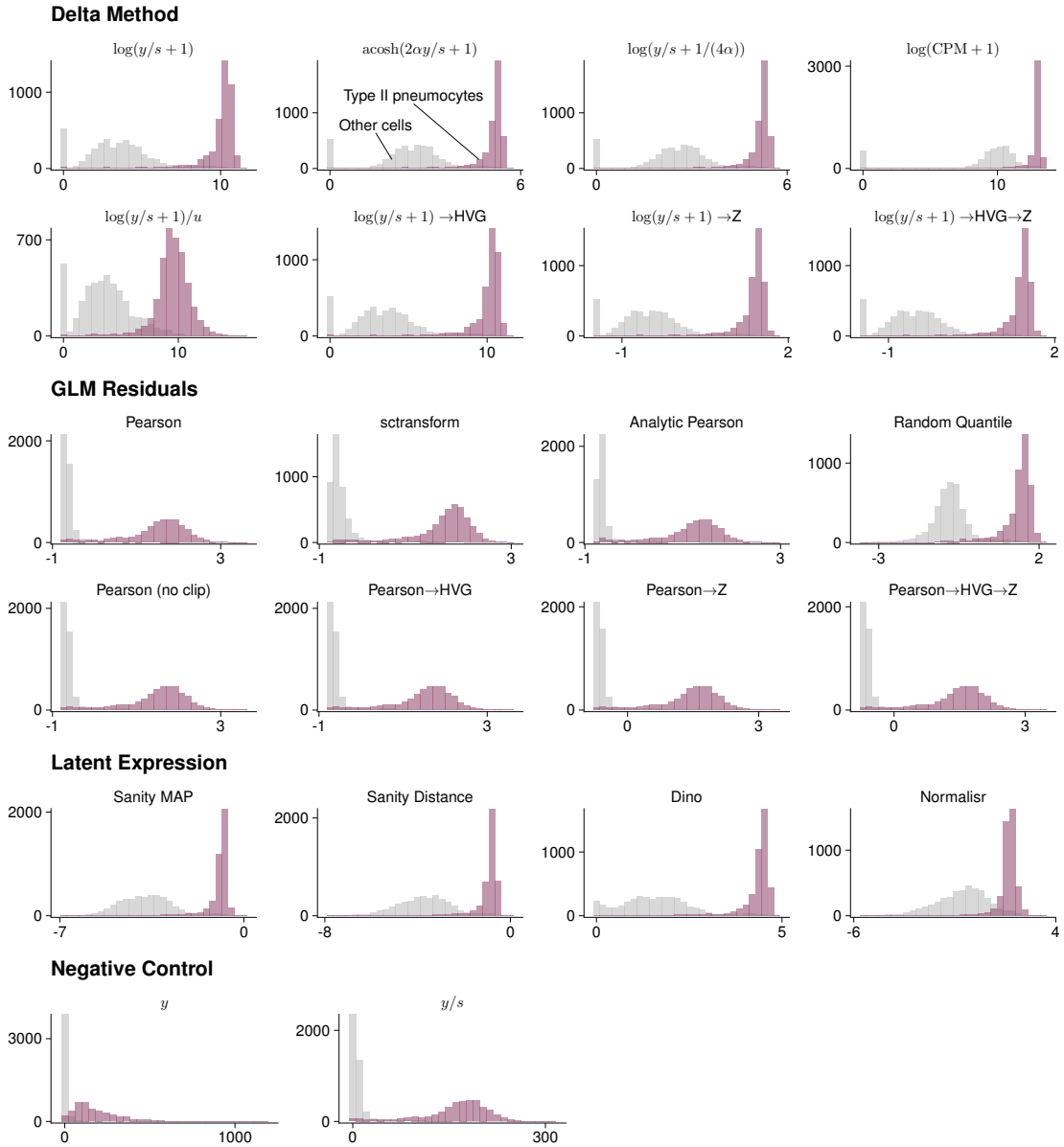


Figure 23 | Histogram of the transformed expression values for *Sftpc*, a single gene with a bimodal expression pattern, colored by cell-type For visualization purposes, I subsampled the number of cells of the *Other cells* to match the size of the *Type II pneumocyte* cell population. The data for this figure was generated by me and adapted from Ahlmann-Eltze and Huber (2022).

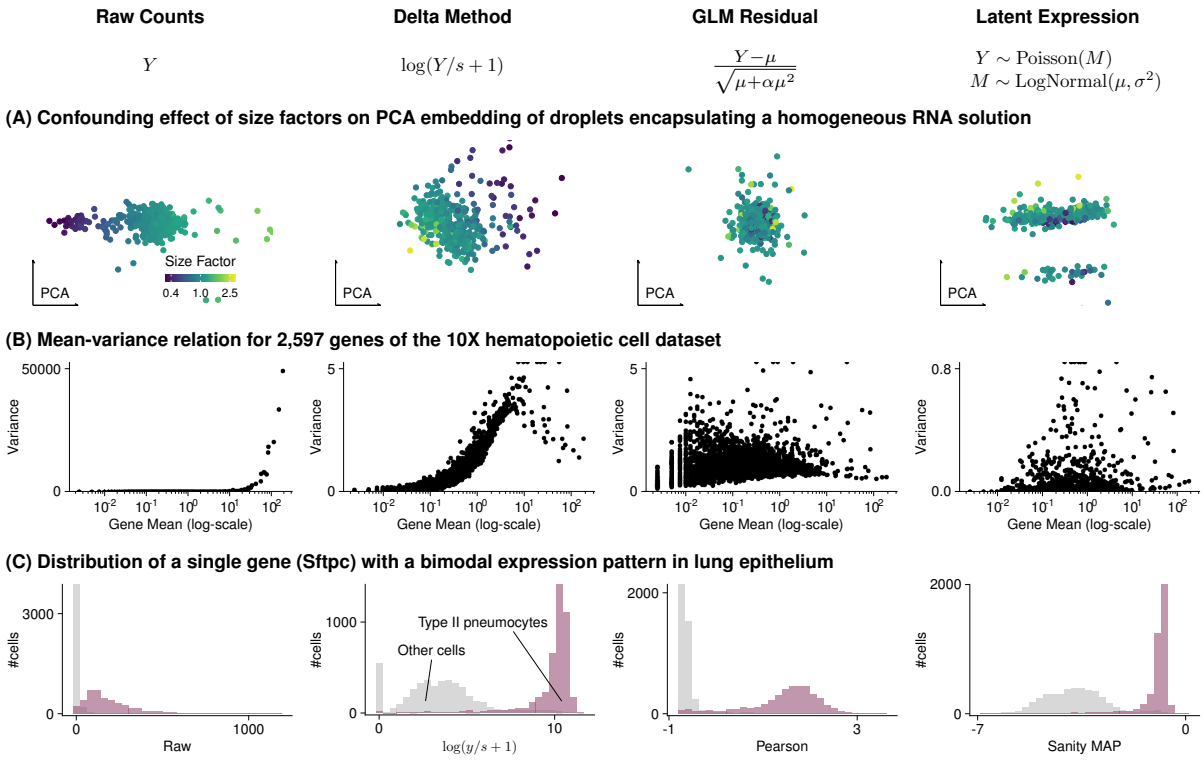


Figure 24 | Overview of the conceptual differences between the transformation approaches. The figure summarizes the most important results of (A) Figure 20, (B) Figure 22, and (C) Figure 23.
The data for this figure was generated by me and adapted from Ahlmann-Eltze and Huber (2022).

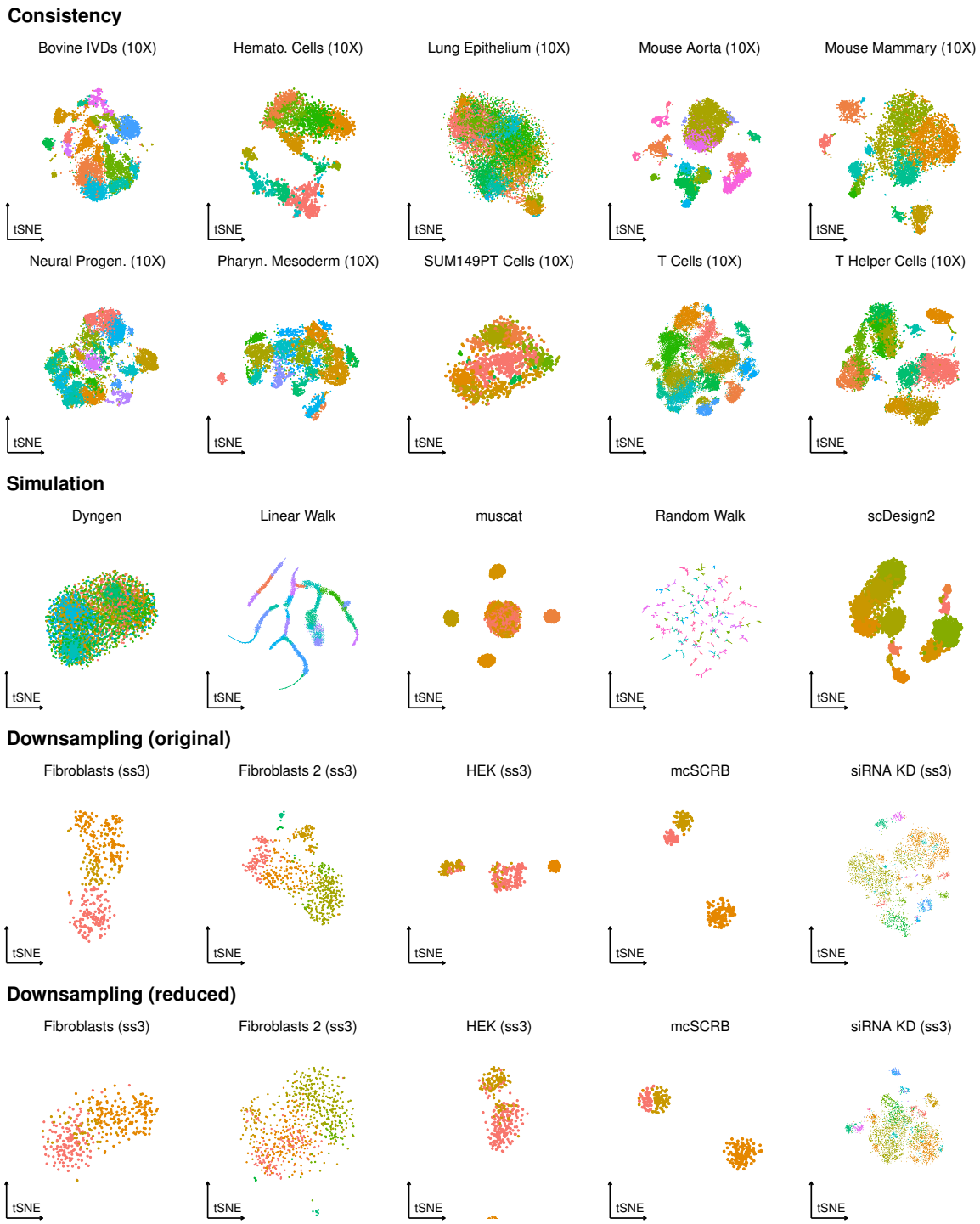


Figure 25 | Scatter plot of the t-stochastic neighborhood embeddings of each benchmark dataset Each point is a cell colored by unsupervised clustering in the PCA space with the walktrap algorithm.
The data for this figure was generated by me and adapted from Ahlmann-Eltze and Huber (2022).

3.6.1 Dataset overview

Table 1 | Overview of the datasets used for the benchmark. This table is adapted from Ahlmann-Eltze and Huber (2022)

	#Cells	#Genes	Perc. Zeros	99% Quant	UMI/cell	Overdisp.
Consistency						
Hematopoietic Cells	2,838	21,398	87%	12	5,020	0.33
SUM149PT Cells	1,196	25,231	74%	35	54,900	0.14
Lung Epithelium	11,407	20,728	90%	5	7,730	0.17
Pharyngeal Mesoderm	7,581	19,939	79%	19	21,700	0.12
Neural Progenitors	13,572	25,711	87%	7	11,500	0.31
Mouse Mammary	6,969	19,757	89%	6	6,970	0.24
Mouse Aorta	10,477	20,020	86%	8	9,420	0.89
Bovine IVDs	8,231	17,464	90%	6	3,940	1.20
T Helper Cells	10,064	21,153	83%	15	19,300	0.33
T Cells	43,283	23,978	92%	4	5,360	0.53
Simulation						
Dyngen	5,000	995	75%	3	291	0.20
Linear Walk	8,569	17,130	90%	5	4,340	2.20
muscat	5,000	999	63%	22	1,830	0.98
Random Walk	8,569	17,192	90%	5	4,820	2.60
scDesign2	2,838	16,199	82%	15	5,170	0.35
Downsampling (original)						
mcSCRB	249	16,864	57%	48	59,000	0.47
Fibroblasts	369	16,535	45%	224	199,000	0.82
Fibroblasts 2	737	18,682	48%	181	197,000	0.33
HEK	339	18,746	63%	38	56,100	0.15

siRNA KD	4,298	18,956	56%	106	122,000	0.36
Downsampling (reduced)						
mcSCRB	249	16,864	87%	5	5,020	0.32
Fibroblasts	369	16,535	85%	6	5,020	0.19
Fibroblasts 2	737	18,682	88%	5	5,020	0.13
HEK	339	18,746	89%	4	5,140	0.11
siRNA KD	4,298	18,956	88%	5	4,990	0.23

First, I used ten existing single-cell datasets and measured the overlap of the k nearest neighbors for each cell after splitting the genes into disjoint halves. This benchmark measured the *consistency* of the data. The data is as realistic as possible but producing consistent results is not sufficient (nonetheless desirable) for a good preprocessing method.

Second, I used five *simulation* frameworks to generate data where I knew the ground truth precisely for each cell.

Third, I took five published datasets with an exceptionally high sequencing depth. For each cell, I determined reliable nearest neighbors by considering the intersection of the k nearest neighbors across all transformations on the deeply sequenced data. Then, I compared these reliable nearest neighbors against the k nearest neighbors after *downsampling* the counts to levels typical for droplet-based experiments. The reliable nearest neighbors served as a proxy for the ground truth and this benchmark thus balances a realistic structure of the data with an informative ground truth. Our preprint (Ahlmann-Eltze and Huber, 2022) was, to our knowledge, the first to use such an approach.

Figure 25 shows scatter plots of the t-stochastic neighborhood embedding (tSNE) (Van der Maaten and Hinton, 2008) and Table 1 lists some properties of all 20 datasets used in the benchmarks. The datasets used in the downsampling benchmark (except for the siRNA knockdown (KD) data) had fewer cells and were less complex than the data for the consistency benchmark. This is because the deeply sequenced data came from Smart-seq3 (Hagemann-Jensen et al., 2020) and mcSCRB datasets (Bagnoli et al., 2018) which are, unlike 10x datasets, plate and not droplet-based. The data generated by the simulation frameworks has similar properties as the 10x datasets, except for the cell type complexity. The latent structure of the dyngen and muscat data was simpler than the 10x datasets, whereas the linear and random walk datasets are more complex. In particular, the data lacked the multi-level structure apparent in the 10x data.

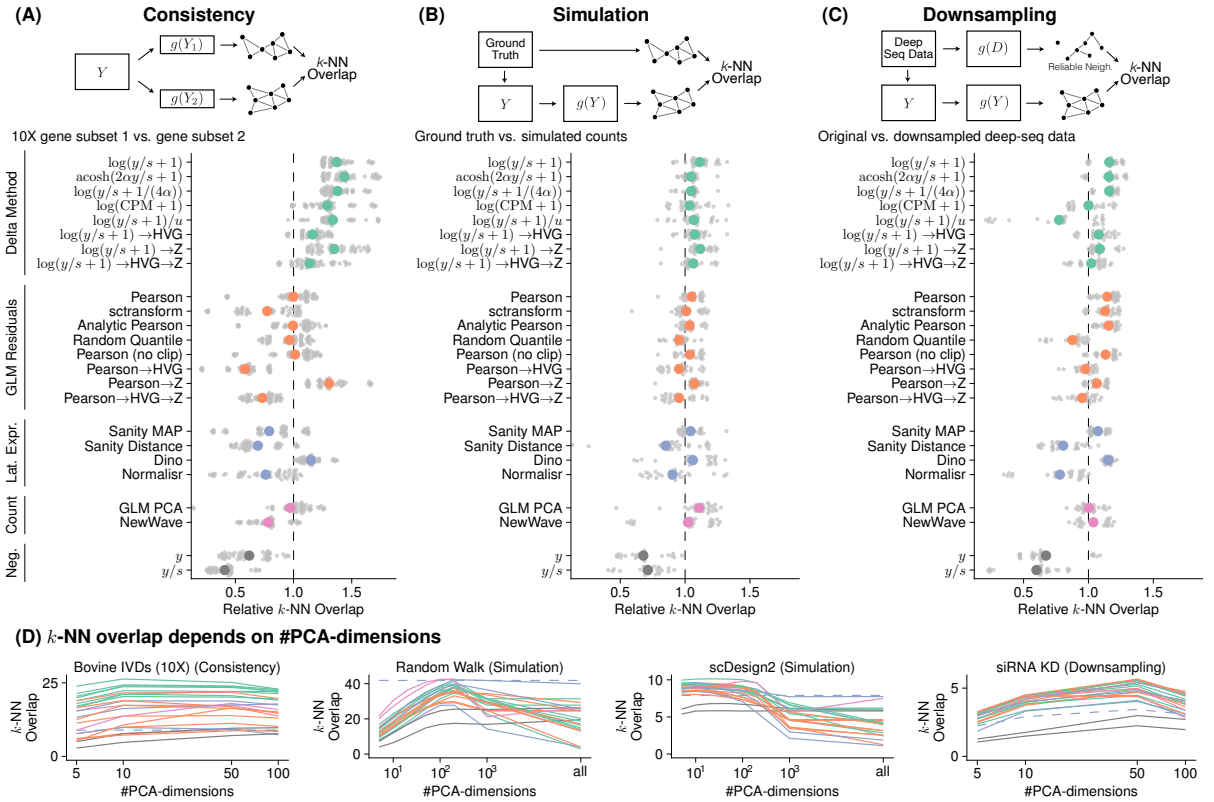


Figure 26 | Benchmark result summary. (A-C) Aggregated results for the consistency, simulation, and downsampling benchmark. The grey points show the average overlap of the k-nearest neighbor graphs between the ground truth and the transformed data. To make the results comparable across underlying data, the performance is plotted relative to the average performance per transformation. (D) Line plots that show the dependence of the k-nearest neighbor overlap on the number of dimensions used for the principal component analysis (PCA). The lines are colored using the same color scheme as in panel A-C. The performance of Sanity Distance is plotted as a dashed line because it does not depend on the number of PCA dimensions, as it calculates the cell-by-cell distance matrix without an intermediate dimension reduction step. The data for this figure was generated by me and adapted from Ahlmann-Eltze and Huber (2022).

3.6.2 Benchmark results

I ran 22 transformations across all 20 datasets and Figure 26A-C shows the aggregated results for the three benchmark types. The pictogram on top gives a stylized overview how the benchmark was conducted. Each benchmark was repeated five times and the large colored dot summarizes the results across five to ten datasets. Figure 27 shows the underlying data from each dataset. To make the values comparable across datasets, I assessed the fold change compared with the mean across all transformations (dashed line in Figure 27).

In Figure 26 and 27, I show the results for $k = 50$ nearest neighbors. To exclude the possibility that the results are an artifact of the k, I repeated the calculation for $k = 10$ and $k = 100$ in Figure 28. The results confirmed the patterns that I found in Figure 26.

Dimension reduction of the transformed values before searching for the k nearest neighbors is

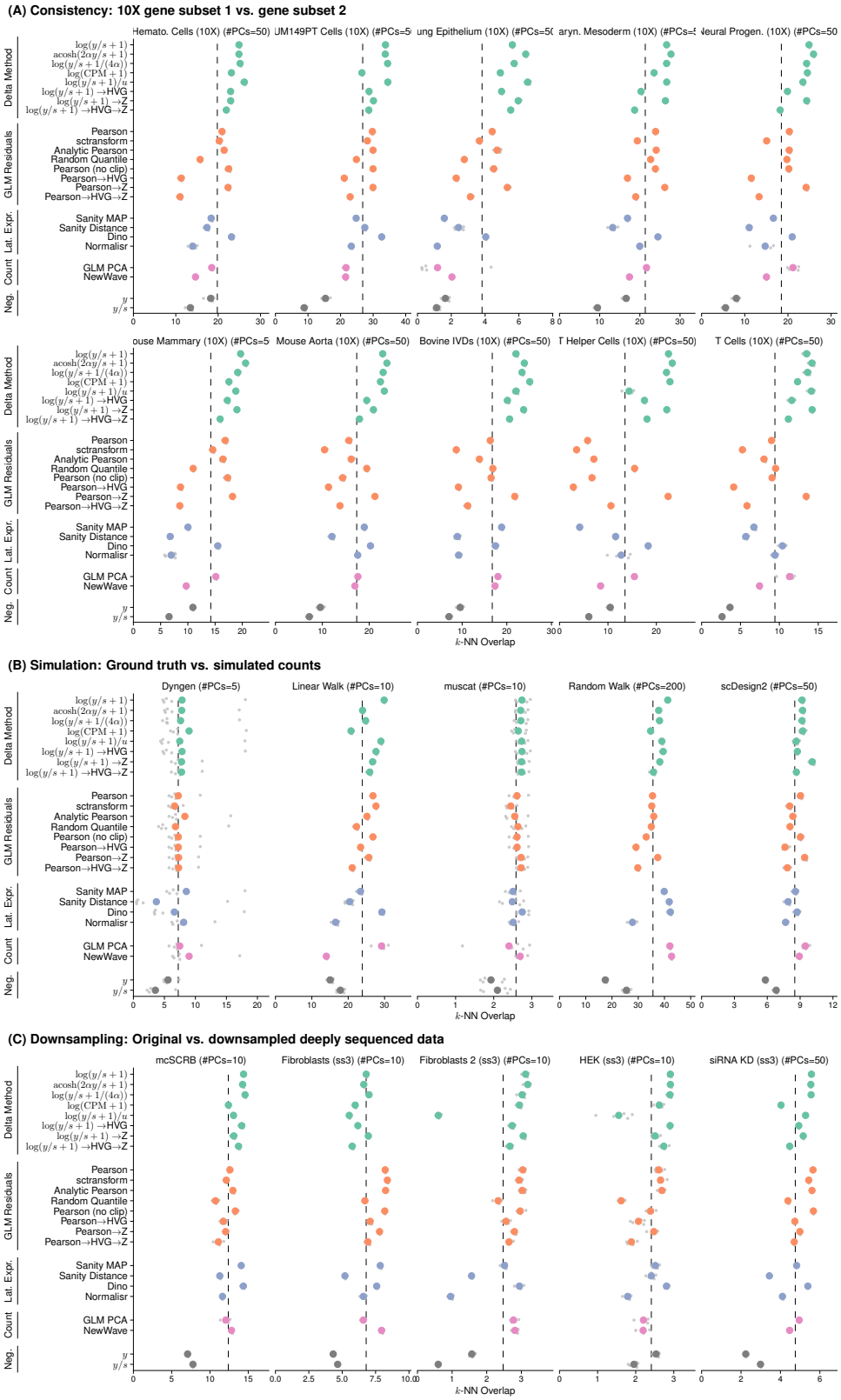


Figure 27 | Benchmark result for each dataset.
The data for this figure was generated by me and adapted from Ahlmann-Eltze and Huber (2022).

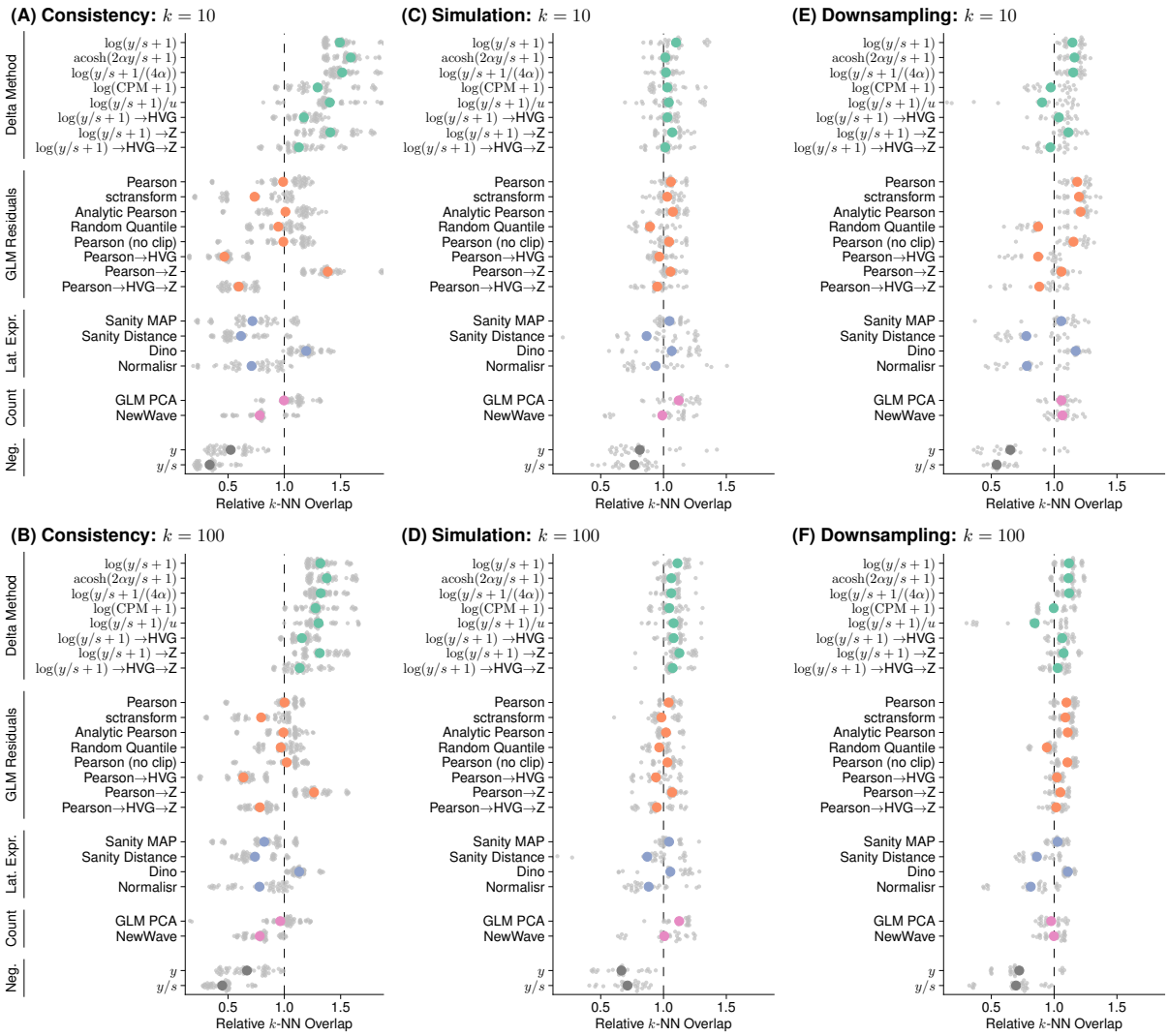


Figure 28 | Benchmark result for other k.
The data for this figure was generated by me and adapted from Ahlmann-Eltze and Huber (2022).

an important step to get good performance (Figure 26D). I ran the consistency and downsampling benchmarks with 5, 10, 50, and 100 dimensions and the simulations additionally with 200, 1 000, and without PCA. The optimal number of principal component dimensions was dataset dependent, but both too small and too large number of dimensions were detrimental to performance. At first, it might seem surprising that increasing the number of PCA dimensions decreased the performance because PCA is a lossy compression. However, the lossy nature of PCA was beneficial because it smoothed away some of the uncorrelated Poisson noise in the data, which otherwise negatively affects the k nearest neighbor inference.

The explanatory power of the downsampling benchmark depended on the accuracy of the reliable nearest neighbors. In Figure 29, I show the pairwise overlap of the k nearest neighbors for all transformations inferred on the deeply sequenced data. The heatmaps mostly clustered along the approaches (e.g., two delta method-based variance-stabilizing transformations produced

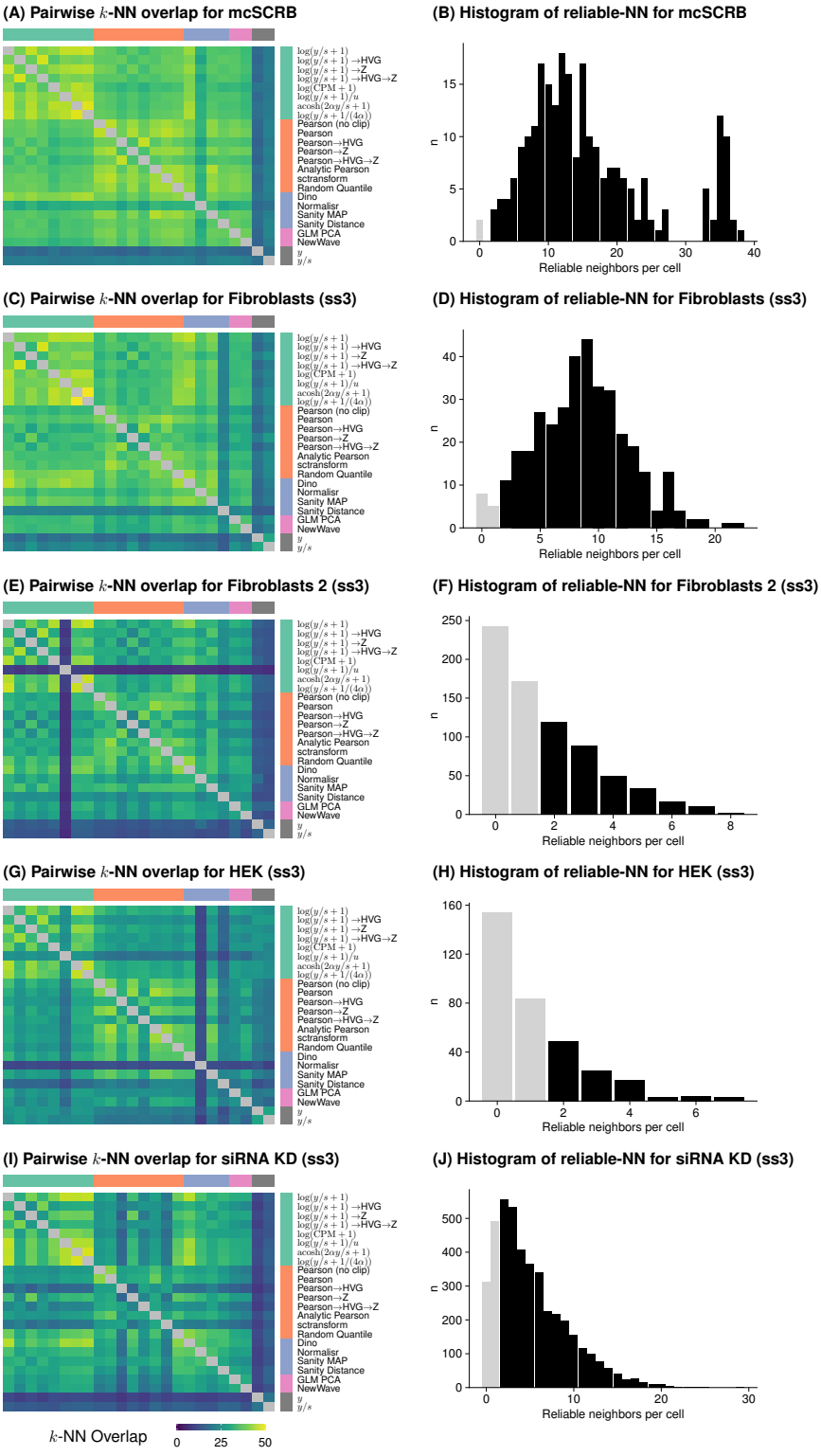


Figure 29 | Overview of reliable neighbor inference. The left column shows heatmaps of the overlap between pairs of transformations. The right column shows the number of reliable nearest neighbors for all cells across datasets. The black bars come from the cells that are used to calculate the performance in Figure 27.
The data for this figure was generated by me and adapted from Ahlmann-Eltze and Huber (2022).

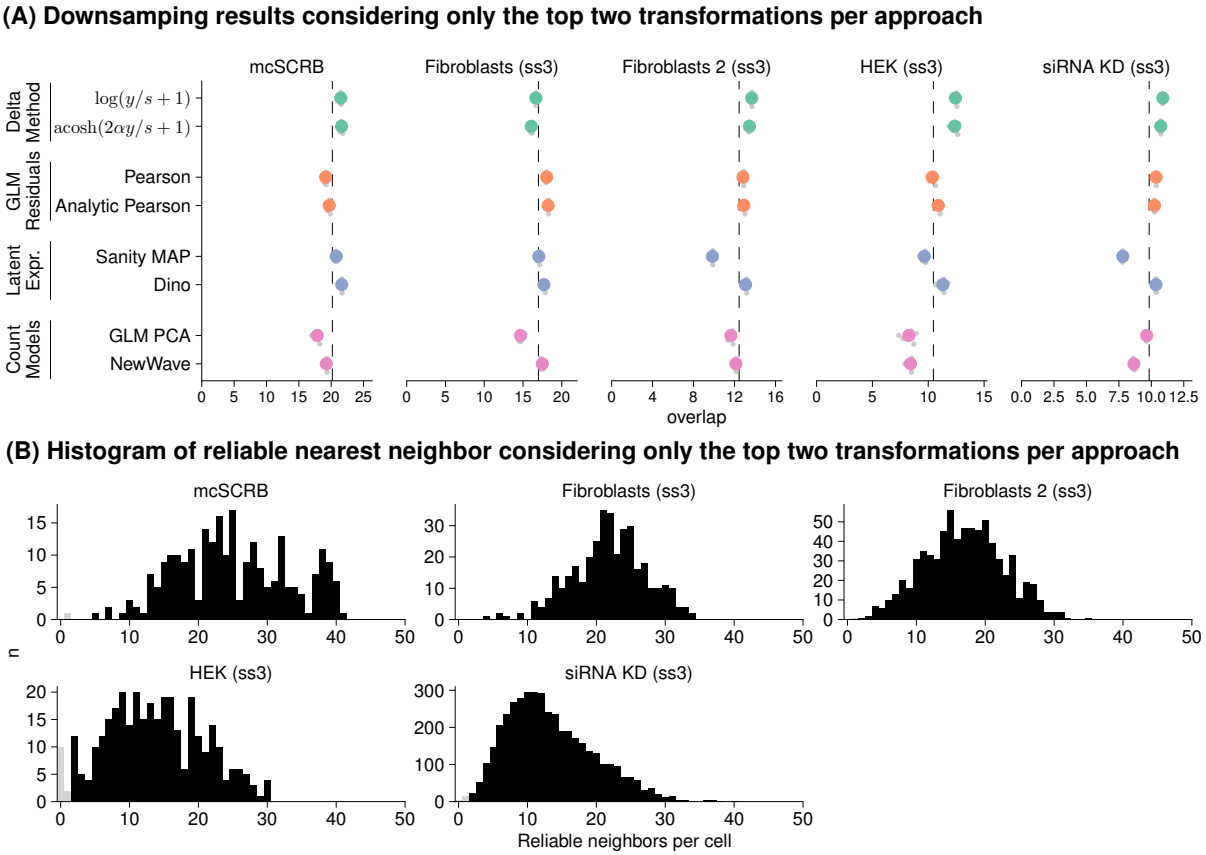


Figure 30 | Results for the downsampling benchmark considering a reduced set of transformations. (A) Performance overview for the best two transformations per approach across the five deeply sequenced datasets. (B) Histogram of the reliable nearest neighbors per cell considering the restricted set of transformations (analogous to Figure 29).
The data for this figure was generated by me and adapted from Ahlmann-Eltze and Huber (2022).

results that were more similar than the log transformation and the Pearson residuals). The two negative controls (not considered for the reliable nearest neighbors) were outliers, dissimilar to all other transformations. For the Fibroblasts 2 dataset, the $\log(y/s+1)/u$ method was an outlier sharing almost no neighbor with the other methods. Similarly, the Normalisr method was an outlier in the mcSCRB and the HEK datasets.

To confirm that the outlier transformations did not negatively impact the accuracy of the downsampling, I repeated the benchmark with a different set of reliable nearest neighbors based on the top two transformations per approach (Figure 30A). The benchmark confirmed the patterns that I observed originally. Figure 30B shows that the average number of reliable nearest neighbors is larger with the reduced set of transformations than with all transformations (Figure 29 right column).

So far, I have focused my analysis exclusively on the k nearest neighbor overlap. In Figure 31A,B, I show the adjusted rand index and adjusted mutual information of the ground truth clustering with the clustering after transformation on the simulation datasets. The results were

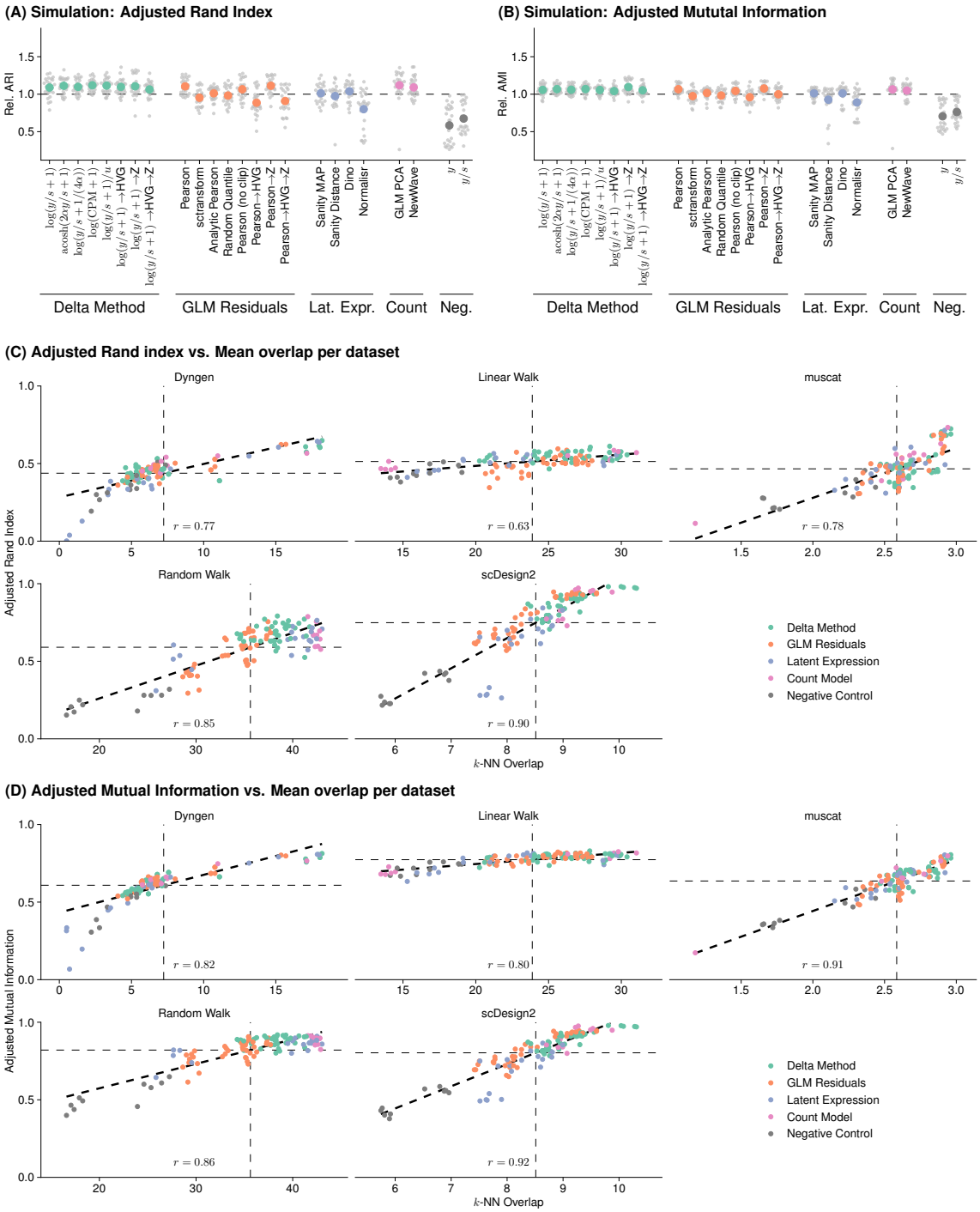


Figure 31 | Simulation benchmark results using cluster consistency metrics (A,B) Benchmark results for the simulation frameworks quantified by comparing (A) the adjusted rand index and (B) the adjusted mutual information of the ground truth clusters with the walktrap clustering results on the transformed data. (C,D) Scatter plot of the k nearest neighbor overlap against (C) the adjusted rand index and (D) the adjusted mutual information. The number ρ at the bottom is the correlation coefficient.
The data for this figure was generated by me and adapted from Ahlmann-Eltze and Huber (2022).

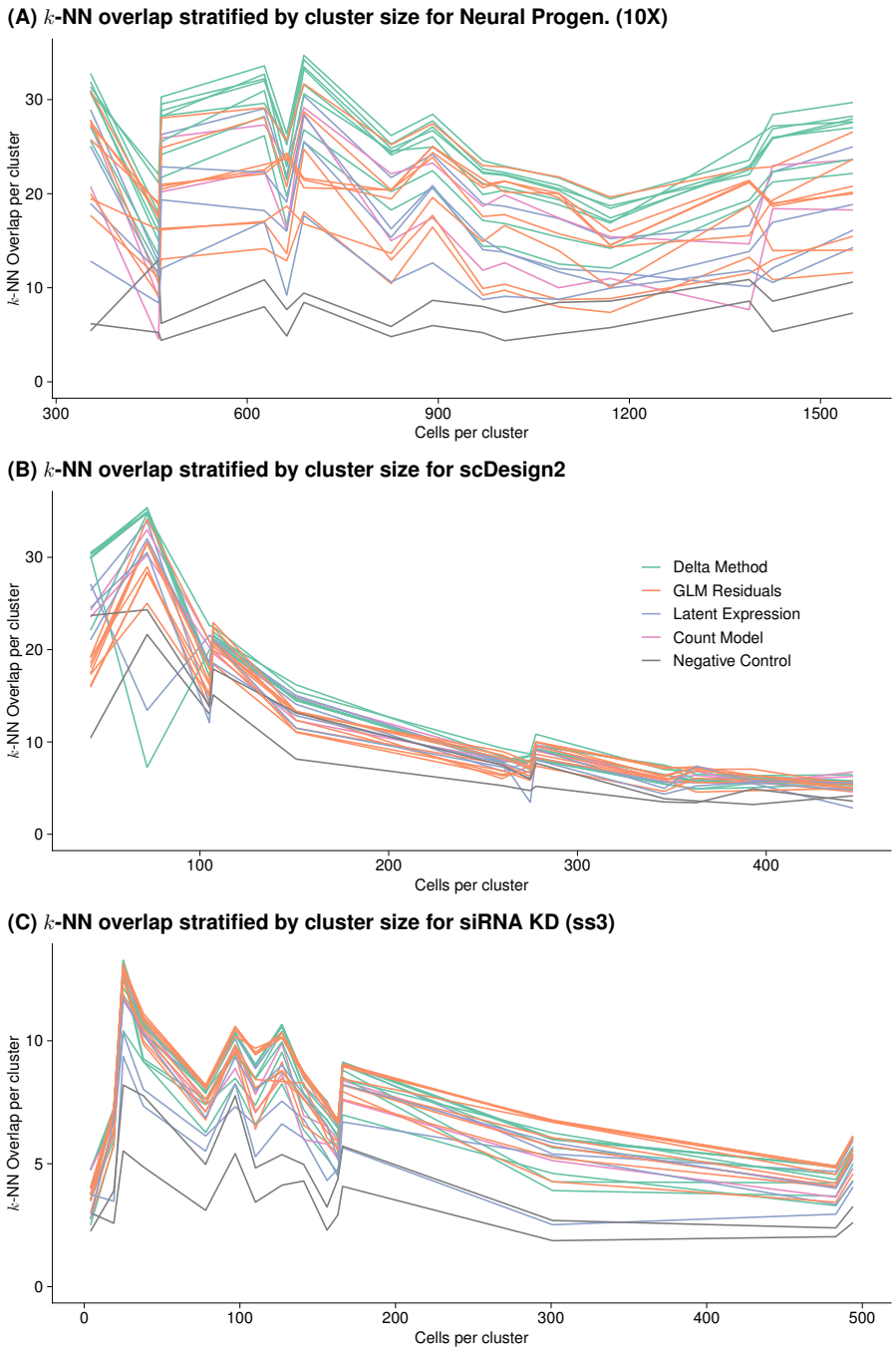


Figure 32 | k nearest neighbor overlap dependence on cluster size.
The data for this figure was generated by me and adapted from Ahlmann-Eltze and Huber (2022).

consistent with my earlier analysis based on the k nearest neighbor overlap, which is expected because the different metrics are highly correlated (Figure 31C,D). For the linear walk simulation, the k nearest neighbor overlap metric had a larger dynamical range than the adjusted rand index and the adjusted mutual information, demonstrating its advantage on data with mainly continuous gradients (instead of discrete clusters).

I further explored if the ability of transformations to uncover the local structure of the data

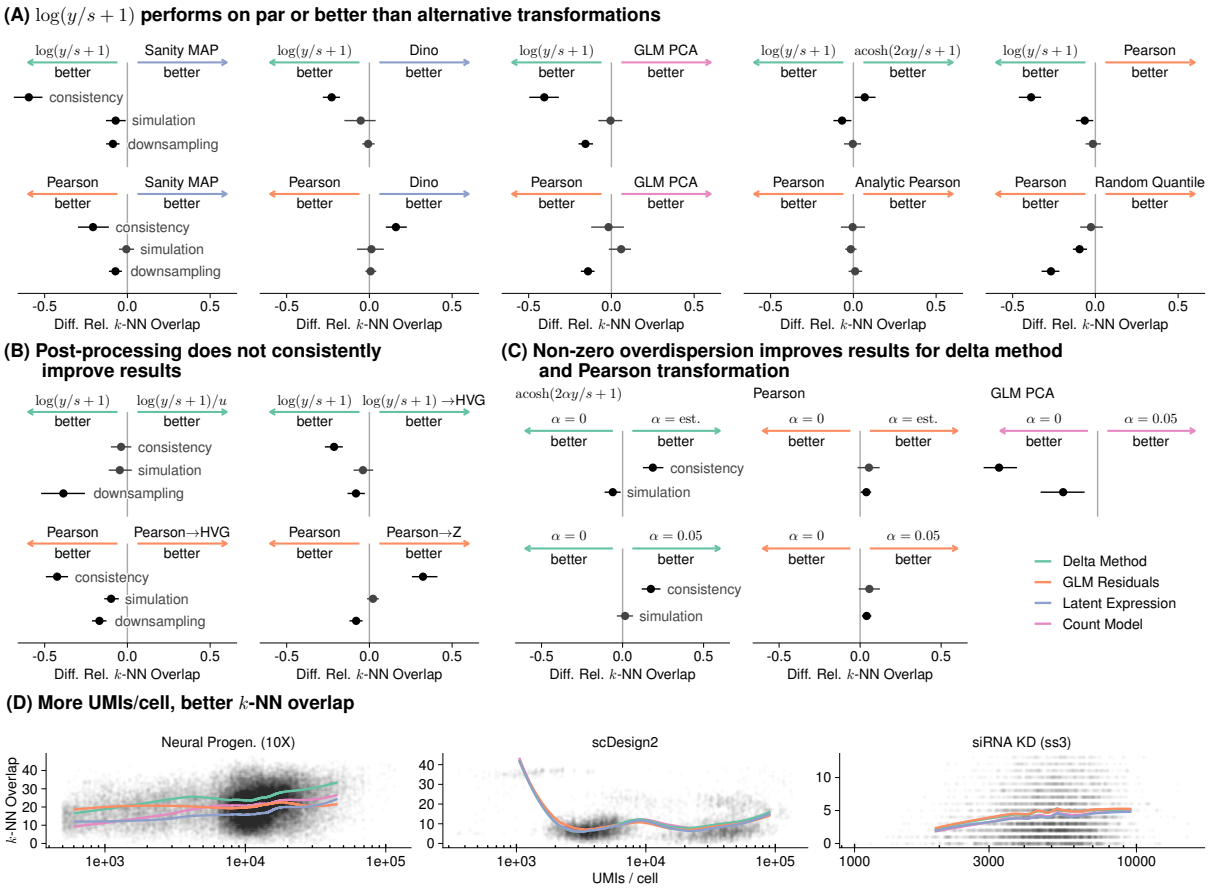


Figure 33 | Direct comparison of some selected transformation approaches (A-C) The bars show the 95% confidence intervals calculated using the five replicates across the five/ten datasets. The arrows are colored by the transformation approach. (A) Comparison of the shifted logarithm and Pearson residuals against the best transformation from the other approaches. (B) Comparison of post-processing steps such as highly variable gene selection or z transformation. (C) Comparison of the overdispersion settings across approaches. (D) Line plot of the k nearest neighbor overlap dependence on the sequencing depth per cell.
The data for this figure was generated by me and adapted from Ahlmann-Eltze and Huber (2022).

depended on how dense a particular area of the latent space was. As a proxy of the density, I used the size of the cluster, but no clear trend was apparent (Figure 32).

3.6.3 Direct comparison

In Figure 33A, I directly compared the results of the shifted logarithm and the Pearson residuals against the respective best-performing approaches for the other transformation approaches. The shifted logarithm produced more consistent results than other transformation approaches and outperformed the Sanity MAP and GLM PCA in the downsampling benchmark. The randomized quantile transformation that I developed to address some of the problems with the Pearson residuals, performed worse in the simulation and downsampling benchmark than the Pearson residuals.

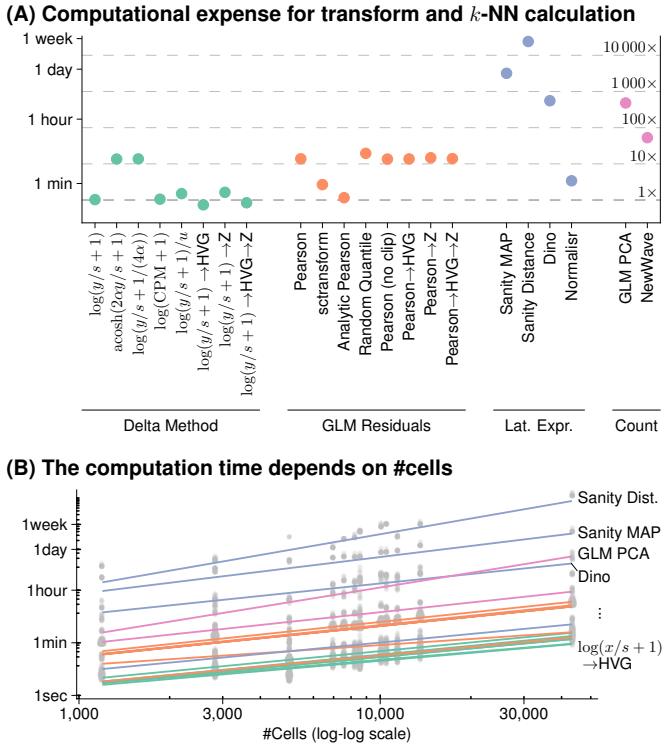


Figure 34 | Comparison of the runtime on the 10x human T-helper cell dataset. (A) Runtime plotted on a log scale, the secondary axis shows the relative duration compared with the shifted logarithm. (B) Scaling of the methods with the number of cells. A double-logarithmic plot of the runtime dependence on the number of cells. Most transformations scale with a slope of 1 (i.e., linearly) except for Sanity Distance and GLM PCA which have a slope > 1.5 (this suggests a quadratic dependence of the runtime on the number of cells).
The data for this figure was generated by me and adapted from Ahlmann-Eltze and Huber (2022).

Post-processing the results using sphereing, highly variable gene selection and z transformation degrades the performance of the shifted logarithm. The results were similar for the Pearson residuals except that the z transformation improved the consistency of the results (Figure 33B).

Including a non-zero overdispersion improved the performance for the acosh transformation and the Pearson residuals in the consistency and simulation benchmark (Figure 33C). (I did not compare the effect in the downsampling benchmark to keep the number of transformations contributing to the reliable nearest neighbors in check.) I did not find a difference between estimating the overdispersion from the data and fixing it to a small non-zero value (i.e., $\alpha = 0.05$). GLM PCA performed worse for a non-zero overdispersion which might be due to the additional complication inferring the parameters of the gamma-Poisson model.

Figure 33D shows that the methods performed better with increased sequencing depth per cell. This makes sense because the increased number of counts allows a more precise inference of the latent state of the cell (see section 3.1.1).

In Figure 34A, I compared the runtime of all transformations on the 10x human T-helper cell dataset. The plot shows the large difference between the fastest methods (shifted logarithm with a

fixed overdispersion and the analytical approximation of the Pearson residuals) which took a few seconds and Sanity Distance which needed multiple days of CPU time for the same dataset. I also compared the runtime across datasets as a function of the number of cells (Figure 34B). Most transformations scaled linearly with the number of cells, except for Sanity Distance and GLM PCA which appear to scale quadratically (i.e., their slope is > 1.5 in the double logarithmic plot).

3.6.4 Methods

This section is a copy of the methods section from Ahlmann-Eltze and Huber (2023b), which I wrote originally. The only edits are formatting and figure references.

We compared 22 transformations that can be grouped into four approaches.

The delta method-based transformations were: the shifted logarithm ($\log(y/s+1)$), the acosh transformation ($\operatorname{acosh}(2\alpha y/s+1)$), the shifted logarithm with pseudo-count dependent on the overdispersion ($\log(y/s+1/(4\alpha))$), the shifted logarithm with counts-per-million ($\log(\mathrm{CPM}+1)$), the shifted logarithm with subsequent size normalization as suggested by Booeshaghi et al. (2022) (x_{gc}/u_c, where $x_{gc} = \log(y_{gc}/s_c + 1)$ and $u_c = \sum_g x_{gc}$), the shifted logarithm with subsequent highly variable gene selection ($\log(y/s+1) \rightarrow$ HVG), the shifted logarithm with subsequent z-scoring per gene ($\log(y/s+1) \rightarrow$ Z), the shifted logarithm with subsequent highly variable gene selection and z-scoring per gene ($\log(y/s+1) \rightarrow$ HVG $\rightarrow$ Z). For all composite transformations, we first calculated the variance-stabilizing transformation, then chose the highly variable genes and used the results without recalculating the variance-stabilizing transformation.

To retain the sparsity of the output also if the pseudo-count $y_0 \neq 1$, *transformGamPoi* uses the relation

$$\log\left(\frac{y}{s} + y_0\right) = \log\left(\frac{y}{y_0\, s} + 1\right) + \log y_0. \tag{67}$$

Subtracting the constant $\log y_0$ from this expression does not affect its variance-stabilizing properties, but has the desirable effect that data points with $y = 0$ are mapped to 0.

The residuals-based transformations were: Pearson residuals implemented with the *transformGamPoi* package where each residual is clipped to be within $\pm\sqrt{\#\mathrm{Cells}}$, as suggested by Hafemeister and Satija (2019) (Pearson), Pearson residuals with clipping and additional heuristics implemented by *sctransform* Version 2 (sctransform), an analytic approximation to the Pearson residuals with clipping suggested by Lause et al. (2021) (Analytic Pearson), randomized quantile residuals implemented by *transformGamPoi* (Random. Quantile), Pearson residuals without clipping implemented by *transformGamPoi* (Pearson (no clip)), Pearson residuals with clipping and subsequent highly variable gene selection (Pearson $\rightarrow$ HVG), Pearson residuals with clipping and subsequent z-scoring per gene (Pearson $\rightarrow$ Z), Pearson residuals with clipping and subsequent highly variable gene selection and z-scoring per gene (Pearson $\rightarrow$ HVG $\rightarrow$ Z). For each composite Pearson residual transformation (i.e., with HVG. and / or z-scoring), we used the *transformGamPoi* implementation.

The latent expression-based transformations were: *Sanity* with point estimates for the latent

expression (*Sanity MAP*) and with calculation of all cell-by-cell distances taking into account uncertainty provided by the posteriors (*Sanity Distance*), *Dino* as provided in the corresponding R package, and *Normalisr* with variance normalization, implemented in Python, which we called from R using the *reticulate* package.

The count-based factor analysis models were: *GLM PCA* using the Poisson model and the Gamma-Poisson model with $\alpha = 0.05$. In the figures, we show the results for the Poisson model unless otherwise indicated. We used the *avagrad* optimizer. We ran *NewWave* with 100 genes for the mini-batch overdispersion estimation.

For the delta method-based transformations and the residuals-based transformations calculated with the *transformGamPoi* package, we calculated the size factor s using Eq. (51).

We defined highly variable genes (HVG) as the 1 000 most variable genes based on the variance of the transformed data.

For z-scoring, we took the transformed values $x_{gc} = g(y_{gc})$ and computed $z_{gc} = \frac{x_{gc} - \text{mean}(\boldsymbol{x}_g)}{\sqrt{\text{var}(\boldsymbol{x}_g)}}$, where mean and variance are the empirical mean and variance taken across cells.

In the overview figures (Fig. 24, 26, and 33), we use a gene-specific overdispersion estimate for all residuals-based transformations and for the delta method-based transformations which can handle a custom overdispersion; for *GLM PCA*, we use $\alpha = 0$, because these settings worked best for the respective transformations. The latent expression-based transformations and NewWave do not support custom overdispersion settings.

Conceptual differences For the visualization of the residual structure after adjusting for the varying size factors, we chose a control dataset of a homogeneous RNA solution encapsulated in droplets (Svensson et al., 2017). We filtered out RNAs that were all zero and plotted the first two principal components. Where applicable, we used the gene-specific overdispersion estimates. For visualizing the results of Sanity Distance, instead of the PCA, we used multi-dimensional scaling of the cell-by-cell distance matrix using R's *cmdscale* function. We calculated the canonical correlation using R's *cancor* function on the size factors and the first 10 dimensions from PCA and multi-dimensional scaling.

The plots of the mean-variance relation are based on the *10x human hematopoietic cell* dataset (Bulaeva et al., 2020). Where applicable, we used the gene specific overdispersion estimates. The panel of Sanity Distance shows the variance of samples drawn from a normal distribution using the inferred mean and standard deviation.

For the mouse lung dataset (Angelidis et al., 2019), we filtered out cells with extreme size factors ($0.1 s_{\text{median}} < s_c < 10 s_{\text{median}}$, where s_{median} is the median size factor). We also removed cells that did not pass the *scran* quality control criterion regarding the fraction of reads assigned to mitochondrial genes. To account for the fact that some transformations share information across genes, we applied all transformations to the 100 most highly expressed genes and three genes (Sftpc, Scgb1a1, Ear2) known to be differentially expressed in some cell types according to the assignment from the original publication.

Benchmark The benchmarks were executed using a custom work scheduler for slurm written in R on CentOS7 and R 4.1.2 with Bioconductor version 3.14. The set of R packages used in the benchmark with exact version information was stored using the *renv* package and is available from the GitHub repository.

k-NN identification and dimensionality reduction To calculate the PCA, we used the *irlba* package. To infer the k nearest neighbors, we used *annoy*, which implements an approximate nearest neighbor search algorithm. To calculate the t-SNEs, which we only used for visualization, we used the *Rtsne* package on data normalized with the shifted logarithm with a pseudo-count of 1.

Consistency Benchmark We downloaded ten single-cell datasets listed in the gene expression omnibus database (GEO) browser after searching for the term *mtx* on 2021-10-14. All datasets are listed in the Data Availability section. To measure the consistency of the transformations, we randomly assigned each gene to one of two groups and processed the two resulting data subsets separately. We calculated the consistency as the mean overlap of the k nearest neighbors for all cells.

Simulation Benchmark We used five frameworks to simulate single-cell counts in R: we ran *dyngen* (Cannoodt et al., 2021) using a consecutive bifurcating mode and the default parameters otherwise. We ran *muscat* (Crowell et al., 2020) with 4 clusters, a default of 30% differentially expressed genes with an average log-fold change of 2, and a decreasing relative fraction of log-fold changes per cluster. We ran *scDesign2* (Sun et al., 2021) with the *10x human hematopoietic cell* dataset as the reference input with a copula model and a Gamma-Poisson marginal distribution. We simulated the *Random Walk* by translating the Matlab code of Breda et al. (2021) to R and using the data by Baron et al. (2016) as a reference. For the *Linear Walk*, we adapted the Random Walk simulation and, instead of following a random walk for each branch, we interpolated the cells linearly between a random start and end point. For both benchmarks, we used a small non-zero overdispersion of $\alpha = 0.01$ to mimic real data.

With each simulation framework, we knew which cells were, in fact, the k nearest neighbors to each other. We calculated the overlap as the mean overlap of this ground truth with the inferred nearest neighbors on the simulated counts for all cells. Furthermore, we calculated the adjusted Rand index (ARI) and adjusted mutual information (AMI) by clustering the ground truth and the transformed values with the graph-based walktrap clustering algorithm from the *igraph* package.

Downsampling Benchmark We searched the literature for single-cell datasets with high sequencing depth and found five (one from mcSCRB, four from Smart-seq3) that had a sequencing depth of more than 50 000 UMIs per cell on average. We defined reliable nearest neighbors as the set of k nearest neighbors of a cell that were identified with all 22 transformations on the deeply

sequenced data (excluding the two negative controls). We used the *downsampleMatrix* function from the *scuttle* package to reduce the number of counts per cell to approximately 5 000, a typical value for 10x data. We considered only one setting for the overdispersion per transformation (instead of allowing multiple overdispersion settings for some transformations as in the other benchmarks). We ran all transformations, which supported the setting, with a gene-specific overdispersion estimate (except *GLM PCA* which performed better with an overdispersion fixed to 0). Finally, we computed the mean overlap between the k nearest neighbors identified on the downsampled data with the set of reliable nearest neighbors for all cells with more than one reliable nearest neighbor.

k-NN Overlap For all three benchmarks, we calculated overlaps between pairs of k nearest neighbor graphs. Denoting their #cell × #cell adjacency matrices (i.e., a matrix of zeros and ones, where an entry is is one if a cell d is among the k nearest neighbors of cell c) by N^1 and N^2, we defined their overlap as

$$\frac{1}{\#\text{cells}} \sum_{c,d=1}^{\#\text{cells}} N^1_{cd} N^2_{cd}. \tag{68}$$

Data availability All datasets used for this study are freely available. Table 2 and 3 list the details where to access the datasets.

Table 2 | Datasets used to illustrate the conceptual differences between transformations. This table is adapted from Ahlmann-Eltze and Huber (2022)

Confounding effect of varying sequencing depth on dimensionality reduction	Droplet encapsulated RNA	Chromium v1	Svensson et al. (2017), CalTech Data Repo entry 1264
Mean-variance relation	Human hematopoietic cells	Chromium v3	Bulaeva et al. (2020) GEO GSE130931
Effect of transformation on marker genes	Mouse lung	DropSeq	Angelidis et al. (2019), GEO GSE124872

Table 3 | Datasets used to benchmark the performance differences between transformations. This table is adapted from Ahlmann-Eltze and Huber (2022)

Consistency	Human hematopoietic cells	Chromium v3	Bulaeva et al. (2020) GEO GSE130931
	SUM149PT cell line	10x Genomics	No corresponding publication GEO GSE142647
	Human lung epithelium	Chromium v3	Kathiriya et al. (2022) GEO GSE150068
	Mouse pharyngeal mesoderm	Chromium v2	Nomaru et al. (2021) GEO GSE158941
	Human neural progenitor cells	Chromium v3	De Santis et al. (2021) GEO GSE163505
	Mouse mammary	Chromium v2	Pal et al. (2021) GEO GSE164017
	Mouse aorta	Chromium v3	Porritt et al. (2021) GEO GSE178765
	Bovine intervertebral discs (IVDs)	Chromium v3	Panebianco et al. (2021) GEO GSE179714
	Human T helper cells	Chromium 5' v1	Qian et al. (2021) GEO GSE179831
	Human T cells	Chromium 5' v1.1	Lu et al. (2021) GEO GSE184806
Simulation	Human pancreas	InDrops	Baron et al. (2016), scRNAseq BioC package
Downsampling	JM8 cells	mcSCRB-seq	Bagnoli et al. (2018), GEO GSE103568
	HEK cells	Smart-seq3	Hagemann-Jensen et al. (2020), ArrayExpress E-MTAB-8735
	Fibroblasts (1)	Smart-seq3	Hagemann-Jensen et al. (2020), ArrayExpress E-MTAB-8735
	Fibroblasts (2)	Smart-seq3	Larsson et al. (2021), ArrayExpress E-MTAB-10148

siRNA (KD)	Knockdown	Smart-seq3	Johnsson et al. (2022), Github Sandberg-lab

Code availability and reproducibility I developed an R package called *transformGamPoi* which implements the delta method-based variance-stabilizing transformations and simple implementation of the Pearson residuals. The package is hosted on Bioconductor at bioconductor.org/packages/transformGamPoi/.

The benchmarks were implemented using a custom-made workflow manager build around the SLURM high-performance computing system. All scripts to reproduce the benchmarks, their analysis and the corresponding figures are stored on github.com/const-ae/transformGamPoi-Paper and Zenodo.

Because the benchmark produced more results than I could visualize, I created a shiny app to let anyone explore the results interactively. The shiny app is hosted by EMBL at shiny-portal.embl.de/shinyapps/app/08_single-cell_transformation_benchmark.

3.7 Discussion

In this chapter, I presented my comparison of transformations for single-cell RNA-seq data. I explained how variance-stabilizing transformations are derived using the delta method. I then showed which properties of the transformed values differed between the approaches: for example, the log-transformed values had a higher correlation with the size factors than the Pearson residuals. On the flip side, I showed that the Pearson residuals could not stabilize the heteroskedasticity of a gene with a bimodal gene expression pattern.

The main part of the chapter focused on the comparison of the empirical performance of the transformation. I chose to measure the performance of the transformations with regard to their ability to uncover the local latent structure of the data as measured by the overlap of the k nearest neighbors of each cell with some (proxy for the) ground truth. Surprisingly, the simple shifted log transformation performed among the best and none of the mathematically more sophisticated (and computationally more demanding) transformations consistently outperformed it.

I used five different simulation frameworks to avoid dependence of the results on a particular simulation approach. However, two simulation frameworks produced data with a simplistic latent structure, whereas two others produced data with a more complex latent structure than real data. Only one approach (scDesign2) produced data that seemed realistic when plotting the cells structure with two-dimensional tSNE. I averaged the benchmark values across all five simulations; however, the results did not separate the transformations. This suggests that using more sophisticated simulation frameworks could help to assess the performance difference more accurately.

I used the deeply sequenced data from Smart-seq 3 experiments, to create a new type of benchmark that compares the structure inferred on the deeply sequenced data against the structure identified on downsampled data. The advantage of this approach is that it provides a realistic latent structure, unlike the simulation frameworks which sometimes only superficially resemble real single-cell data. A challenge of this approach is that only a few such datasets are publicly available, which meant that I had to revert to the datasets (except one) which analyzed fairly homogeneous systems like cell lines.

In summary, my comparison can help practitioners to choose an appropriate transformation for their single-cell data and can guide the development of novel transformations for single-cell RNA-seq counts or other data modalities.

4 lemur

In this chapter, I will present the third project of my PhD where I developed a new tool for the analysis of multi-condition single-cell data. I will begin by explaining the conceptual innovation of my model, before demonstrating its utility on a glioblastoma dataset. I, together with Wolfgang Huber, published this project as a preprint called "Analysis of multi-condition single-cell data with latent embedding multivariate regression" (Ahlmann-Eltze and Huber, 2023a). Note that this chapter describes version 0.0.19 of our R package *lemur*.

4.1 Multi-condition single-cell data

To establish the effect of an intervention or experimental condition on a complex tissue, we can use single-cell profiling of multiple samples. There are several experimental methods to generate such datasets which minimize the batch effects between samples, for example, through multiplexing. The analysis of such datasets is complicated by the fact that these datasets contain several sources of variation:

1. effect of the experimental condition,
2. sample-specific batch effects,
3. cell type heterogeneity,
4. plus interactions between all three sources of variation.

Computational method development for multi-condition single-cell data initially focused on addressing the sample-specific batch effects. Prominent tools for this task are mutual nearest neighbor integration (MNN) (Haghverdi et al., 2018), Harmony (Korsunsky et al., 2019), integration with Seurat (Butler et al., 2018), and scVI (Lopez et al., 2018). These tools try to find corresponding cell populations across samples and an *integrated* latent space where they locate close to each other. Luecken et al. (2022) compared their ability to integrate separate datasets of the same tissue.

After integration, the cell type heterogeneity is divided into discrete groups (using clustering and / or manual annotation) and the effect of the experimental condition is studied per group using differential expression analysis. Crowell et al. (2020) showed that summing the counts per group and condition is surprisingly effective for inferring reliable treatment effects. The resulting values are called *pseudobulks* because they are equivalent to sorting the data using flow cytometry and performing regular bulk RNA sequencing. More complex mixed-effects models require considerably more computational resources for little or no gain in performance.

4.1.1 Methods for multi-condition single-cell experiments

There are several experimental methods to tag the origin of a cell before mixing the cells and proceeding with the multiplexed sample preparation. There are two methods that are based on antibodies:

- Cell Hashing uses oligonucleotide-tagged antibodies against common surface proteins (Stoeckius et al., 2018).
- Gaublomme et al. (2019) used oligonucleotide-tagged antibodies against the nuclear pore complex to demultiplex nuclei.

A popular alternative is based on lipid or cholesterol-based anchors with an oligonucleotide label:

- MULTI-seq uses oligonucleotide-labeled lipid anchors that integrate into the cell or nuclear membrane (McGinnis et al., 2019).
- 10x Genomics 3' CellPlex uses similar oligonucleotide-labeled lipid anchors.

Besides these, there are additional demultiplexing approaches designed for specific circumstances:

- demuxlet is a computational tool using SNPs for deconvolution (Kang et al., 2018).
- MIX-seq uses SNP-based deconvolution (McFarland et al., 2020).
- ClickTag multiplexing attaches oligonucleotides to cellular proteins of fixed cells (Gehring et al., 2020).
- CellTag Indexing uses a lentiviral barcode delivery system (Guo et al., 2019)

4.2 Latent embedding multivariate regression

Here, I will describe our new method for the analysis of multi-condition single-cell data. I call this method *latent embedding multivariate regression* (LEMUR) because its innovation is that it performs regression on latent spaces. Unlike the existing analysis workflow which discretizes the latent space after integration into clusters, LEMUR operates on a continuous latent space.

The advantage of avoiding premature clustering is that choosing the optimal clustering parameters is an impossible task. For some genes, the clusters will be too large obscuring interesting variation. At the same time, the clusters will be too small for other genes, resulting in a loss of power to detect the interesting variation. Furthermore, many biological effects vary in gradual patterns, which is difficult to capture using clusters.

LEMUR performs regression on the latent subspaces that best interpolate the observations of each condition. Here, a condition is the unique combination of all experimental covariates. To make this concept more rigorous, I will now give a brief overview of differential geometry and geodesic regression.

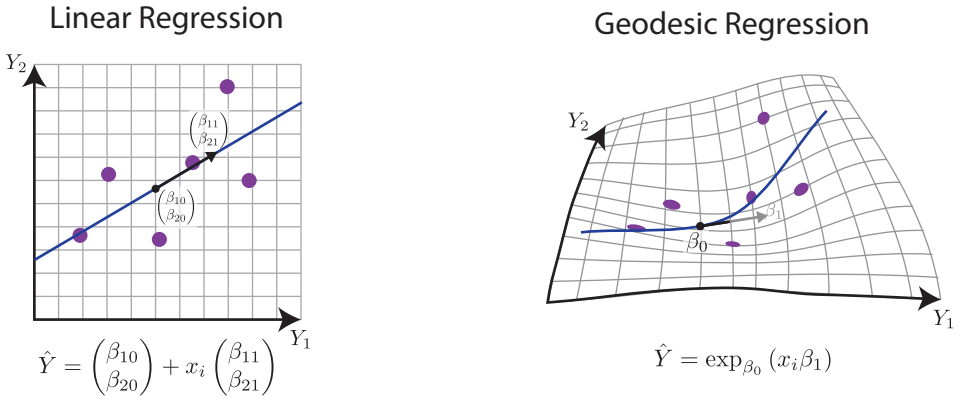


Figure 35 | Illustration of linear regression and geodesic regression. Regression finds a straight line (blue) on the surface of a smooth manifold (grey) that interpolates the observations (purple). In geodesic regression the manifold can have non-zero curvature, which makes any line look bent in the embedding space.

4.2.1 Geodesic regression

Geodesic regression generalizes the concept of linear regression to arbitrary manifolds (Fletcher, 2011). In linear regression, we find the coefficients B that connect the observations Y and the covariates X

$$Y = BX^T + \epsilon \tag{69}$$

where the columns of Y are the observations and the rows are the features. If we have only a single covariate, we can rewrite eq. (69) as

$$y_i = \beta_0 + \beta_1 x_i + \epsilon_i \tag{70}$$

where i indexes the observations and β_0 is the intercept and β_1 is the slope.

If y_i is a scalar, it is often plotted against the covariate x_i and β_0 and β_1 define the intercept and slope of a line. If y_i is two-dimensional and x_i is still a scalar, the regression line is parameterized by β_1. The line can be understood as an object traveling with constant speed in direction β_1 and β_0 is the location at which the object is at time $x = 0$.

Generalizing the idea of an object moving with constant velocity on non-Euclidean surfaces leads to geodesic regression (Fig. 35). Here, the y_i are points on a Riemannian manifold $\mathcal{M}$. The intercept β_0 is called the *base point* and β_1 is a tangent vector at the base point. The geodesic equivalent of eq. (70) is

$$\begin{aligned}\hat{y}_i &= \mathrm{Exp}_{\beta_0}(x_i\beta_1)\\ y_i &= \mathrm{Exp}_{\hat{y}_i}(\epsilon_i).\end{aligned} \tag{71}$$

Here, Exp is the exponential map that takes a base point $p \in \mathcal{M}$ and a tangent vector $v \in \mathcal{T}_p\mathcal{M}$ and returns a new point on the manifold $p' \in \mathcal{M}$. For Euclidean space, the exponential map is just addition, i.e., $\mathrm{Exp}_p(v) = p + v$ recovering the simple linear regression from eq. (70).

The name *exponential map* comes from the fact that for some manifolds (i.e., Lie groups) the exponential map coincides with the matrix exponential

$$\exp(X) = I + X + \frac{1}{2}X^2 + \frac{1}{6}X^3 + \cdots, \tag{72}$$

which has the name because this is the same Taylor series as the familiar exponential function on scalars

$$\exp(x) = 1 + x + \frac{1}{2}x^2 + \frac{1}{6}x^3 + \cdots. \tag{73}$$

In linear regression, we optimize β_0 and β_1 by minimizing the squared residuals ϵ^2. In geodesic regression, we proceed similarly and minimize

$$\underset{\beta_0,\beta_1}{\arg\min} \sum_i^N \epsilon_i^2 = \sum_i^N \mathrm{Log}(y_i, \hat{y}_i)^2, \tag{74}$$

where Log is the logarithmic map and is defined as the inverse of the exponential map. It takes two points $p, p' \in M$ and returns a vector from the tangent space $v \in \mathcal{T}_p\mathcal{M}$.

Solving eq. (74) is not straightforward. Fletcher (2011) and Rentmeesters (2011) have proposed algorithms for optimizing β_0 and β_1 using a gradient descent approach. Fletcher (2011) explained that calculating the gradient requires computing the derivative of the distance function (i.e., the squared logarithmic map) and the exponential map. The derivative of the exponential map with respect to a single tangent vector is the Jacobi field along the geodesic. Rentmeesters (2011) showed that the Jacobi fields can be calculated efficiently for symmetric Riemannian manifolds (e.g., spheres, symmetric positive definite matrices, special orthogonal matrices, and Grassmann manifolds).

The biggest limitation of Jacobi field-based optimization strategies as proposed by Fletcher (2011) and Rentmeesters (2011) is their restriction to univariable regression problems. For linear regression (model (70)) it is natural to include multiple covariates (called multivariable regression). Multivariable geodesic regression extends model (71) by summing over multiple tangent vectors

$$\hat{y}_i = \mathrm{Exp}_o\left(\sum_k^K X_{ik} v_k\right) \tag{75}$$

where $o \in \mathcal{M}$ is the base point, there are K covariates, and all coefficients are tangent vectors $v_k \in \mathcal{T}_o\mathcal{M}$. The sum of two tangent vectors is again a tangent vector because the tangent vectors are a vector space.

Kim et al. (2014) use a trick to find the tangent vectors that approximate the solution of eq. (75). For a fixed basis point o, they project all y_i into the log space of o

$$\tilde{y}_i = \mathrm{Log}(o, y_i) \tag{76}$$

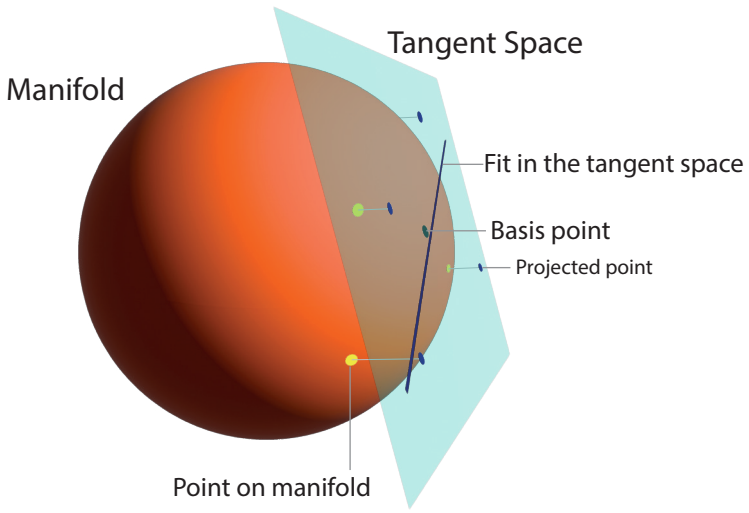


Figure 36 | Schematic of the log-space approximation. Points on the manifold (yellow) are projected into the tangent space of a base point (blue, translucent). Fitting a regression in the tangent space using the projected points approximates fitting a regression on the manifold with the original points.

and then solve the regression problem on $\tilde{y}_i$

$$\arg\min_{v_k} \sum_{i}^{N} ||\tilde{y}_i - \sum_{k} X_{ik} v_k||^2. \tag{77}$$

Eq. (77) is a linear regression problem I solve using `R`'s built-in functions.

Figure 36 illustrates the principle for a 2D-sphere. The accuracy of the approximation depends on the base point. It needs to be at the center of the observations and the spread of the points should be small to minimize distortions of the projection of the curved manifold onto the non-curved tangent space.

The concept of projecting manifold-valued data into the tangent space and solving the optimization problem there generalizes to arbitrary machine learning methods. Despite obtaining only approximate solutions, this approach is appealing because one can reuse existing machine learning algorithms for any manifold for which we have an implementation of the exponential map and exponential logarithm.

4.2.2 Matrix decompositions

Principal component analysis (PCA) is a matrix decomposition of a matrix Y into two matrices

$$Y = RZ + \gamma. \tag{78}$$

Y is a rectangular matrix with n rows and m columns, R is an orthonormal matrix with $\min(n, m)$ columns and n rows. γ is a vector and centers the rows. I overload the + operator for matrix-vector summation and implicitly assume that γ is repeated m times to match the dimensions of Y.

PCA is closely related to two other matrix decompositions: singular value decomposition and eigenvalue decomposition. The singular value decomposition (SVD) decomposes a rectangular matrix Y into three components

$$Y = U \operatorname{diag}(\boldsymbol{\sigma}) V^T. \tag{79}$$

The vector $\boldsymbol{\sigma}$ contains the singular values and U and V are called the left and right singular values of Y, respectively.

The eigendecomposition takes a square matrix A and forms

$$A = Q \operatorname{diag}(\boldsymbol{\lambda}) Q^{-1}. \tag{80}$$

The vector $\boldsymbol{\lambda}$ contains the eigenvalues of A and Q is a square matrix with the eigenvectors in the columns. The fundamental property of eigenvalue and eigenvectors pairs is that they satisfy

$$AQ_{:i} = \lambda_i Q_{:i} \tag{81}$$

where $Q_{:i}$ is one of the eigenvectors.

If $A = YY^T$ is the covariance matrix of Y, then the eigendecomposition and the singular value decomposition are related through $\boldsymbol{\sigma}^2 = \boldsymbol{\lambda}$. The PCA decomposition is a slightly simplified version of the singular value decomposition where $Z = \boldsymbol{\sigma} V^T$ and where the γ parameter explicitly acknowledges that the interpretability of the decompositions benefits from centering the observations.

A popular application for PCA is as a dimension reduction method (Figure 37A). The idea is to sort the rows of Z by variance and only consider the p most variable rows: i.e., $R \in \{\mathbb{R}^{n \times p} : R^T R = I\}$ and $Z \in \mathbb{R}^{p \times m}$. Consequently, eq. (78) is no longer an exact equality, but the decomposition becomes an approximation

$$Y \approx RZ + \gamma \tag{82}$$

which is optimal with respect to the least squares error.

The main statistical innovation of the LEMUR model is that I combine the ideas of PCA with regression to allow PCA to adjust for known covariates.

4.2.3 Multi-condition PCA

I call the new statistical decomposition that I have developed *multi-condition PCA*. It extends the regular PCA by accounting for known experimental covariates (Figure 37). In the simplest case, these could be a treated and a control condition and multi-condition PCA is equivalent to

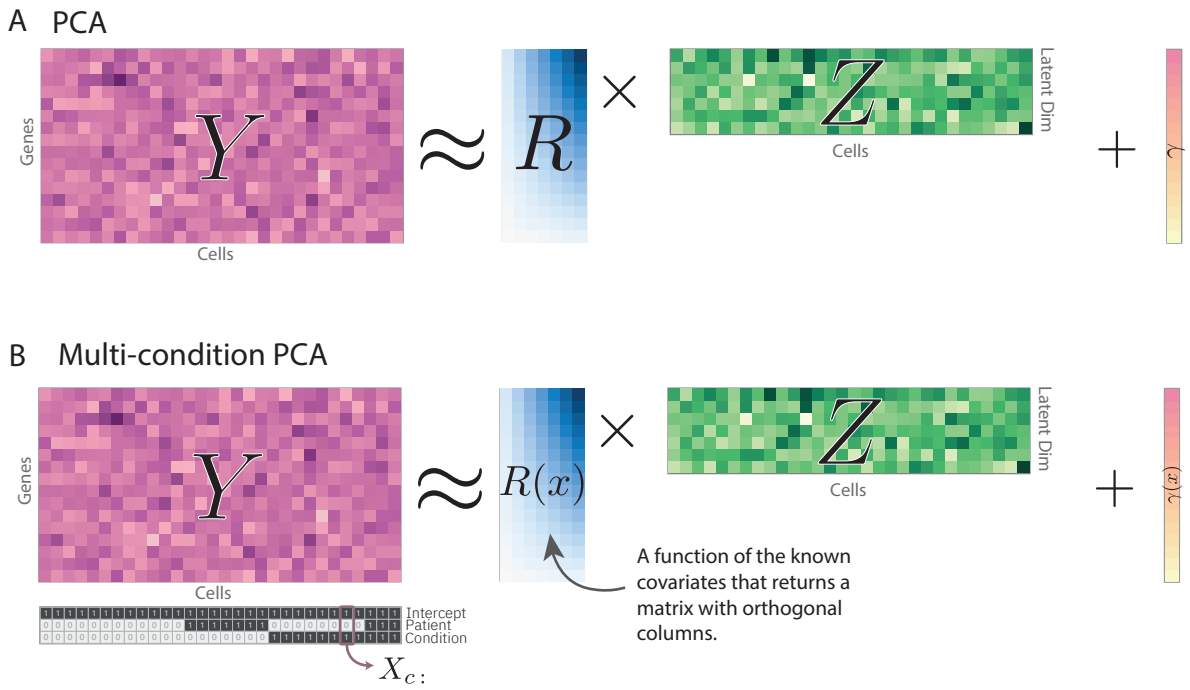


Figure 37 | PCA and multi-condition PCA. (A) Dimensionality reduction with principal component analysis of a matrix Y into an orthonormal matrix R and the embedding Z. The rank of the decomposition is smaller than the rank of Y which makes the decomposition approximative. (B) Multi-condition PCA extends PCA by treating R as a function of known covariates that returns orthonormal matrices.
The data for this figure was generated by me and adapted from Ahlmann-Eltze and Huber (2023b).

performing regular PCA on each condition separately. However, for more complex experimental designs, such as treatment gradients or incomplete nesting of the experimental conditions, multi-condition PCA provides a powerful new framework for disentangling the effects of the treatment and the latent variation.

The multi-condition PCA decomposition is

$$Y_{:c} \approx R(X_{c:})Z_{:c} + \gamma(X_{c:}) \tag{83}$$

where c indexes the cells and R is a function of the rows of the design matrix X that returns an orthonormal matrix: i.e., $R : \mathbb{R}^K \to \{A \in \mathbb{R}^{G \times P} : A^T A = I\}$. In the simplest case $R(x) = A$ in which case eq. (83) reduces to regular PCA (eq. (82)).

To account for the known covariates, I parameterize the R using the exponential map on the Grassmann manifold

$$R(x) = \mathrm{Exp}_o\left(\sum_{k=1}^{K} x_k V_k\right). \tag{84}$$

The basis point o is the reduced space for PCA on all data. $\{V_1, \cdots, V_K\}$ are elements of the tangent space of a Grassmann manifold and the coefficients which are optimized for during the

multi-condition PCA fit.

A Grassmann manifold is the set of all $G \times P$-dimensional subspaces. I parameterize the elements of a $G \times P$ Grassmann manifold using orthonormal matrices $\{\text{span}(A) : A \in \mathbb{R}^{G \times P}, A^T A = I\}$. Multiple orthonormal matrices that span the same spaces are thus considered identical because they represent the same element of the Grassmann manifold. The advantage of the orthonormal matrix representation is that it saves memory and is directly amenable to other linear algebra operations.

The exponential map of the Grassmann manifold is

$$\text{Exp}_p(x) = pV \,\text{diag}(\cos(\sigma))V^T + U \,\text{diag}(\sin(\sigma))V^T, \tag{85}$$

where p is the base point and $x = U \,\text{diag}(\sigma)V^T$ is a singular value decomposition of the tangent vector x (Bendokat et al., 2020).

I find the coefficients $\{V_1, \cdots, V_K\}$ using the log-space trick by Kim et al. (2014). However, to apply equation (77), I need to first find the points on the Grassmann manifold. For each condition (i.e., the set of identical rows of the design matrix), I collect the corresponding observations and perform regular reduced-rank principal component analysis. With the respective orthonormal matrices spanning the latent space that best approximates the observations in each condition, I project them into the log space of a base point and run classical linear regression to find the coefficients V_k.

In addition to the parameters of the Grassmann-exponential map, I also need to find the parameters of the offset function γ. In my current implementation, I support two settings.

1. $\gamma(x) = 0$, in which case I do not attempt to center the data, which leads to a large first component of Z.

2. $\gamma(x) = \sum_k x_k \Gamma_k$, i.e., regular linear regression which I solve independently of the optimization of the Grasmann-exponential map coefficients.

Lastly, I rotate the embedding so that the first row of the embedding explains the maximum amount of variation, the second row explains the second most variation and so on. This process mirrors the resolution of the rotational degeneracy in PCA. I set

$$\begin{aligned} Z' &= V^T \,\text{diag}(\sigma) \\ o' &= oU \\ V'_k &= V_k U \end{aligned} \tag{86}$$

where $Z = U \,\text{diag}(\sigma)V^T$ is the singular value decomposition of the unsorted embedding matrix.

In algorithm 1, I provide the pseudo-code for the multi-condition PCA algorithm. In listing 1, I have implemented the pseudo-code in `R` to provide a functional code example.

Algorithm 1 Multi-condition PCA algorithm

Require: $Y \in \mathbb{R}^{G \times C}$ ▷ The input data
Require: $X \in \mathbb{R}^{C \times K}$ ▷ The design matrix
Require: $n \in 0 < \mathbb{Z}^{+} \leq \min(G, C)$ ▷ The latent space dimension

function MULTICONDITIONPCA(Y, X, n)
$\Gamma \leftarrow YX(X^T X)^{-1}$
$Y \leftarrow Y - \Gamma X^T$
$o \leftarrow$ GETSUBSPACE(PCA(Y, n))
for all $e \in \{\{i | X_{i_1:} = X_{i_2:} = \cdots\}\}$ **do**
$A_e \leftarrow$ LOG(o, GETSUBSPACE(PCA($Y_{:e}$, n)))
end for
$V \leftarrow$ WEIGHTEDLINEARREGRESSION($\{A_e\}$, X_e, $\{\#e\}$)
$V \leftarrow$ RESHAPE(V, {G,n,K})
for all $e \in \{\{i | X_{i_1:} = X_{i_2:} = \cdots\}\}$ **do**
$Z_{:e} \leftarrow \mathrm{Exp}_o \left(\sum_k^K X_{ek} V_{::k}\right)^T Y_{:e}$
end for
$U, \mathrm{diag}(\sigma), V^T \leftarrow \mathrm{SVD}(Z)$
$o \leftarrow oU$
for $k = 1, \ldots, K$ **do**
$V_k \leftarrow V_k U$
end for
$Z \leftarrow V^T \mathrm{diag}(\sigma)$
return Z, o, $\{V_k\}$, Γ
end function

Listing 1 | Simplified multi-condition PCA implementation in R.

```r
grassmann_log <- function(p, q){
  n <- nrow(p)
  k <- ncol(p)
  z <- t(q) %*% p
  At <- t(q) - z %*% t(p)
  Bt <- lm.fit(z, At)$coefficients
  svd <- svd(t(Bt), k, k)
  svd$u %*% diag(atan(svd$d), nrow = k) %*% t(svd$v)
}

grassmann_map <- function(x, base_point){
  svd <- svd(x)
  base_point %*% svd$v %*% diag(cos(svd$d), nrow = length(svd$d)) %*% t(svd$v) +
    svd$u %*% diag(sin(svd$d), nrow = length(svd$d)) %*% t(svd$v)
}

#' @param Y is a matrix with features in the rows and observations in the columns
#' @param design_matrix a matrix with one row per observation coding the covariates
#'
#' @return a list with the embedding, the base_point, the coefficients for the
#'   Grassmann exponential map, and the coefficients for the linear regression.
multicondition_pca <- function(Y, design_matrix, n_embedding = 15){
  # Center observations with linear regressiong
  fit <- lm.fit(design_matrix, t(as.matrix(Y)))
  Y <- t(residuals(fit))
```

```
  # Find base point with PCA over all data points
  base_point <- irlba::prcomp_irlba(t(Y), n = n_embedding, center = FALSE)$rotation

  # Find the subspace of each condition
  red_design <- unique(design_matrix)
  cond_ids <- vctrs::vec_group_id(design_matrix)
  cond_weights <- c(table(cond_ids))
  cond_subspaces <- lapply(seq_len(nrow(red_design)), \(cond){
    irlba::prcomp_irlba(t(Y[,cond_ids == cond,drop=FALSE]),
                        n = n_embedding, center = FALSE)$rotation
  })

  # Find coefficients of Grassmann exponential map
  log_points <- do.call(cbind, lapply(cond_subspaces, \(subspace){
    as.vector(grassmann_log(base_point, subspace))
  }))
  coefficients <- t(lm.wfit(red_design, t(log_points), w = cond_weights)$coefficients)
  coefficients <- array(coefficients, dim = c(nrow(Y), n_embedding, ncol(design_matrix)))

  # Project the points on the embedding
  embedding <- matrix(NA, nrow = n_embedding, ncol = ncol(Y))
  for(cond in seq_len(nrow(red_design))){
    tang_vec <- matrix(0, nrow = nrow(Y), ncol = n_embedding)
    for(k in seq_len(ncol(red_design))){
      tang_vec <- tang_vec + red_design[cond, k] * coefficients[,,k]
    }
    embedding[,cond_ids == cond] <- t(grassmann_map(tang_vec, base_point)) %c*%
                                                              Y[,cond_ids == cond]
  }

  # Order axes by variance
  svd_emb <- svd(embedding)
  base_point <- base_point %c*% svd_emb$u
  for(k in seq_len(ncol(design_matrix))){
    coefficients[,,k] <- coefficients[,,k] %c*% svd_emb$u
  }
  embedding <- t(svd_emb$v) * svd_emb$d

  # Return values
  list(embedding = embedding, base_point = base_point,
       coefficients = coefficients, linear_coefficients = t(fit$coefficients))
}
```

4.2.4 Alignment

Multi-condition PCA (model (83)) removes condition differences that are captured by the latent space they occupy. However, multi-condition PCA cannot account for changes in the relative position of the cell populations. I will thus add an additional function to account for those changes

$$Y_{:c} \approx R(X_{c:})S(X_{c:})Z'_{:c} \tag{87}$$

where S is a function that takes a row from the design matrix as input and returns a square matrix ($S : \mathbb{R}^K \to \mathbb{R}^{P \times P}$). To ensure that the alignment term S does not affect the overall approximation,

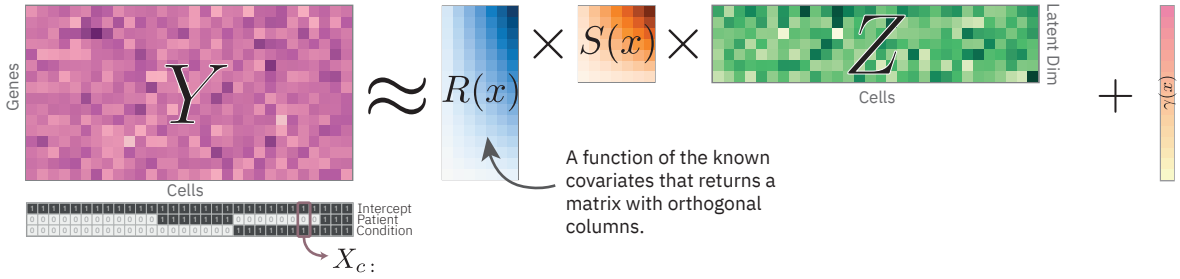


Figure 38 | Extension of the multi-condition PCA model with an additional alignment term. The data for this figure was generated by me and adapted from Ahlmann-Eltze and Huber (2023b).

I define

$$R(X_{c:})Z = R(X_{c:})S(X_{c:}) \underbrace{S(X_{c:})^{-1} Z_{:c}}_{=Z'_{:c}} . \tag{88}$$

I refer to this model as *latent embedding multivariate regression* (LEMUR).

There are several choices for parameterizing S. Below I list five options in order of increasing flexibility:

1. $S(x) = I$: No alignment is performed and the LEMUR model reverts to the multi-condition PCA model described in eq. (83).
2. $S(x) = \text{diag}\left(\sum_k x_k \boldsymbol{v}_k\right)$: This enables the model to shrink each dimension independently.
3. $S(x) = \text{Exp}^{(\text{Rot})}\left(\sum_k x_k V_k\right)$: Using the exponential map of the rotation manifold (also known as special orthogonal group ($SO(n)$)) keeps the transformation rigid, but allows more precise alignment of matching cell populations.
4. $S(x) = \text{Exp}^{(\text{SPD})}\left(\sum_k x_k V_k\right)$: Using the exponential map of the symmetric positive definite manifold (SPD) allows the transformation to shrink and extend the data along an orthogonal coordinate system (which, unlike option (2), does not have to be axis parallel).
5. $S(x) = \text{Exp}^{(\text{Rot})}\left(\sum_k x_k V_k^{(1)}\right) \text{Exp}^{(\text{SPD})}\left(\sum_k x_k V_k^{(2)}\right)$: Allowing both rotations and symmetric positive definite transformations enables all affine transformations except for reflections. This parametrization is inspired by the polar decomposition that converts any square matrix into a special orthogonal (with determinant ± 1) and a symmetric positive semi-definite matrix.
6. $S(x) = \sum_k x_k V_k^{(1)}$ allows for any affine transformation and is thus the most generic linear transformation.

As I define $S(x)$ and Z' such that the overall approximation of the observations does not change, optimizing the parameters R and S is separable. After optimizing the parameters of R by

matching the latent spaces of the conditions, I optimize the parameters of S so that corresponding cell populations across conditions have similar values Z'.

I assume the user defines which cell populations correspond or uses an automated method to identify corresponding populations (for example, mutual nearest neighbors or harmony's maximum diversity clustering (Korsunsky et al., 2019; Haghverdi et al., 2018)). I expect that this preference is encoded in a list of sets, each containing the indices of cells that are considered similar. I denote the e-th set ($e = 1, \ldots, E$) as $\mathbb{E}_e$. If the user specifies one or more sets of matching cells, S is obtained as the solution to

$$\underset{S \in \mathcal{S}(x)}{\arg\min} \sum_{e=1}^{E} \sum_{c_1, c_2 \in \mathbb{E}_e} (S(X_{c_1:})^{-1} Z_{:c_1} - S(X_{c_2:})^{-1} Z_{:c_2})^2, \tag{89}$$

to minimize the pairwise distances between the cells of each similarity set. I can simplify the expression and, instead of optimizing all pairwise distances, minimize the distance to the center of each similarity set

$$\underset{S \in \mathcal{S}(x)}{\arg\min} \sum_{e=1}^{E} \sum_{c \in \mathbb{E}_e} (M_e - S(X_{c:})^{-1} Z_{:c})^2, \tag{90}$$

and

$$M_e = \frac{1}{\#\mathbb{E}_e} \sum_{c \in \mathbb{E}_e} S(X_{c:}) Z_{:c} \tag{91}$$

is the mean of the cells in that similarity set. The structure of eq. (90) is amenable to an iterative solution where I optimize M_e and the parameters of S in turns. I start with

$$M_e = \frac{1}{\#\mathbb{E}_e} \sum_{c \in \mathbb{E}_e} Z_{:c} \tag{92}$$

because I can calculate it directly.

4.2.5 Procrustes Problems

Problem (90) is a further generalization of the (generalized) Procrustes problem (Gower, 1975). Procrustes problem tries to find the solution to

$$\underset{S}{\arg\min} \sum (Y - SZ)^2 \tag{93}$$

and depending on the domain of S the problem is called orthogonal, semi-definite, or affine Procrustes problem (Schönemann, 1966; Gillis and Sharma, 2018).

Algorithm 2 Simon Segert's algorithm to find the minimal rotation for a rank-deficient orthogonal Procrustes problem with some modifications by me.

Require: $Y \in \mathbb{R}^{P \times C}$ ▷ The fixed point cloud
Require: $Z \in \mathbb{R}^{P \times C}$ ▷ The transformed point cloud
  function ORTHOGONAL PROCRUSTES(Y, Z)
    if P = 1 **then**
      return 1
    else if P = 2 **then**
      $a \leftarrow \sum Z \odot Y$
      $b \leftarrow \sum (Y_{1:} \odot Z_{2:} - Y_{2:} \odot Z_{1:})$
      $\theta \leftarrow \tan^{-1}(a, b) - \pi/2$
      return $\begin{pmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{pmatrix}$
    else
      $Q \leftarrow ZY^T$
      U,σ,$V \leftarrow$ SVD(Q)
      if rank$(Q) < P$ **then**
        $\begin{bmatrix} V^{(1)} & V^{(2)} \end{bmatrix} \leftarrow \begin{bmatrix} V_{:,\sigma>\epsilon} & V_{:,\sigma\leq\epsilon} \end{bmatrix}$
        $\begin{bmatrix} U^{(1)} & U^{(2)} \end{bmatrix} \leftarrow \begin{bmatrix} U_{:,\sigma>\epsilon} & U_{:,\sigma\leq\epsilon} \end{bmatrix}$
        $S^{(\text{sub})} \leftarrow$ ORTHOGONAL PROCRUSTES($V^{(2)}$, $U^{(2)}$)
        return $V \begin{bmatrix} U^{(1)} & S^{(\text{sub})} U^{(2)} \end{bmatrix}^T$
      else
        return VU^T
      end if
    end if
  end function

Orthogonal Procrustes problem The solution to the orthogonal Procrustes problem (eq. (93) with $S \in SO(n)$) can be solved using a singular value decomposition of $ZY^T = U \operatorname{diag}(\sigma) V^T$

$$\underset{S \in O(n)}{\arg\min} \sum (Y - SZ)^2 = VU^T. \tag{94}$$

To ensure that S is a rotation ($S \in SO(n)$), I need to check if $\det(VU^T)$ equals 1 or -1. Thus, I calculate S as

$$\underset{S \in SO(n)}{\arg\min} \sum (Y - SZ)^2 = V \operatorname{diag}([1, \ldots, 1, \det(VU^T)]) U^T. \tag{95}$$

One challenge with the orthogonal optimization occurs if ZY^T is not full rank which means that the output of eq. (95) is not unique. The problem is that S might contain additional rotation. Because I was stuck with this problem, I posted the problem on the Cross Validated forum and Segert (2022) suggested a solution using a recursive algorithm (see algorithm 2).

Semi-definite Procrustes problem A solution to the symmetric positive semi-definite Procrustes problem (eq. (93) with $S \in \text{SPD}(n)$, i.e., $\{S \in \mathbb{R}^{n \times n} : S^T = S \wedge vSv^T \leq 0 \text{ for all } p \in \mathbb{R}^n \backslash \{0\}\}$) was developed by (Gillis and Sharma, 2018). For initialization, I use the analytical method from Jingjing et al. (2019) or the initializations suggested by Gillis and Sharma (2018).

Affine Procrustes problem The solution of the affine Procruste problem (eq. (93) with $S \in \mathbb{R}^{P \times P}$) is the well-known least squares solution

$$\underset{S \in \mathbb{R}^{P \times P}}{\arg\min} \sum (Y - SZ)^2 = YZ^{\dagger}, \tag{96}$$

where $Z^{\dagger}$ is the Moore-Penrose generalized inverse.

Generalized Procrustes analysis Generalized Procrustes analysis is concerned with matching $N > 2$ point clouds (Gower, 1975, 2010)

$$\underset{\{S_i\}}{\arg\min} \sum_{i}^{N} \sum_{j}^{N} (S_i Z_i - S_j Z_j)^2. \tag{97}$$

This problem becomes substantially simpler, if we define a group average

$$M = \frac{1}{N} \sum_{i=1}^{N} (S_i Z_i) \tag{98}$$

because plugging M into eq. (97) becomes

$$\underset{\{S_i\}}{\arg\min} \sum_{i}^{N} (M - S_j Z_j)^2. \tag{99}$$

which can be solved for each S_i using the solution to the pair-wise Procrustes problem discussed above. Instead of an iterative algorithm, Bai and Bartoli (2022) suggested a closed-form solution based on the analysis of the eigenvalues of each Z_i.

4.2.6 Procrustes regression

Equation (89) is a generalization of the *generalized Procrustes analysis*, where instead of optimizing the elements of a manifold directly, I optimize the coefficients which come from a tangent space of some manifold

$$\underset{S \in \mathcal{S}(x)}{\arg\min} \sum_{i}^{N} \sum_{j}^{N} (S_i(X_{i:})Z_i - S_i(X_{j:})Z_j)^2 \tag{100}$$

where $X \in \mathbb{R}^{N \times K}$ is a design matrix and the function S is

$$S(x) = \mathrm{Exp}_o \left(\sum_{k=1}^{K} x_k V_k \right) \tag{101}$$

and $o \in \mathcal{M}$ is the base point and the coefficients are from the tangent space of some manifold ($V_k \in \mathcal{T}_o \mathcal{M}$).

For a fixed mean M

$$\underset{S \in \mathcal{S}(x)}{\arg\min} \sum_{i}^{N} (M - S_i(X_{j:}) Z_j)^2 \tag{102}$$

we can use the log-space trick by Kim et al. (2014) (eq. (77)) to solve eq. (102) approximatly.

Procrustes regression for affine transformations If $S(x) = \sum_k x_k V_k$ for $V_k \in \mathbb{R}^{P \times P}$, the Procrustes regression can be solved analytically. The trick is to rewrite the matrix product of summed coefficients with Z as a Hadamard (in-place) product

$$\begin{bmatrix} | & & | \\ (\sum_k X_{1k} V_k)\, Z_{:1} & \cdots & (\sum_k X_{Ck} V_k)\, Z_{:C} \\ | & & | \end{bmatrix} = \begin{bmatrix} V_{::1} & \cdots & V_{::K} \end{bmatrix} ((X \otimes \mathbf{1}_G)^T \odot (\mathbf{1}_K \otimes Z)), \tag{103}$$

Where $\otimes$ is the Kronecker product, $\odot$ is the Hadamard product and $\mathbf{1}_G$ is a vector of ones with length G. Accordingly, eq. (90) with $S^{-1}(x) = \sum_{k=1}^{K} x_k V_{::k}$ simplifies to

$$\begin{aligned} & \underset{\{V_k\}}{\arg\min} \sum_{e=1}^{E} \sum_{c \in \mathbb{E}_e} (M_e - S(X_{c:})^{-1} Z_{:c})^2 \\ \Leftrightarrow \quad & \underset{\{V_k\}}{\arg\min} \left(\underbrace{\begin{bmatrix} M_1 & \cdots & M_E \end{bmatrix}}_{=M'} - \begin{bmatrix} V_{::1} & \cdots & V_{::K} \end{bmatrix} \underbrace{((X_{\mathbb{E}:} \otimes \mathbf{1}_G)^T \odot (\mathbf{1}_K \otimes Z_{:\mathbb{E}}))}_{=X'} \right)^2 \end{aligned} \tag{104}$$

and thus $\begin{bmatrix} V_{::1} & \cdots & V_{::K} \end{bmatrix} = M' X'^{\dagger}$.

Procrustes regression for rotation and stretchings Solving eq. (90) with $S(x) = \mathrm{Exp}^{(\mathcal{M})} (\sum_k x_k V_k)$ for $\mathcal{M} = \mathrm{SO}(n)$ or $\mathcal{M} = \mathrm{SPD}(n)$ follows the same principles as solving the Grassmann manifold regression problem (eq. (83)). For each condition, I solve the orthogonal or semi-definite Procrustes problem. Then, I project the resulting points from the manifold into the log space of a base point and find the regression coefficients using weighted linear regression in the tangent space.

I set the base point to the identity matrix I, because then taking the inverse corresponds to flipping the sign of the coefficients

$$\begin{aligned} \mathrm{Exp}_I^{(\mathrm{SO}(n))}(V)^{-1} &= \mathrm{Exp}_I^{(\mathrm{SO}(n))}(-V) \\ \mathrm{Exp}_I^{(\mathrm{SPD}(n))}(V)^{-1} &= \mathrm{Exp}_I^{(\mathrm{SPD}(n))}(-V). \end{aligned} \tag{105}$$

Summary Figure 39 summarizes the functionality of LEMUR: it is a matrix decomposition algorithm that considers known covariates (Fig. 39A). The data are approximated by a latent space for each condition (Fig. 39B) which are connected through some parametric rotation of a base space (Fig. 39C). To ensure that corresponding cell populations appear at the same position in the latent embedding, the matrix S aligns their position (Fig. 39D).

4.3 Cluster-free differential expression

LEMUR infers a parametric model of multi-condition single-cell data. I use this model to predict the expression of unobserved cell states and calculate the expected change between two conditions to learn which cells and genes show differential expression. The function to predict the expression of a cell at position z and with the covariates x is

$$f(x, z) = R(x)S(x)z + \gamma(x) \tag{106}$$

where R and S use the coefficients inferred in the previous steps.

The differential expression for a cell at position $Z_{c:}$ between condition $x^{(A)}$ and condition $x^{(B)}$ is

$$\Delta_{:c} = f(x^{(A)}, Z_{c:}) - f(x^{(A)}, Z_{c:}). \tag{107}$$

4.3.1 Differential expression neighborhoods

I calculate the differential expression for all cells and genes $\Delta \in \mathbb{R}^{G \times C}$. For some genes, all cells will have a large Δ_g, whereas for others, none of the cells will have a large Δ_g. These cases are not of primary interest because they can also be found with bulk RNA-seq. The promise of single-cell is the ability to find changes restricted to a subpopulation of cells. The challenge is to find the optimal resolution to find the cell neighborhoods with a distinct differential expression pattern.

I use the matrix of expected differential expression values Δ to guide the inference of differential expression neighborhoods. I identify a direction in the low-dimensional embedding space that best captures the trend along which the differential expression values Δ_g vary. I currently support three approaches for choosing the direction per gene:

1. I generate a set of random directions in the embedding space by sampling pairs of cells and calculating the difference: $v = Z_{:2} - Z_{:1}$. Then, for each gene, I choose the vector with

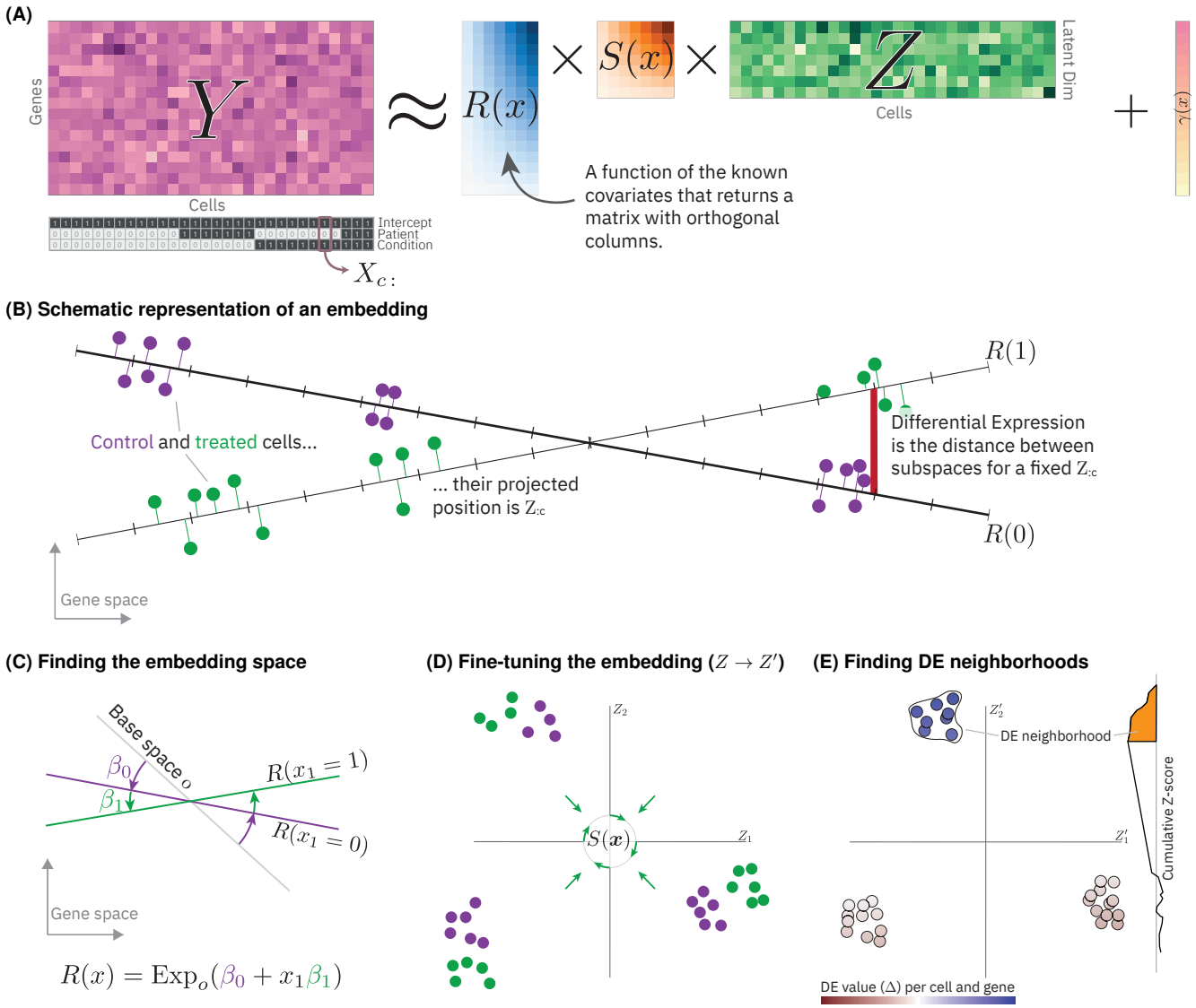


Figure 39 | Overview of the LEMUR method. (A) Schematic of the matrix decomposition. (B) Schematic of LEMUR's data approximation. Data points from two conditions (green and purple) are approximated by a condition-specific latent space (black). The points are projected onto the respective latent space. (C) The latent spaces are produced by the function $R(x)$. The coefficients of the function (here called β) determine the location of the latent space relative to the base point o. (D) The function S aligns corresponding cell populations. (E) The LEMUR model can make counterfactual predictions for the gene expression of a cell if had been in one or another condition. The difference is the differential expression value. I provide a method to find neighborhoods in the latent space with consistent differential expression.
The data for this figure was generated by me and adapted from Ahlmann-Eltze and Huber (2023b).

the highest correlation of Z projected on v with the differential expression values Δ_g.

2. A slightly simpler approach is to use axis parallel vectors $v = e_p$ and then choose the axis with the highest correlation.

3. Alternatively, the difference between the subspaces for the respective conditions $v =$

$(R(x^{(A)}) - R(x^{(B)}))_g$ points for each gene in the direction along which the differential expression changes most.

I define the differential expression neighborhood as the set of cells on one side of a cut-off point p along the respective vector v for a gene. I have implemented two methods to find the cut-off point p.

The first option is to sort the cells by their position along the vector v. I start with the cell at one end of the projection, calculate the z-score, then go to the next cell and calculate the z-score for both cells, and so on until I reach the cell at the other end of the projection. Ultimately, I have two vectors with z-scores for each cell because I calculate the running z-score from both extremes. The cut-off point p is the cell in combination with the preceding cells had the largest absolute z-score. This approach is illustrated in Figure 39E.

The advantage of the z-score approach is that it is fast to calculate because the z-score can be calculated with an online algorithm. The z-score is the mean divided by the standard deviation. The online algorithm for the mean is called cumulative average. Let $\mu^{(t)}$ be the mean of the first t cells and $y^{(t+1)}$ the next value to add. Then, the next mean is

$$\mu^{(t+1)} = \mu^{(t)} + \frac{y^{(t+1)} - \mu^{(t)}}{t+1}. \tag{108}$$

Similarly, the next standard deviation $\sigma^{(t+1)}$ can be calculated using a variation of Welford's algorithm (Welford, 1962) and is

$$\sigma^{(t+1)} = \sqrt{\frac{(\sigma^{(t)})^2 t^2 + (y^{(t+1)} - \mu^{(t)})(y^{(t+1)} - \mu^{(t+1)})}{t(t+1)}}. \tag{109}$$

The disadvantage of the z-score-based cut-off point selection is that it ignores the pseudo-replication structure of the data by treating each cell as an independent replicate. Accordingly, the z-score does not reflect the uncertainty accurately.

To get an accurate reflection of the uncertainty, I implemented an online algorithm for penalized linear regression with pseudo-bulking. The algorithm takes as input a vector of observations y, the design matrix X, a contrast vector v, and an indicator vector b that assigns each observation to a pseudo-bulk (details in Algorithm 3). The algorithm keeps a running mean for each pseudobulk group and calculates the updates of regression coefficients using the Woodbury matrix identity with the recursive least square algorithm (Woodbury, 1950). I could not find an online algorithm for calculating the squared sum of the residuals and, thus, recalculate this value after each update of β.

Algorithm 3 Online algorithm for linear regression with pseudo-bulking

Require: $y \in \mathbb{R}^C$ ▷ The observations
Require: $X \in \mathbb{R}^{C \times K}$ ▷ The design matrix
Require: $v \in \mathbb{R}^K$ ▷ The contrast vector
Require: $b \in \{1, 2, \ldots, B\}^C$ ▷ The pseudo-bulk indicators
Require: $\lambda \in \mathbb{R}^{+K}$ ▷ The ridge penalty vector

function ONLINE PSEUDO-BULK LINEAR REGRESSION(y, X, v, b, λ)

$m \leftarrow 0^B$ ▷ the pseudo-bulk means
$n \leftarrow 0^B$ ▷ the number of observations per pseudo-bulk
$X^{(\text{act})} = 0^{B \times K}$ ▷ The design matrix with the active rows
$n^{(\text{obs})} \leftarrow 0$ ▷ The number of pseudo-bulk observations
$\Gamma \leftarrow \text{diag}(1/\lambda)$ ▷ The inverse of the covariance matrix
$\beta \leftarrow 0^K$ ▷ The coefficients of the linear regression
$t \leftarrow 0^C$ ▷ The vector with the t-statistic calculated online

for $c \in 1, \ldots, C$ **do**

$\delta \leftarrow \frac{y_{b_c}}{n_{b_c}+1} - \frac{(1-v_{b_c})m_{b_c}}{n_{b_c}+1}$
$m_{b_c} \leftarrow m_{b_c} + \delta$
$n_{b_c} \leftarrow n_{b_c} + 1$
if $n_{b_c} = 1$ **then**
$X^{(\text{act})}_{b_c:} \leftarrow X_{c:}$
$n^{(\text{obs})} \leftarrow n^{(\text{obs})} + 1$
$\Gamma \leftarrow \Gamma - \frac{\Gamma X^{(\text{act})}_{b_c:} (X^{(\text{act})}_{b_c:})^T \Gamma}{1 + X^{(\text{act})}_{b_c:} \Gamma (X^{(\text{act})}_{b_c:})^T}$
$\beta \leftarrow \beta + \Gamma X^{(\text{act})}_{b_c:} (m_{b_c} - (X^{(\text{act})}_{b_c:})^T \beta)$
else
$\beta \leftarrow \beta + \Gamma(\delta X^{(\text{act})}_{b_c:})$
end if
$\text{RSS} \leftarrow ||m - X^{(\text{act})}\beta||_2^2$ ▷ After each update of β, I need to recalculate the residuals for each pseudo-bulk group
$\sigma^2 \leftarrow v \left(\frac{\text{RSS}}{n^{(\text{obs})} - K} \Gamma \right) v^T$
$t_c \leftarrow \frac{v^T \beta}{\sigma}$

end for
return t

end function

4.4 Analysis of ten glioblastoma samples

I will illustrate the functionality of LEMUR using a dataset by Zhao et al. (2021). They measured the gene expression of 2892 to 21 596 single cells for five deceased glioblastoma patients. They extracted tissue slices and for each patient measured gene expression in two conditions: with a panobinostat, a non-selective histone deacetylase (HDAC) inhibitor, and with DMSO, as a control condition (Fig. 40A and Table 4). I call each treatment and patient combination a separate sample; accordingly, the dataset consists of ten samples.

Table 4 | Overview of the glioblastoma dataset and the samples. This table is adapted from Ahlmann-Eltze and Huber (2023b)

Patient ID	Conditon	Age	Gender	Tumor location	# Cells
PW030	0.2 uM panobinostat	65	M	right parietal	7118
PW030	vehicle (DMSO)	65	M	right parietal	14478
PW032	0.2 uM panobinostat	61	M	left frontal	1401
PW032	vehicle (DMSO)	61	M	left frontal	1491
PW034	0.2 uM panobinostat	68	F	left parieto-occipital	2679
PW034	vehicle (DMSO)	68	F	left parieto-occipital	7782
PW036	0.2 uM panobinostat	56	M	right temporal	3096
PW036	vehicle (DMSO)	56	M	right temporal	6749
PW040	0.2 uM panobinostat	69	M	right temporal	1987
PW040	vehicle (DMSO)	69	M	right temporal	1119

Plotting all cells using a two-dimensional uniform manifold approximation and projection (UMAP) embedding on the log-transformed and size factor adjusted data shows that the cells separate mostly by treatment condition and patient (Fig. 40B). There is additional variation due to cell type heterogeneity, but it is obscured by the variation caused by the known covariates.

With LEMUR, I can absorb the variation from the known covariates. LEMUR models the known covariates with R and S; the remaining variation is the cell type heterogeneity modeled with Z. Figure 40B shows a UMAP of the cell embedding Z after fitting LEMUR adjusting for patient and treatment with a $P = 15$-dimensional latent space. As expected, the cell type heterogeneity shared across patients becomes clearer (Fig. 40C). The picture becomes even

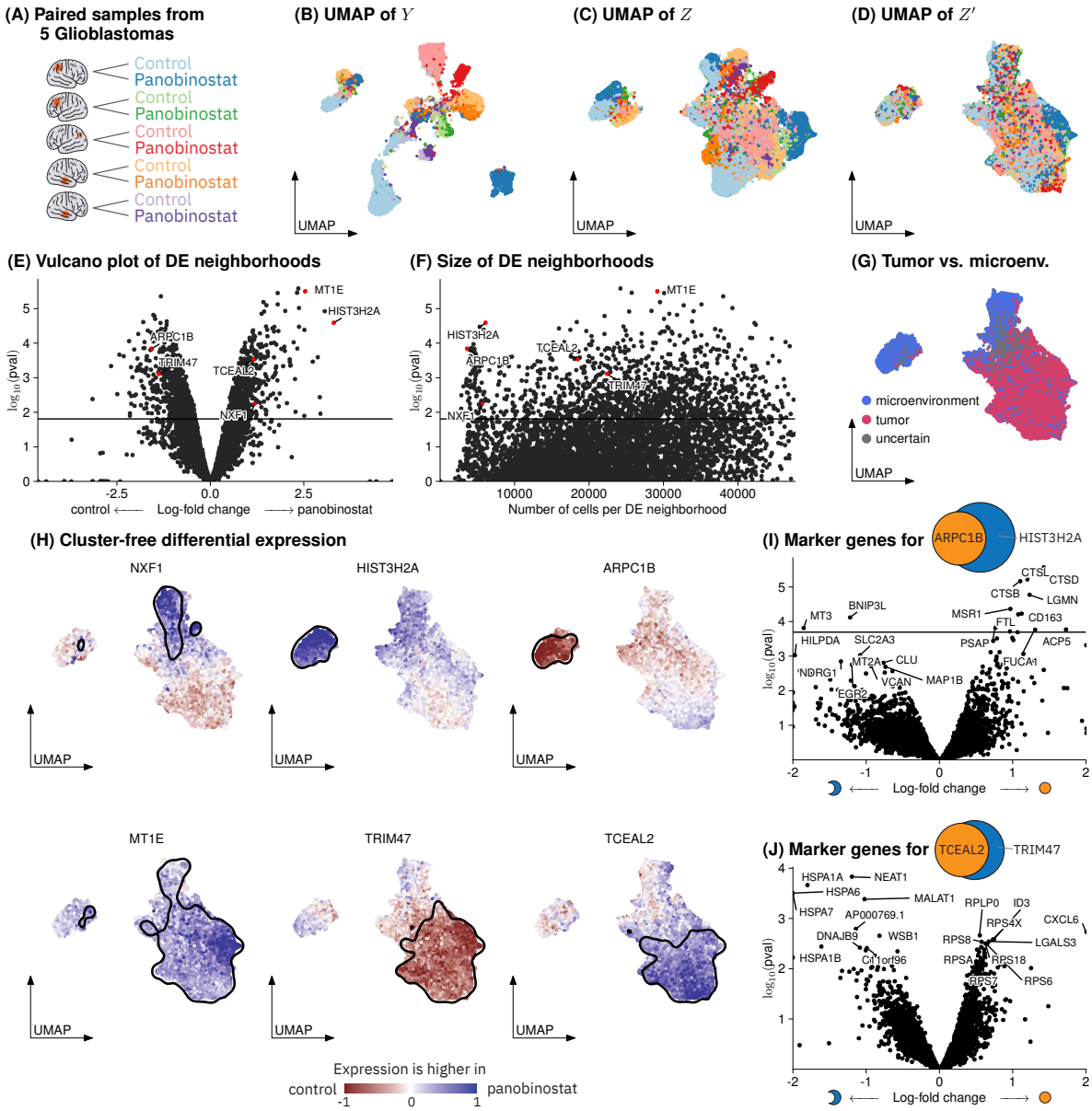


Figure 40 | Analysis of ten glioblastoma samples with LEMUR. (A) Schematic of the experimental design. (B-D) UMAP plots colored by sample (patient and condition) as shown in (A). Panel (B) shows a UMAP of Y, (C) Z with $S(x) = I$, and (D) Z' after adjusting the position with Harmony's maximum diversity clustering algorithm. (E) Volcano plot of the significance and log-fold change for each differential expression neighborhood with six selected genes highlighted in red. (F) Scatter plot of the number of cells for the same set of neighborhoods. (H) Differential expression pattern for the six selected genes plotted on the 2D UMAP embedding. The black boundary surrounds 90% of the cells in the differential expression neighborhood. (I) Volcano plot for the pseudo-bulked comparison of the differential expression neighborhoods of ARPC1B and HIST3H2A. The genes above the horizontal line have an FDR of less than 10%. (J) Same as (I), but for TCEAL2 and TRIM47. The brain ion is by `https://smart.servier.com`, licensed under *CC-BY 3.0 Unported*, and was adapted by me for this figure.
The data for this figure was generated by me and adapted from Ahlmann-Eltze and Huber (2023b).

clearer after aligning with S corresponding cell populations identified through the maximum diversity clustering of Harmony (Fig. 40D). Here, two non-tumor subpopulations and a large tumor subpopulation were visible (Fig. 40G). The classification was based on Zhao et al. (2021), who used a chromosome 7 amplification and chromosome 10 deletion.

I predicted the expression changes between the treated and control condition for each gene and cell. Furthermore, I identified the neighborhood that showed the most differential expression for each gene. In Figure 40E, I plotted the significance of the change for that neighborhood as a volcano plot and, in Figure 40F, the size of the neighborhood. It is known that panobinostat causes broad transcriptional changes (Atadja, 2009) and accordingly, I found that more than 20% of the gene-neighborhood pairs showed differential expression.

For each gene, I predicted the gene expression change for each cell. Figure 40H shows the predicted differences for six selected genes on the 2D UMAP. For example, the upregulation of *NXF1* in the panobinostat treatment appears to be limited to one cluster of microenvironment cells that expressed oligodendrocyte marker genes (Fig. 41). Figure 42 confirms that the inferred differences are reasonable based on the raw expression values per condition.

HIST3H2A and *ARPC1B* show the most differential expression in the non-tumor population expressing macrophage markers. In Figure 40I, I compared within the control condition if the cells that are in both differential neighborhoods express different genes than the cells that are exclusively in *HIST3H2A*'s differential expression neighborhood. *MT3* is more expressed in the cells exclusively in the *HIST3H2A* differential expression neighborhood. The cells in the *HIST3H2A* differential expression neighborhood expressed genes linked to tumor-associated macrophages (*CTSB, CTSD, CTLS*) at lower levels than the shared neighborhood. The cells in the *HIST3H2A* differential expression neighborhood showed higher expression of *MT3*, which has been associated with brain tissue macrophages (Yoshiyama et al., 1998), and *BNIP3L*, which is related to apoptosis (Imazu et al., 1999).

The differential expression neighborhoods of *TCEAL* and *TRIM47* were limited to the tumor cells and the *TCEAL2* neighborhood was largely a subset of the *TRIM47* neighborhood. Again, comparing the genes that are typical for the cells shared between *TCEAL2* and *TRIM47* with the genes up-regulated for cells exclusively in the *TRIM47* differential expression neighborhood showed a clear pattern: cells exclusively in the *TRIM47* differential expression neighborhood showed increased expression of heat shock proteins which suggests cellular stress (Benjamin and McMillan, 1998). In contrast, the cells in both neighborhoods show increased expression of many ribosomal genes, suggesting transcriptional activity, and chemokines linked to immunosuppressive microenvironments (Wang et al., 2021). It is important to note that even though the individual genes were not significant individually (based on a FDR $< 10\%$), a gene set enrichment analysis found up-regulation of translation and down-regulation of response to unfolded proteins as significant.

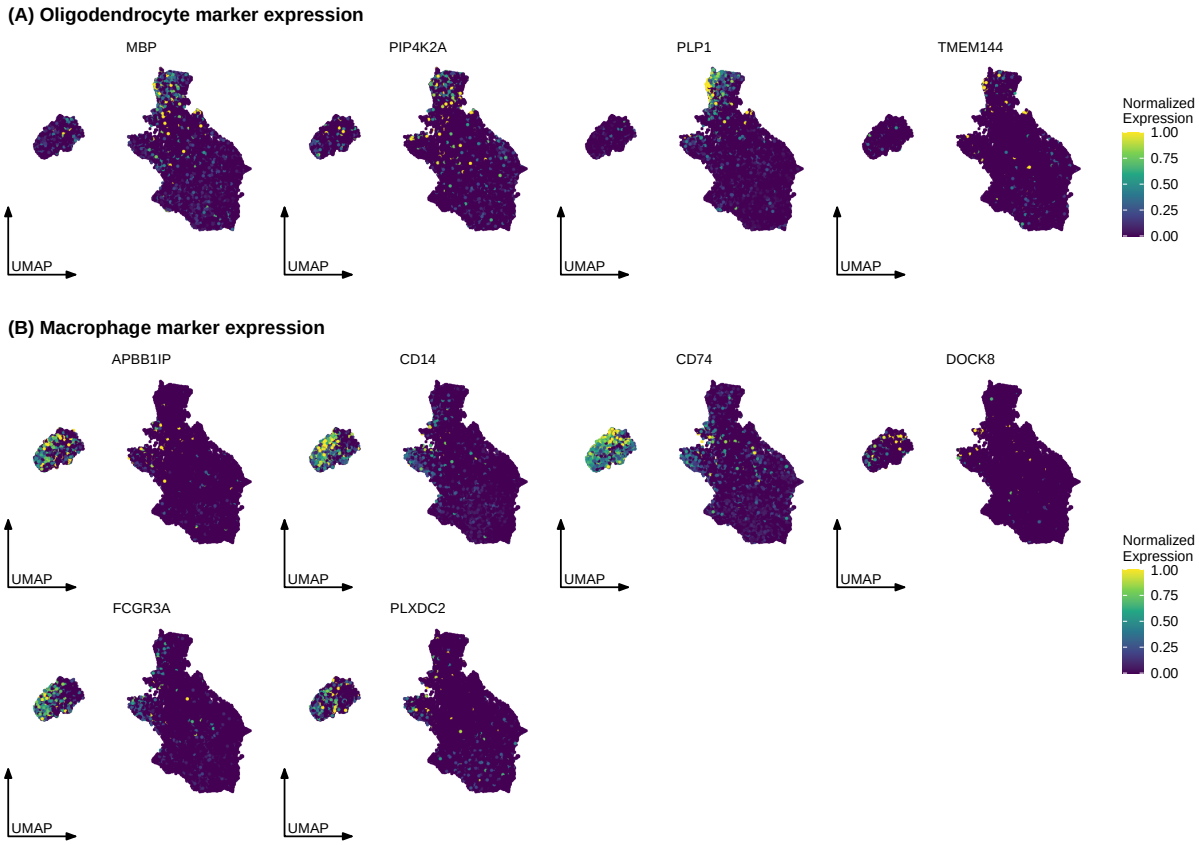


Figure 41 | Oligodendrocyte and macrophage marker expression of the glioblastoma cell populations. Normalized marker gene expression plotted on the UMAP of Z' for (A) oligodendrocyte markers and (B) for macrophages. The selected genes were manually curated based on the CZ CELLxGENE cell type annotation (Chan-Zuckerberg Initiative, 2023).
The data for this figure was generated by me and adapted from Ahlmann-Eltze and Huber (2023b).

4.5 Discussion

In this chapter, I introduced my new model for analyzing multi-condition single-cell data called *latent embedding multivariate regression*, short *LEMUR*. The core innovation of LEMUR is the regression on the subspaces to relate the experimental conditions with a parametric model. Furthermore, I showed that it is possible to fine-tune the position of corresponding cell populations in the latent space with a global linear transformation. I demonstrated LEMUR's utility on a glioblastoma dataset that compared the expression of samples treated with panobinostat with control samples. LEMUR's alignment can adjust for the known treatment and patient covariates and expose the latent cell type heterogeneity.

The motivation for LEMUR is to provide a flexible tool for analyzing differential expression in multi-condition single-cell data. I demonstrated that LEMUR can infer differential expression for each gene and cell and aggregate the information to find differential expression neighborhoods of cells that show consistent differential expression. I showed that the differences between the differential expression neighborhoods are biologically meaningful and can help to characterize

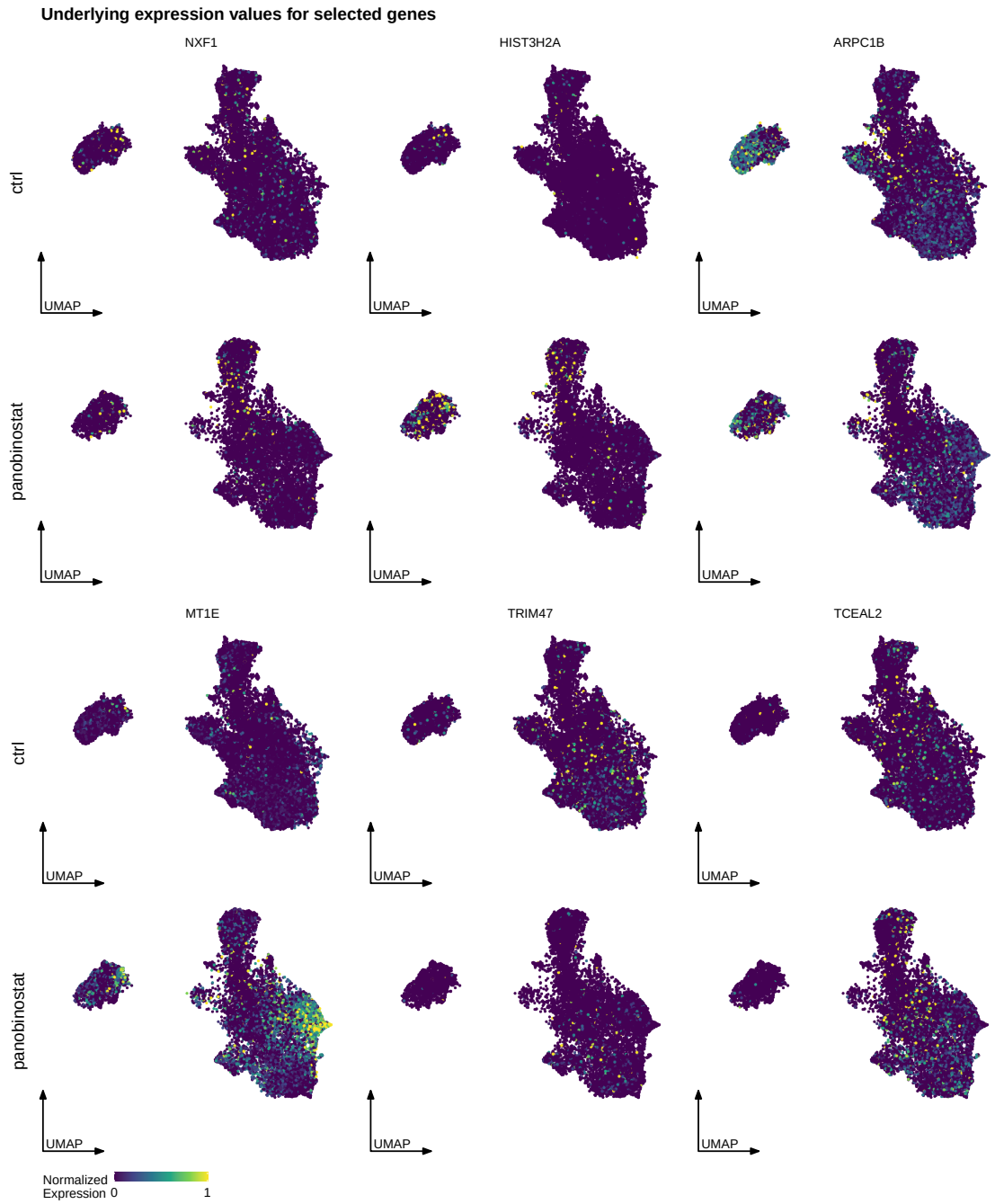


Figure 42 | Expression values for six selected genes for which Figure 40H shows the predicted differential expression values.
The data for this figure was generated by me and adapted from Ahlmann-Eltze and Huber (2023b).

the response pattern to the experimental treatment in depth.

Some aspects of the LEMUR implementation can still be improved in future versions: (1) the inference of the multi-condition PCA coefficients is only approximative. It is possible to refine the initial fit using gradient descent. However, this requires computing the gradient through the Grassmann manifold exponential map which is not straightforward. (2) The alignment step currently calculates the mean of the landmarks. This can lead to pathological alignment results

if taking the mean drastically changes the volume of the space spanned by the landmarks. Bai and Bartoli (2022) suggested an alternative perspective based on calculating the eigenvalues and vectors for each landmark set which can be solved in closed form for the generalized Procrustes problem. However, it is not immediately obvious how to generalize this approach to the Procrustes regression framework. (3) There is currently no empirical guidance on how to best infer the differential expression neighborhoods. The results can be sensitive to small changes in the number of latent dimensions. Here, more work is needed to systematically compare the different options and develop heuristics under which circumstances each approach performs best.

Overall, I believe that LEMUR is already a valuable tool for the analysis of multi-condition single-cell data. It delays the need to cluster the cells and thus avoids problems like choosing the optimal clustering resolution. And despite the challenging task of optimizing a regression on a manifold, my implementation is very efficient, in particular, compared with deep learning methods. The simplicity of my model and its parametric nature make it easy to inspect and enable follow-up on discoveries.

5 Discussion

In this thesis, I presented my contributions to the field of statistical method development in single-cell transcriptomics. In my first project, I developed a tool to fit gamma-Poisson generalized linear models on raw counts of single-cell datasets. In my second project, I compared different approaches to transform single-cell count data to stabilize the variance. In my third project, I developed a new statistical approach for the analysis of multi-condition single-cell data, which combines ideas from matrix factorization and regression.

The first project was motivated by the great success of count-based analysis tools for analyzing bulk RNA-seq data. The hope was that a more accurate noise model of single-cell data would improve the performance of many single-cell analysis tasks, such as dimensionality reduction and k nearest neighbor inference.

The main innovation implemented in *glmGamPoi* was the optimized overdispersion estimation procedure that made it more robust and faster than the algorithms in *DESeq2* and *edgeR*. This feature was one of the reasons that *sctransform* adopted *glmGamPoi* as their inference engine in the second version.

Beyond that, the impact of *glmGamPoi* was limited. Count-based statistical models have been most successful in bulk-RNA-seq for differential expression analysis. *glmGamPoi* can conduct differential expression analysis on large single-cell datasets faster than *edgeR* and *DESeq2*, but this use-case is explicitly discouraged. Treating each cell as an independent replicate violates the indepence assumption of generalized linear models and leads to unrealistically small errors. Instead, it is encouraged to form pseudobulks per sample, and *glmGamPoi* provides functions and syntactic sugar to do so. However, after pseudobulking, *glmGamPoi* offers few additional benefits for differential expression analysis over *DESeq2* and *edgeR*.

Count-based dimensionality reduction was a third field, where I hoped that *glmGamPoi* or a derived implementation could improve the state-of-the-art inference. I began my PhD soon after the publication of *GLM PCA*, and it seemed an appealing statistical framework for dimensionality reduction of count data (Townes, 2019). Yet, after experimenting with *GLM PCA* and my implementation of the model, I found that the inference was unreliable as steps diverged or got stuck in local minima. In contrast, the inference algorithms for regular PCA benefit from the connection with the eigenvalue problem and can directly find the global optimum.

These observations and the debate between Hafemeister and Satija (2019) and Lause et al. (2021) motivated me to look more deeply into the best way to preprocess single-cell data. Initially, I wrote, together with Wolfgang Huber, an explainer article that gave an overview of the three approaches for transforming and variance-stabilizing single-cell count data (Ahlmann-Eltze and Huber, 2022, version 1). In subsequent revisions, we included additional benchmarks to give an empirical comparison of the performance of preprocessing methods (version 2-4 of the preprint (Ahlmann-Eltze and Huber, 2022) and the published manuscript (Ahlmann-Eltze and Huber, 2023b)). The benchmark demonstrated that the count-based models *GLM PCA* and *NewWave*

performed worse than a log transformation for dimensionality reduction.

The most surprising outcome of the benchmark was that none of the computationally more sophisticated transformations consistently outperformed the simple logarithm transformation. This result is important because, for most single-cell analyses, the question to log transform or to use an alternative transformation stands at the very beginning. My work provides empirical guidance in this question and hopefully spurns research into better approaches to preprocess single-cell RNA-seq counts.

The third project developing a new statistical framework for the analysis of multi-condition single-cell data, has run in parallel to the two other projects. It began in 2020 when Wolfgang Huber asked me to help write a grant on the analysis of such data. Since then, the project has gone through many iterations, but now we are at a point where we feel confident about the model and have shifted our focus to gathering user feedback demonstrating its capabilities.

I originally proposed a tensor factorization framework that modeled the hierarchical structure with a three-dimensional tensor. In later iterations, the model was refined and I implemented it as an interaction model between the known covariates and the latent factors. This problem was tractable, and I could find the parameters using an iterative inference scheme. However, the parameters were difficult to interpret because the coefficients for the interaction between known covariates and latent factors were correlated with the eigenvectors. The attempt to resolve the issue by introducing an orthogonality constraint led me to dive into differential geometry, Stiefel and Grassmann manifolds, and geodesic regression.

My current model, which frames the problem as regression on latent subspaces, is attractive because it neatly generalizes and combines regular PCA and regression and avoids the fiddly inference of latent interaction coefficients. I hope my tool will be useful as a generic first-line analysis tool for differential expression analysis of multi-condition single-cell data.

References

Agostinis, F., Romualdi, C., Sales, G., and Risso, D. (2022). NewWave: a scalable R/Bioconductor package for the dimensionality reduction and batch effect removal of single-cell RNA-seq data. *Bioinformatics*, 38(9):2648–2650.

Ahlmann-Eltze, C. (2021). Gamma-poisson distribution: const-ae.name.

Ahlmann-Eltze, C. and Huber, W. (2020). glmGamPoi: fitting gamma-Poisson generalized linear models on single cell count data. *Bioinformatics*, 36(24):5701–5702.

Ahlmann-Eltze, C. and Huber, W. (2022). Comparison of transformations for single-cell RNA-seq data. *bioRxiv*.

Ahlmann-Eltze, C. and Huber, W. (2023a). Analysis of multi-condition single-cell data with latent embedding multivariate regression. *bioRxiv*.

Ahlmann-Eltze, C. and Huber, W. (2023b). Comparison of transformations for single-cell RNA-seq data. *Nature Methods*.

Allison, D. B., Cui, X., Page, G. P., and Sabripour, M. (2006). Microarray data analysis: from disarray to consolidation and consensus. *Nature reviews genetics*, 7(1):55–65.

Anders, S. and Huber, W. (2010). Differential expression analysis for sequence count data. *Heidelberg, Germany: European Molecular Biology Laboratory (EMBL)*, 11:Genome Biology.

Angelidis, I., Simon, L. M., Fernandez, I. E., Strunz, M., Mayr, C. H., Greiffo, F. R., Tsitsiridis, G., Ansari, M., Graf, E., Strom, T.-M., et al. (2019). An atlas of the aging lung mapped by single cell transcriptomics and deep tissue proteomics. *Nature Communications*, 10(1):1–17.

Atadja, P. (2009). Development of the pan-dac inhibitor panobinostat (lbh589): successes and challenges. *Cancer letters*, 280(2):233–241.

Bagnoli, J. W., Ziegenhain, C., Janjic, A., Wange, L. E., Vieth, B., Parekh, S., Geuder, J., Hellmann, I., and Enard, W. (2018). Sensitive and powerful single-cell RNA sequencing using mcSCRB-seq. *Nature Communications*, 9(1):1–8.

Bai, F. and Bartoli, A. (2022). Procrustes analysis with deformations: A closed-form solution by eigenvalue decomposition. *International Journal of Computer Vision*, pages 1–27.

Baron, M., Veres, A., Wolock, S. L., Faust, A. L., Gaujoux, R., Vetere, A., Ryu, J. H., Wagner, B. K., Shen-Orr, S. S., Klein, A. M., et al. (2016). A single-cell transcriptomic map of the human and mouse pancreas reveals inter-and intra-cell population structure. *Cell Systems*, 3(4):346–360.

Bartlett, M. S. (1936). The Square Root Transformation in Analysis of Variance. *Supplement to the Journal of the Royal Statistical Society*, 3(1):68.

Bartlett, M. S. (1947). The Use of Transformations. *Biometrics*, 3(1):39.

Becht, E., McInnes, L., Healy, J., Dutertre, C.-A., Kwok, I. W., Ng, L. G., Ginhoux, F., and Newell, E. W. (2019). Dimensionality reduction for visualizing single-cell data using UMAP. *Nature Biotechnology*, 37(1):38–44.

Bendokat, T., Zimmermann, R., and Absil, P.-A. (2020). A Grassmann Manifold Handbook: Basic Geometry and Computational Aspects. arXiv:2011.13699 [cs, math].

Benjamin, I. J. and McMillan, D. R. (1998). Stress (heat shock) proteins: molecular chaperones in cardiovascular biology and disease. *Circulation research*, 83(2):117–132.

Benjamini, Y. (2010). Discovering the false discovery rate. *Journal of the Royal Statistical Society: series B (statistical methodology)*, 72(4):405–416.

Benjamini, Y. and Hochberg, Y. (1995). Controlling the false discovery rate: a practical and powerful approach to multiple testing. *Journal of the Royal statistical society: series B (Methodological)*, 57(1):289–300.

Blondel, V. D., Guillaume, J.-L., Lambiotte, R., and Lefebvre, E. (2008). Fast unfolding of communities in large networks. *Journal of statistical mechanics: theory and experiment*, 2008(10):P10008.

Booeshaghi, A. S., Hallgrímsdóttir, I. B., Gálvez-Merchán, Á., and Pachter, L. (2022). Depth normalization for single-cell genomics count data. *bioRxiv*, pages 2022–05.

Bourgon, R., Gentleman, R., and Huber, W. (2010). Independent filtering increases detection power for high-throughput experiments. *Proceedings of the National Academy of Sciences*, 107(21):9546–9551.

Breda, J., Zavolan, M., and van Nimwegen, E. (2021). Bayesian inference of gene expression states from single-cell RNA-seq data. *Nature Biotechnology*, 39(8):1008–1016.

Brown, J., Ni, Z., Mohanty, C., Bacher, R., and Kendziorski, C. (2021). Normalization by distributional resampling of high throughput single-cell RNA-sequencing data. *Bioinformatics*, 37(22):4123–4128.

Bulaeva, E., Pellacani, D., Nakamichi, N., Hammond, C. A., Beer, P. A., Lorzadeh, A., Moksa, M., Carles, A., Bilenky, M., Lefort, S., et al. (2020). MYC-induced human acute myeloid leukemia requires a continuing IL-3/GM-CSF costimulus. *Blood*, 136(24):2764–2773.

Burkhardt, D. B., Stanley III, J. S., Tong, A., Perdigoto, A. L., Gigante, S. A., Herold, K. C., Wolf, G., Giraldez, A. J., van Dijk, D., and Krishnaswamy, S. (2021). Quantifying the effect of experimental perturbations at single-cell resolution. *Nature Biotechnology*, 39(5):619–629.

Butler, A., Hoffman, P., Smibert, P., Papalexi, E., and Satija, R. (2018). Integrating single-cell transcriptomic data across different conditions, technologies, and species. *Nature Biotechnology*, 36(5):411–420.

Cannoodt, R., Saelens, W., Deconinck, L., and Saeys, Y. (2021). Spearheading future omics analyses using dyngen, a multi-modal simulator of single cells. *Nature Communications*, 12(1):1–9.

Chan-Zuckerberg Initiative (2023). CZ CELLxGENE Discover. `https://cellxgene.cziscience.com/`. Accessed: 2023-03-02.

Chen, Y., Dougherty, E. R., and Bittner, M. L. (1997). Ratio-based decisions and the quantitative analysis of cdna microarray images. *Journal of Biomedical optics*, 2(4):364–374.

Choudhary, S. and Satija, R. (2022). Comparison and evaluation of statistical error models for scrna-seq. *Genome biology*, 23(1):27.

Crowell, H. L., Soneson, C., Germain, P.-L., Calini, D., Collin, L., Raposo, C., Malhotra, D., and Robinson, M. D. (2020). Muscat detects subpopulation-specific state transitions from multi-sample multi-condition single-cell transcriptomics data. *Nature communications*, 11(1):6077.

Dann, E., Henderson, N. C., Teichmann, S. A., Morgan, M. D., and Marioni, J. C. (2022). Differential abundance testing on single-cell data using k-nearest neighbor graphs. *Nature Biotechnology*, 40(2):245–253.

De Santis, R., Etoc, F., Rosado-Olivieri, E. A., and Brivanlou, A. H. (2021). Self-organization of human dorsal-ventral forebrain structures by light induced SHH. *Nature Communications*, 12(1):1–11.

Ding, C. and Cantor, C. R. (2004). Quantitative analysis of nucleic acids-the last few years of progress. *BMB Reports*, 37(1):1–10.

Dorfman, R. (1938). A note on the δ-method for finding variance formulae. *Biometric Bulletin*.

Dunn, P. K. and Smyth, G. K. (1996). Randomized quantile residuals. *Journal of Computational and graphical statistics*, 5(3):236–244.

Dunn, P. K. and Smyth, G. K. (2018). *Generalized linear models with examples in R*, volume 53. Springer.

Efron, B. (2005). Local false discovery rates.

Efron, B. and Tibshirani, R. (2002). Empirical bayes methods and false discovery rates for microarrays. *Genetic epidemiology*, 23(1):70–86.

Eid, J., Fehr, A., Gray, J., Luong, K., Lyle, J., Otto, G., Peluso, P., Rank, D., Baybayan, P., Bettman, B., et al. (2009). Real-time DNA sequencing from single polymerase molecules. *Science*, 323(5910):133–138.

Emrich, S. J., Barbazuk, W. B., Li, L., and Schnable, P. S. (2007). Gene discovery and annotation using lcm-454 transcriptome sequencing. *Genome research*, 17(1):69–73.

Falcon, S., Morgan, M., and Gentleman, R. (2006). *An introduction to Bioconductor's ExpressionSet class*. Biobase package version 2.61.0.

Finak, G., McDavid, A., Yajima, M., Deng, J., Gersuk, V., Shalek, A. K., Slichter, C. K., Miller, H. W., McElrath, M. J., Prlic, M., et al. (2015). Mast: a flexible statistical framework for assessing transcriptional changes and characterizing heterogeneity in single-cell RNA sequencing data. *Genome biology*, 16(1):1–13.

Fletcher, T. (2011). Geodesic regression on riemannian manifolds. In *Proceedings of the Third International Workshop on Mathematical Foundations of Computational Anatomy-Geometrical and Statistical Methods for Modelling Biological Shape Variability*, pages 75–86.

Garalde, D. R., Snell, E. A., Jachimowicz, D., Sipos, B., Lloyd, J. H., Bruce, M., Pantic, N., Admassu, T., James, P., Warland, A., et al. (2018). Highly parallel direct RNA sequencing on

an array of nanopores. *Nature Methods*, 15(3):201–206.

Gaublomme, J. T., Li, B., McCabe, C., Knecht, A., Yang, Y., Drokhlyansky, E., Van Wittenberghe, N., Waldman, J., Dionne, D., Nguyen, L., et al. (2019). Nuclei multiplexing with barcoded antibodies for single-nucleus genomics. *Nature communications*, 10(1):2907.

Gautier, L., Cope, L., Bolstad, B. M., and Irizarry, R. A. (2004). affy—analysis of affymetrix genechip data at the probe level. *Bioinformatics*, 20(3):307–315.

Gehring, J., Hwee Park, J., Chen, S., Thomson, M., and Pachter, L. (2020). Highly multiplexed single-cell RNA-seq by DNA oligonucleotide tagging of cellular proteins. *Nature Biotechnology*, 38(1):35–38.

Gentleman, R. C., Carey, V. J., Bates, D. M., Bolstad, B., Dettling, M., Dudoit, S., Ellis, B., Gautier, L., Ge, Y., Gentry, J., et al. (2004). Bioconductor: open software development for computational biology and bioinformatics. *Genome biology*, 5(10):1–16.

Gillespie, D. and Spiegelman, S. (1965). A quantitative assay for DNA-RNA hybrids with DNA immobilized on a membrane. *Journal of molecular biology*, 12(3):829–842.

Gillis, N. and Sharma, P. (2018). A semi-analytical approach for the positive semidefinite procrustes problem. *Linear Algebra and its Applications*, 540:112–137.

Goltsev, Y., Samusik, N., Kennedy-Darling, J., Bhate, S., Hale, M., Vazquez, G., Black, S., and Nolan, G. P. (2018). Deep profiling of mouse splenic architecture with codex multiplexed imaging. *Cell*, 174(4):968–981.

Golub, T. R., Slonim, D. K., Tamayo, P., Huard, C., Gaasenbeek, M., Mesirov, J. P., Coller, H., Loh, M. L., Downing, J. R., Caligiuri, M. A., et al. (1999). Molecular classification of cancer: class discovery and class prediction by gene expression monitoring. *science*, 286(5439):531–537.

Gower, J. C. (1975). Generalized procrustes analysis. *Psychometrika*, 40:33–51.

Gower, J. C. (2010). Procrustes methods. *Wiley Interdisciplinary Reviews: Computational Statistics*, 2(4):503–508.

Guo, C., Kong, W., Kamimoto, K., Rivera-Gonzalez, G. C., Yang, X., Kirita, Y., and Morris, S. A. (2019). Celltag indexing: genetic barcode-based sample multiplexing for single-cell genomics. *Genome biology*, 20(1):1–13.

Hafemeister, C. and Satija, R. (2019). Normalization and variance stabilization of single-cell RNA-seq data using regularized negative binomial regression. *Genome biology*, 20(1):296.

Hagemann-Jensen, M., Ziegenhain, C., Chen, P., Ramsköld, D., Hendriks, G.-J., Larsson, A. J., Faridani, O. R., and Sandberg, R. (2020). Single-cell RNA counting at allele and isoform resolution using Smart-seq3. *Nature Biotechnology*, 38(6):708–714.

Haghverdi, L., Lun, A. T., Morgan, M. D., and Marioni, J. C. (2018). Batch effects in single-cell RNA-sequencing data are corrected by matching mutual nearest neighbors. *Nature Biotechnology*, 36(5):421–427.

Hoerl, A. E. and Kennard, R. W. (1970). Ridge regression: Biased estimation for nonorthogonal

problems. *Technometrics*, 12(1):55–67.

Hotelling, H. (1936). Relations between two sets of variates. *Biometrika*, 28(3/4):321–377.

Huber, W., Von Heydebreck, A., Sültmann, H., Poustka, A., and Vingron, M. (2002). Variance stabilization applied to microarray data calibration and to the quantification of differential expression. *Bioinformatics*, 18(suppl_1):S96–S104.

Ignatiadis, N., Klaus, B., Zaugg, J. B., and Huber, W. (2016). Data-driven hypothesis weighting increases detection power in genome-scale multiple testing. *Nature Methods*, 13(7):577–580.

Imazu, T., Shimizu, S., Tagami, S., Matsushima, M., Nakamura, Y., Miki, T., Okuyama, A., and Tsujimoto, Y. (1999). Bcl-2/e1b 19 kda-interacting protein 3-like protein (bnip3l) interacts with bcl-2/bcl-xl and induces apoptosis by altering mitochondrial membrane permeability. *Oncogene*, 18(32):4523–4529.

Islam, S., Zeisel, A., Joost, S., La Manno, G., Zajac, P., Kasper, M., Lönnerberg, P., and Linnarsson, S. (2014). Quantitative single-cell RNA-seq with unique molecular identifiers. *Nature Methods*, 11(2):163–166.

Jingjing, P., Qingwen, W., Zhenyun, P., and Zhencheng, C. (2019). Solution of symmetric positive semidefinite procrustes problem. *The Electronic Journal of Linear Algebra*, 35:543–554.

Johnsson, P., Ziegenhain, C., Hartmanis, L., Hendriks, G.-J., Hagemann-Jensen, M., Reinius, B., and Sandberg, R. (2022). Transcriptional kinetics and molecular functions of long noncoding RNAs. *Nature Genetics*, 54(3):306–317.

Kafatos, F. C., Jones, C. W., and Efstratiadis, A. (1979). Determination of nucleic acid sequence homologies and relative concentrations by a dot hybridization procedure. *Nucleic acids research*, 7(6):1541–1552.

Kang, H. M., Subramaniam, M., Targ, S., Nguyen, M., Maliskova, L., McCarthy, E., Wan, E., Wong, S., Byrnes, L., Lanata, C. M., et al. (2018). Multiplexed droplet single-cell RNA-sequencing using natural genetic variation. *Nature Biotechnology*, 36(1):89–94.

Kathiriya, J. J., Wang, C., Zhou, M., Brumwell, A., Cassandras, M., Le Saux, C. J., Cohen, M., Alysandratos, K.-D., Wang, B., Wolters, P., et al. (2022). Human alveolar type 2 epithelium transdifferentiates into metaplastic KRT5+ basal cells. *Nature Cell Biology*, 24(1):10–23.

Kathleen Kerr, M. (2003). Design considerations for efficient and effective microarray studies. *Biometrics*, 59(4):822–828.

Kharchenko, P. V. (2021). The triumphs and limitations of computational methods for scrna-seq. *Nature Methods*, 18(7):723–732.

Kharchenko, P. V., Silberstein, L., and Scadden, D. T. (2014). Bayesian approach to single-cell differential expression analysis. *Nature Methods*, 11(7):740–742.

Kim, H. J., Adluru, N., Collins, M. D., Chung, M. K., Bendin, B. B., Johnson, S. C., Davidson, R. J., and Singh, V. (2014). Multivariate General Linear Models (MGLM) on Riemannian Manifolds with Applications to Statistical Analysis of Diffusion Weighted Images. In *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pages 2705–2712, Columbus,

OH. IEEE.

Klein, A. M., Mazutis, L., Akartuna, I., Tallapragada, N., Veres, A., Li, V., Peshkin, L., Weitz, D. A., and Kirschner, M. W. (2015). Droplet barcoding for single-cell transcriptomics applied to embryonic stem cells. *Cell*, 161(5):1187–1201.

Kobak, D. and Berens, P. (2019). The art of using t-sne for single-cell transcriptomics. *Nature communications*, 10(1):5416.

Korsunsky, I., Millard, N., Fan, J., Slowikowski, K., Zhang, F., Wei, K., Baglaenko, Y., Brenner, M., Loh, P.-r., and Raychaudhuri, S. (2019). Fast, sensitive and accurate integration of single-cell data with harmony. *Nature Methods*, 16(12):1289–1296.

Larsson, A. J., Ziegenhain, C., Hagemann-Jensen, M., Reinius, B., Jacob, T., Dalessandri, T., Hendriks, G.-J., Kasper, M., and Sandberg, R. (2021). Transcriptional bursts explain autosomal random monoallelic expression and affect allelic imbalance. *PLoS Computational Biology*, 17(3):e1008772.

Lause, J., Berens, P., and Kobak, D. (2021). Analytic pearson residuals for normalization of single-cell RNA-seq umi data. *Genome biology*, 22(1):1–20.

Law, C. W., Chen, Y., Shi, W., and Smyth, G. K. (2014). voom: Precision weights unlock linear model analysis tools for RNA-seq read counts. *Genome biology*, 15(2):1–17.

Liang, P. (2002). A decade of differential display. *Biotechniques*, 33(2):338–346.

Liang, P. and Pardee, A. B. (1992). Differential display of eukaryotic messenger RNA by means of the polymerase chain reaction. *Science*, 257(5072):967–971.

Lipshutz, R. J., Fodor, S., Gingeras, T. R., and Lockhart, D. J. (1999). High density synthetic oligonucleotide arrays. *Nature genetics*, 21(1):20–24.

Lister, R., O'Malley, R. C., Tonti-Filippini, J., Gregory, B. D., Berry, C. C., Millar, A. H., and Ecker, J. R. (2008). Highly integrated single-base resolution maps of the epigenome in arabidopsis. *Cell*, 133(3):523–536.

Lockhart, D. J., Dong, H., Byrne, M. C., Follettie, M. T., Gallo, M. V., Chee, M. S., Mittmann, M., Wang, C., Kobayashi, M., Norton, H., et al. (1996). Expression monitoring by hybridization to high-density oligonucleotide arrays. *Nature Biotechnology*, 14(13):1675–1680.

Lockhart, D. J. and Winzeler, E. A. (2000). Genomics, gene expression and DNA arrays. *Nature*, 405(6788):827–836.

Lönnstedt, I. and Speed, T. (2002). Replicated microarray data. *Statistica sinica*, pages 31–46.

Lopez, R., Regier, J., Cole, M. B., Jordan, M. I., and Yosef, N. (2018). Deep generative modeling for single-cell transcriptomics. *Nature Methods*, 15(12):1053–1058.

Love, M. I., Huber, W., and Anders, S. (2014). Moderated estimation of fold change and dispersion for RNA-seq data with DESeq2. *Genome biology*, 15(12):1–21.

Lu, X., Hosono, Y., Nagae, M., Ishizuka, S., Ishikawa, E., Motooka, D., Ozaki, Y., Sax, N., Maeda, Y., Kato, Y., et al. (2021). Identification of conserved SARS-CoV-2 spike epitopes that expand public cTfh clonotypes in mild COVID-19 patients. *Journal of Experimental*

Medicine, 218(12).

Luecken, M. D., Büttner, M., Chaichoompu, K., Danese, A., Interlandi, M., Müller, M. F., Strobl, D. C., Zappia, L., Dugas, M., Colomé-Tatché, M., et al. (2022). Benchmarking atlas-level data integration in single-cell genomics. *Nature Methods*, 19(1):41–50.

Luecken, M. D. and Theis, F. J. (2019). Current best practices in single-cell RNA-seq analysis: a tutorial. *Molecular systems biology*, 15(6):e8746.

Lun, A. T., Pagès, H., and Smith, M. L. (2018). beachmat: A bioconductor c++ api for accessing high-throughput biological data from a variety of r matrix types. *PLoS Computational Biology*, 14(5):e1006135.

Lund, S. P., Nettleton, D., McCarthy, D. J., and Smyth, G. K. (2012). Detecting differential expression in RNA-sequence data using quasi-likelihood with shrunken dispersion estimates. *Statistical applications in genetics and molecular biology*, 11(5).

Macosko, E. Z., Basu, A., Satija, R., Nemesh, J., Shekhar, K., Goldman, M., Tirosh, I., Bialas, A. R., Kamitaki, N., Martersteck, E. M., et al. (2015). Highly parallel genome-wide expression profiling of individual cells using nanoliter droplets. *Cell*, 161(5):1202–1214.

Marioni, J. C., Mason, C. E., Mane, S. M., Stephens, M., and Gilad, Y. (2008). Rna-seq: an assessment of technical reproducibility and comparison with gene expression arrays. *Genome research*, 18(9):1509–1517.

McCarthy, D. J., Chen, Y., and Smyth, G. K. (2012). Differential expression analysis of multifactor RNA-seq experiments with respect to biological variation. *Nucleic acids research*, 40(10):4288–4297.

McFarland, J. M., Paolella, B. R., Warren, A., Geiger-Schuller, K., Shibue, T., Rothberg, M., Kuksenko, O., Colgan, W. N., Jones, A., Chambers, E., et al. (2020). Multiplexed single-cell transcriptional response profiling to define cancer vulnerabilities and therapeutic mechanism of action. *Nature communications*, 11(1):4296.

McGinnis, C. S., Patterson, D. M., Winkler, J., Conrad, D. N., Hein, M. Y., Srivastava, V., Hu, J. L., Murrow, L. M., Weissman, J. S., Werb, Z., et al. (2019). Multi-seq: sample multiplexing for single-cell RNA sequencing using lipid-tagged indices. *Nature Methods*, 16(7):619–626.

McInnes, L., Healy, J., and Melville, J. (2018). UMAP: Uniform manifold approximation and projection for dimension reduction. *arXiv preprint arXiv:1802.03426*.

Nagalakshmi, U., Wang, Z., Waern, K., Shou, C., Raha, D., Gerstein, M., and Snyder, M. (2008). The transcriptional landscape of the yeast genome defined by RNA sequencing. *Science*, 320(5881):1344–1349.

Nelder, J. A. and Wedderburn, R. W. M. (1972). Generalized Linear Models. *Journal of the Royal Statistical Society. Series A (General)*, 135(3):370.

Nomaru, H., Liu, Y., De Bono, C., Righelli, D., Cirino, A., Wang, W., Song, H., Racedo, S. E., Dantas, A. G., Zhang, L., et al. (2021). Single cell multi-omic analysis identifies a Tbx1-dependent multilineage primed population in murine cardiopharyngeal mesoderm.

Nature Communications, 12(1):1–19.

Pages, H., Hickey, P., and Lun, A. (2022). *DelayedArray: A unified framework for working transparently with on-disk and in-memory array-like datasets*. R package version 0.24.0.

Pal, B., Chen, Y., Milevskiy, M. J., Vaillant, F., Prokopuk, L., Dawson, C. A., Capaldo, B. D., Song, X., Jackling, F., Timpson, P., et al. (2021). Single cell transcriptome atlas of mouse mammary epithelial cells across development. *Breast Cancer Research*, 23(1):1–19.

Palla, G., Spitzer, H., Klein, M., Fischer, D., Schaar, A. C., Kuemmerle, L. B., Rybakov, S., Ibarra, I. L., Holmberg, O., Virshup, I., et al. (2022). Squidpy: a scalable framework for spatial omics analysis. *Nature Methods*, 19(2):171–178.

Panebianco, C. J., Dave, A., Charytonowicz, D., Sebra, R., and Iatridis, J. C. (2021). Single-cell RNA-sequencing atlas of bovine caudal intervertebral discs: Discovery of heterogeneous cell populations with distinct roles in homeostasis. *The FASEB Journal*, 35(11):e21919.

Picelli, S., Björklund, Å. K., Faridani, O. R., Sagasser, S., Winberg, G., and Sandberg, R. (2013). Smart-seq2 for sensitive full-length transcriptome profiling in single cells. *Nature Methods*, 10(11):1096–1098.

Picelli, S., Faridani, O. R., Björklund, Å. K., Winberg, G., Sagasser, S., and Sandberg, R. (2014). Full-length RNA-seq from single cells using Smart-seq2. *Nature protocols*, 9(1):171–181.

Porritt, R. A., Zemmour, D., Abe, M., Lee, Y., Narayanan, M., Carvalho, T. T., Gomez, A. C., Martinon, D., Santiskulvong, C., Fishbein, M. C., et al. (2021). NLRP3 inflammasome mediates immune-stromal interactions in vasculitis. *Circulation Research*, 129(9):e183–e200.

Qian, Y., Arellano, G., Ifergan, I., Lin, J., Snowden, C., Kim, T., Thomas, J. J., Law, C., Guan, T., Balabanov, R. D., et al. (2021). ZEB1 promotes pathogenic Th1 and Th17 cell differentiation in multiple sclerosis. *Cell Reports*, 36(8):109602.

Qiu, C., Martin, B. K., Welsh, I. C., Daza, R. M., Le, T.-M., Huang, X., Nichols, E. K., Taylor, M. L., Fulton, O., Gomes, A. R., et al. (2023). A single-cell transcriptional timelapse of mouse embryonic development, from gastrula to pup. *bioRxiv*.

Regev, A., Teichmann, S. A., Lander, E. S., Amit, I., Benoist, C., Birney, E., Bodenmiller, B., Campbell, P., Carninci, P., Clatworthy, M., et al. (2017). The human cell atlas. *elife*, 6:e27041.

Rentmeesters, Q. (2011). A gradient method for geodesic data fitting on some symmetric Riemannian manifolds. In *IEEE Conference on Decision and Control and European Control Conference*, pages 7141–7146, Orlando, FL, USA. IEEE.

Robinson, M. D., McCarthy, D. J., and Smyth, G. K. (2009). edgeR: a bioconductor package for differential expression analysis of digital gene expression data. *Bioinformatics*, 26(1):139–140.

Schaum, N., Karkanias, J., Neff, N. F., May, A. P., Quake, S. R., Wyss-Coray, T., Darmanis, S., Batson, J., Botvinnik, O., Chen, M. B., et al. (2018). Single-cell transcriptomics of 20 mouse organs creates a tabula muris: The tabula muris consortium. *Nature*, 562(7727):367.

Schena, M., Shalon, D., Davis, R. W., and Brown, P. O. (1995). Quantitative monitoring of gene expression patterns with a complementary DNA microarray. *Science*, 270(5235):467–470.

Schena, M., Shalon, D., Heller, R., Chai, A., Brown, P. O., and Davis, R. W. (1996). Parallel human genome analysis: microarray-based expression monitoring of 1000 genes. *Proceedings of the National Academy of Sciences*, 93(20):10614–10619.

Schönemann, P. H. (1966). A generalized solution of the orthogonal procrustes problem. *Psychometrika*, 31(1):1–10.

Segert, S. (2022). Procrustes problem for rank-deficient input. Cross Validated. URL:https://stats.stackexchange.com/q/599015 (version: 2022-12-14).

Siletti, K., Hodge, R. D., Mossi Albiach, A., Hu, L., Lee, K. W., Lönnerberg, P., Bakken, T. E., Ding, S.-L., Clark, M., Casper, T., et al. (2022). Transcriptomic diversity of cell types across the adult human brain. *bioRxiv*, pages 2022–10.

Smyth, G. K. (2004). Linear models and empirical bayes methods for assessing differential expression in microarray experiments. *Statistical applications in genetics and molecular biology*, 3(1).

Smyth, G. K. and Speed, T. (2003). Normalization of cdna microarray data. *Methods*, 31(4):265–273.

Stark, R., Grzelak, M., and Hadfield, J. (2019). Rna sequencing: the teenage years. *Nature Reviews Genetics*, 20(11):631–656.

Stoeckius, M., Zheng, S., Houck-Loomis, B., Hao, S., Yeung, B. Z., Mauck, W. M., Smibert, P., and Satija, R. (2018). Cell hashing with barcoded antibodies enables multiplexing and doublet detection for single cell genomics. *Genome biology*, 19(1):1–12.

Storey, J. D. (2002). A direct approach to false discovery rates. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 64(3):479–498.

Storey, J. D. (2003). The positive false discovery rate: a bayesian interpretation and the q-value. *The annals of statistics*, 31(6):2013–2035.

Sun, T., Song, D., Li, W. V., and Li, J. J. (2021). scDesign2: a transparent simulator that generates high-fidelity single-cell gene expression count data with gene correlations captured. *Genome Biology*, 22(1):1–37.

Svensson, V. (2020). Droplet scrna-seq is not zero-inflated. *Nature Biotechnology*, 38(2):147–150.

Svensson, V., da Veiga Beltrame, E., and Pachter, L. (2020). A curated database reveals trends in single-cell transcriptomics. *Database*, 2020.

Svensson, V., Natarajan, K. N., Ly, L.-H., Miragaia, R. J., Labalette, C., Macaulay, I. C., Cvejic, A., and Teichmann, S. A. (2017). Power analysis of single-cell RNA-sequencing experiments. *Nature Methods*, 14(4):381–387.

Tang, F., Barbacioru, C., Wang, Y., Nordman, E., Lee, C., Xu, N., Wang, X., Bodeau, J., Tuch, B. B., Siddiqui, A., et al. (2009). mrna-seq whole-transcriptome analysis of a single cell. *Nature Methods*, 6(5):377–382.

Townes, F. W. (2019). Generalized principal component analysis. *arXiv preprint arXiv:1907.02647*.

Townes, F. W., Hicks, S. C., Aryee, M. J., and Irizarry, R. A. (2019). Feature selection and dimension reduction for single-cell RNA-seq based on a multinomial model. *Genome biology*, 20:1–16.

Traag, V. A., Waltman, L., and Van Eck, N. J. (2019). From louvain to leiden: guaranteeing well-connected communities. *Scientific reports*, 9(1):5233.

Tusher, V. G., Tibshirani, R., and Chu, G. (2001). Significance analysis of microarrays applied to the ionizing radiation response. *Proceedings of the National Academy of Sciences*, 98(9):5116–5121.

Van der Maaten, L. and Hinton, G. (2008). Visualizing data using t-sne. *Journal of machine learning research*, 9(11).

van Ruissen, F., Ruijter, J. M., Schaaf, G. J., Asgharnegad, L., Zwijnenburg, D. A., Kool, M., and Baas, F. (2005). Evaluation of the similarity of gene expression data estimated with sage and affymetrix genechips. *BMC genomics*, 6(1):1–16.

Velculescu, V. E., Zhang, L., Vogelstein, B., and Kinzler, K. W. (1995). Serial analysis of gene expression. *Science*, 270(5235):484–487.

Velculescu, V. E., Zhang, L., Zhou, W., Vogelstein, J., Basrai, M. A., Bassett, D. E., Hieter, P., Vogelstein, B., and Kinzler, K. W. (1997). Characterization of the Yeast Transcriptome. *Cell*, 88(2):243–251.

Wang, L. (2021). Single-cell normalization and association testing unifying crispr screen and gene co-expression analyses with normalisr. *Nature communications*, 12(1):6395.

Wang, Z., Gerstein, M., and Snyder, M. (2009). Rna-seq: a revolutionary tool for transcriptomics. *Nature reviews genetics*, 10(1):57–63.

Wang, Z., Liu, Y., Mo, Y., Zhang, H., Dai, Z., Zhang, X., Ye, W., Cao, H., Liu, Z., and Cheng, Q. (2021). The cxcl family contributes to immunosuppressive microenvironment in gliomas and assists in gliomas chemotherapy. *Frontiers in immunology*, 12:731751.

Warton, D. I. (2018). Why you cannot transform your way out of trouble for small counts. *Biometrics*, 74(1):362–368.

Welford, B. (1962). Note on a method for calculating corrected sums of squares and products. *Technometrics*, 4(3):419–420.

Welsh, J., Chada, K., Dalal, S. S., Cheng, R., Relph, D., and McClelland, M. (1992). Arbitrarily primed pcr fingerprinting of RNA. *Nucleic acids research*, 20(19):4965–4970.

whuber (2014). Which property of count data make mean-variance dependency? Cross Validated. URL:https://stats.stackexchange.com/q/112358 (version: 2014-08-18).

Wilt, F. H. (1962). The ontogeny of chick embryo hemoglobin. *Proceedings of the National Academy of Sciences*, 48(9):1582–1590.

Wolf, F. A., Angerer, P., and Theis, F. J. (2018). Scanpy: large-scale single-cell gene expression data analysis. *Genome biology*, 19:1–5.

Woodbury, M. A. (1950). *Inverting modified matrices*. Department of Statistics, Princeton

University.

Wooldridge, J. M. (2015). *Introductory econometrics: A modern approach*. Cengage learning.

Yang, Y. H. and Speed, T. (2002). Design issues for cdna microarray experiments. *Nature Reviews Genetics*, 3(8):579–588.

Yoshiyama, Y., Sato, H., Seiki, M., Shinagawa, A., Takahashi, M., and Yamada, T. (1998). Expression of the membrane-type 3 matrix metalloproteinase (mt3-mmp) in human brain tissues. *Acta neuropathologica*, 96:347–350.

Zhao, W., Dovas, A., Spinazzi, E. F., Levitin, H. M., Banu, M. A., Upadhyayula, P., Sudhakar, T., Marie, T., Otten, M. L., Sisti, M. B., et al. (2021). Deconvolution of cell type-specific drug responses in human tumor tissue with single-cell RNA-seq. *Genome Medicine*, 13(1):82.

Zheng, G. X., Terry, J. M., Belgrader, P., Ryvkin, P., Bent, Z. W., Wilson, R., Ziraldo, S. B., Wheeler, T. D., McDermott, G. P., Zhu, J., et al. (2017). Massively parallel digital transcriptional profiling of single cells. *Nature communications*, 8(1):14049.