DAESC-GPU: A GPU-powered Scalable Software for Single-cell Allele-Specific Expression Analysis

Tengfei Cui

A thesis

submitted in partial fulfillment of the

requirements for the degree of

Master of Science

University of Washington

2025

Committee:

Guanghao Qi

Kevin Lin

Program Authorized to Offer Degree:

Biostatistics

© 2025
Tengfei Cui

University of Washington

Abstract

DAESC-GPU: A GPU-powered Scalable Software for Single-cell Allele-Specific Expression Analysis

Tengfei Cui

Chair of the Supervisory Committee:

Guanghao Qi

Department of Biostatistics

Allele-specific expression (ASE) is a powerful signal to study cis-regulatory effects. We previously developed DAESC, a statistical method for single-cell differential ASE analysis across multiple individuals. Despite improved power, the lack of computational efficiency limits its utility on large-scale datasets. Here, we present DAESC-GPU, an accelerated version of DAESC powered by Graphics Processing Units (GPUs). DAESC-GPU is dozens of times faster than DAESC and scalable to datasets of over a million cells. Application of the software on single-cell ASE data from the OneK1K cohort identified novel genes with regulatory patterns specific to naïve and central memory CD4+ T cells.

Acknowledgements

I would like to express my heartfelt gratitude to my advisor, Dr. Guanghao Qi, for his patient guidance, unwavering support, and continuous inspiration throughout this project. He gave me the freedom to explore my own ideas, even when some of them did not succeed, and always offered a helping hand when I encountered difficulties or felt lost. I have learned so much from him—not only how to conduct research and overcome challenges, but also how to grow personally and professionally. He has provided me with invaluable advice during key moments in my academic journey and career planning. Dr. Qi is not only an exceptional mentor, but also one of my closest friends who introduced me to the world of genetics and genomics. I am truly grateful to have had the opportunity to learn from him.

Contents

Chapter 1

Introduction

Allele-specific expression (ASE) analysis investigates the differential expression between two alleles of the same gene, across different disease statuses or along the pseudotime trajectory of cells. Specifically, ASE measures the expression of one allele of a gene relative to the other in a diploid individual. ASE provides crucial insights into various regulatory mechanisms, including cis-regulatory variation [2], epigenetic effects [9], and nonsense-mediated decay [12].

Traditionally, ASE studies have relied on bulk RNA sequencing, which does not adequately capture expression variations across different cell types or cell states [10]. With the advent and rapid advancement of single-cell RNA sequencing (scRNA-seq), it is now feasible to measure allele-specific expression at the resolution of individual cells. Single-cell ASE (scASE) data offer valuable insights into the regulatory mechanisms underlying disease-associated loci; however, they also introduce new analytical and computational challenges.

One major methodological challenge is the absence of suitable statistical models specifically designed for scASE data. Previously, we developed DAESC, a statistical method tailored for differential ASE (D-ASE) analysis in scRNA-seq datasets involving multiple individuals [10]. DAESC was shown to maintain robust type I error control and high statistical power.

Furthermore, its application successfully identified hundreds of dynamically regulated genes during endoderm differentiation [10]. A computational challenge arised from the need for high-performance software capable of efficiently handling increasingly large scASE datasets. For DAESC, the initial R-based implementation of the variational EM algorithm struggled with scalability when analyzing datasets comprising hundreds of thousands to over one million cells. Therefore, there was an urgent need to develop scalable computational frameworks and software solutions to overcome these performance bottlenecks.

One potential solution was to develop software based on Graphics Processing Units (GPUs). DAESC-GPU enhanced computational efficiency by utilizing Graphics Processing Units (GPUs), which was an electronic circuit that can perform massively parallel computation at high-speed. GPUs, utilizing stream processing programming models, had become the dominant platforms for deep learning training and inference [5]. The availability of easily programmable GPUs with high floating-point performance had enabled the recent revolution in deep learning and significantly influenced further advancements in the field. Essentially, GPUs was massively parallel threaded processors. To efficiently harness this computational power, it was essential to use CUDA, a thread- and data-parallel C-like programming language [5]. CUDA served GPUs similarly to how an operating system serves a modern computer. Based on the CUDA programming system, the GPU programming ecosystem included numerous numerical libraries such as CuBLAS, CuSparse, and CuPy, which provided highly optimized numerical routines and reduced barriers to GPU adoption [5]. Applications leveraged these libraries to exploit GPU capabilities across various research disciplines, including physics, statistics, biostatistics, and genomics. Indeed, GPU acceleration had already enabled rapid genomic analyses, such as variant calling, at scales previously unattainable [7]. O'Connell et al. [7] compared GPU-accelerated software (NVIDIA Parabricks) with two traditional CPU-based germline callers (HaplotypeCaller and DeepVariant) and found NVIDIA Parabricks to be over 65 times faster. Furthermore, recent studies evaluating the

original CPU-based GATK HaplotypeCaller algorithm against its GPU-powered counterpart from NVIDIA Clara™ Parabricks demonstrated that the GPU-accelerated software provided highly consistent results (over 99.5% concordance) and was more than 21 times faster [11].

Inspired by these successful GPU implementations across diverse research fields, we proposed DAESC-GPU, a high-performance software designed specifically for single-cell D-ASE analysis. DAESC-GPU was GPU-powered and offers the following advantages:

- DAESC-GPU performed all mathematical computations via large-scale matrix operations, enabling simultaneous analysis of thousands of genes.
- DAESC-GPU utilized GPU acceleration, significantly enhancing computational speed compared to the original DAESC frameworks, while maintaining highly consistent estimates.
- DAESC-GPU was user-friendly and freely available. Users can easily deploy DAESC-GPU on platforms such as Google Colab, leveraging free GPU resources without the need for complex environmental setup.

Chapter 2

Methods

2.1 Summary of the DAESC model

DAESC is a generic model for differential ASE analysis using single-cell RNA-seq data of multiple individuals [10] against any variable of interest, such as disease status, genotype (eQTLs), continuous developmental trajectories, and cell types. DAESC has two versions, DAESC-BB and DAESC-Mix, used for different cases. Both versions are based on beta-binomial distributions. The simplified version (DAESC-BB) introduces one latent variable (a_i) which accounts for the sample repeat structure resulted from different cells measured by the same individual i. DAESC-BB is a more flexible version which can be fitted under any sample size [10]. The required computing resource by DAESC-BB is also lower than the other version DAESC-Mix. The advanced version, DAESC-Mix, has not only the latent variable a_i but also the latent variable z_i for each individual i. The latent variable z_i is introduced to perform implicit phasing. Specifically, when an eQTL is expected to induce allele-specific expression (ASE) at a heterozygous transcribed SNP (tSNP), the expression-increasing allele of the eQTL can reside on either haplotype. To account for this phasing uncertainty, we use the latent variable z_i. If the eQTL allele is located on the same haplotype as the alternative allele of the tSNP, we set $z_i = 1$; if it is on the same haplotype

as the reference allele, we set $z_i = -1$. The latent variable z_i is a binary indicator taking values 1 or -1, under the assumption that all individuals in the study are heterozygous for the allele of interest. Deviations from this assumption may introduce bias in the analysis.

Assume we have scRNA-seq data across multiple individuals. For a heterozygous tSNP, we assume y_{ij} to be the alternative allele read count of individual i and and cell j, and n_{ij} to be the total allele-specific read count. We assume x_{ij} is the variable we want to study, x_{ij} can be cell types, cell differentiation time, or other variables. Assume that individual i has J_i cells, and the alternative allele read count of individual i can be represented as a vector $\boldsymbol{y}_i = (y_{i1}, y_{i2}, \ldots, y_{iJ_i})$ (Figure 1). The DAESC-BB model is:

$$\begin{aligned} y_{ij} \mid n_{ij} &\sim \mathrm{BB}(n_{ij}, \mu_{ij}, \phi) \\ \log\left(\frac{\mu_{ij}}{1-\mu_{ij}}\right) &= \beta_0 + \beta_1 x_{ij} + a_i \\ a_i &\sim N(0, \sigma_a^2) \end{aligned} \tag{2.1}$$

In the formula (1), $BB(n_{ij}, \mu_{ij}, \phi)$ is the beta binomial distribution with denominator n_{ij} and mean μ_{ij}. We can find that the latent variable a_i is the same for the same individual i across all cells, which is used to account for the sample repeat structure from the same individual. If we do not introduce this latent variable a_i and consider all cells to be independent, it will result in more false positives [10]. After estimating the model parameters, we test the null hypothesis H_0: $\beta_1 = 0$ using a likelihood ratio test. Rejection of the null hypothesis indicates the presence of differential ASE across levels of the covariate x_{ij}.

Now, we can take a look at the formula of the DAESC-Mix model:

$$
\begin{aligned}
y_{ij} \mid n_{ij} &\sim \mathrm{BB}(n_{ij}, \mu_{ij}, \phi) \\
\log\left(\frac{\mu_{ij}}{1-\mu_{ij}}\right) &= z_i(\beta_0 + \beta_1 x_{ij}) + a_i \\
z_i = 2\delta_i - 1, \delta_i &\sim Bernoulli(\pi_0) \\
a_i &\sim N(0, \sigma_a^2)
\end{aligned}
\tag{2.2}
$$

Compared with the DAESC-BB formulas, the DAESC-Mix version has one more latent variable z_i (-1 or 1). We introduce a latent binary variable $z_i \in -1, 1$ for each individual, representing the unknown phase configuration. Specifically, $z_i = 1$ indicates that the expression-enhancing allele is on the same haplotype as the alternative allele at the tSNP, and $z_i = -1$ indicates it is on the opposite haplotype. By multiplying $(\beta_0 + \beta_1 x_{ij})$ with z_i, we allow the model to adjust the direction of allelic imbalance based on the latent phase. This formulation is biologically motivated and captures the intuitive notion that the direction of allelic expression bias depends on whether the regulatory allele acts on the reference or alternative allele at the tSNP. For example, consider a heterozygous individual with a C/T genotype at a tSNP, where T is the alternative allele. Suppose the expression-enhancing allele at a nearby eQTL is G. If G is phased with T, we expect to see higher expression from the T allele (positive allelic imbalance). If G is phased with C, the effect is reversed. Since we do not know this phase a priori, we model it implicitly via z_i.

Owing to the inclusion of additional latent variables z_i, the DAESC-Mix model has demonstrated greater statistical power compared to the DAESC-BB model. However, it requires a relatively large sample size for stable estimation (typically $N > 20$). When the number of samples exceeds 20, we recommend using the Mix version to achieve higher power. For smaller sample sizes, the BB version is preferred due to its more stable and robust estimation.

We use the variational EM algorithm to estimate the parameters in both of the DAESC-BB version and the DAESC-Mix version. We use the derivation of DAESC-Mix version as an

example, because the derivation of DAESC-BB version is a simplified case of DAESC-Mix version. Let's $\boldsymbol{\beta} = (\beta_0, \beta_1)^T$ as the coefficients of the variable $\boldsymbol{x}$, and a_i, δ_i as the missing variables for individual i. The complete log-likelihood function for all N individual is :

$$
\begin{aligned}
& P(\boldsymbol{y}_1, a_1, \delta_1, \ldots, \boldsymbol{y}_N, a_N, \delta_N | \boldsymbol{\beta}, \sigma_a^2, \phi, \pi_0) \\
& = \Pi_i P(\boldsymbol{y}_i, a_i, \delta_i | \boldsymbol{\beta}, \sigma_a^2, \phi, \pi_0) \\
& \propto \Pi_i \{ \pi_0 \Pi_j \frac{B(\frac{\mu_{ij1}}{\phi} + y_{ij}, \frac{1-\mu_{ij1}}{\phi} + n_{ij} - y_{ij})}{B(\frac{\mu_{ij1}}{\phi}, \frac{1-\mu_{ij1}}{\phi})} \}^{\delta_i} \\
& \times \Pi_i \{ (1-\pi_0) \Pi_j \frac{B(\frac{\mu_{ij2}}{\phi} + y_{ij}, \frac{1-\mu_{ij2}}{\phi} + n_{ij} - y_{ij})}{B(\frac{\mu_{ij2}}{\phi}, \frac{1-\mu_{ij2}}{\phi})} \}^{\delta_i} \\
& \times (\sigma_a^2)^{-\frac{1}{2}} exp(-\frac{a_i^2}{2\sigma_a^2})
\end{aligned}
\tag{2.3}
$$

where $\mu_{ij1} = logist(\beta_0 + \beta_1 x_{ij} + a_i) = \frac{exp(\beta_0 + \beta_1 x_{ij} + a_i)}{exp(\beta_0 + \beta_1 x_{ij} + a_i) + 1}$, and $\mu_{ij2} = logist(-(\beta_0 + \beta_1 x_{ij}) + a_i) = \frac{exp(-(\beta_0 + \beta_1 x_{ij}) + a_i)}{1 + exp(-(\beta_0 + \beta_1 x_{ij}) + a_i)}$, $B(,)$ is the Beta function.

In the E-step, at iteration t, we need to approximate the posterior distribution $P(a_i, \delta_i | \boldsymbol{y}_i, \boldsymbol{\beta}_{(t)}, \sigma^2_{a,(t)}, \phi_{(t)})$ for individual i using the variational inference and the mean-field approximation [1]. Specifically, we can reasonably assume that

$$
P(a_i, \delta_i | \boldsymbol{y}_i, \boldsymbol{\beta}_{(t)}, \sigma^2_{a,(t)}, \phi_{(t)}) = P(a_i | \boldsymbol{y}_i, \boldsymbol{\beta}_{(t)}, \sigma^2_{a,(t)}, \phi_{(t)}) P(\delta_i | \boldsymbol{y}_i, \boldsymbol{\beta}_{(t)}, \sigma^2_{a,(t)}, \phi_{(t)})
$$

We can approximate the two distributions separately. For the latent variable a_i, its posterior distribution can be approximated by a normal distribution as in the Laplace variational inference [1]:

$$
q(a_i) = N(\hat{a}_{i,(t)}, \hat{\sigma}^2_{a_{i,(t)}}) \tag{2.4}
$$

The posterior distribution of the latent variable $q(\delta_i)$ is:

$$q(\delta_i) = Bernoulli(\pi_{i,(t)}) \tag{2.5}$$

There, during M-step, at iteration t, the update of π_0 and σ_a has closed forms :

$$\begin{aligned} \pi_{0,(t+1)} &= \frac{1}{N}\Sigma_i \pi_{i,(t)} \\ \sigma^2_{a,(t+1)} &= \Sigma_i \hat{a}_{i,(t)} + \hat{\sigma}_{a_i,(t)} \end{aligned} \tag{2.6}$$

There is no closed-form update of $\boldsymbol{\beta}$ and ϕ. We can optimize the Q function

$$E_{q(\delta_i,a_i)}[logP(\boldsymbol{y}_1, a_1, \delta_1, \ldots, \boldsymbol{y}_N, a_N, \delta_N)]$$

by the BFGS algorithm. Broyden–Fletcher–Goldfarb–Shanno (BFGS) algorithm is a popular method for solving nonlinear optimization problems. Unlike Newton-method optimization algorithms which needs information of second derivatives of object functions, BFGS algorithm can only make use of the first-order derivatives to approximate the Hessian matrix (second-order derivative) of the object function. It has many extension, such as BFGS-B, L-BFGS, L-BFGS-B, which can use less memory or solve the bounded optimization problem. It is convenient to use BFGS algorithm in R by the function "lbfgs".

2.2 DAESC-GPU

The lack of computation efficiency of DAESC makes it challenging to analyze large single-cell data. To address this limitation, I developed DAESC-GPU, an accelerated, scalable software for single-cell differential ASE analysis to facilitate the analysis of current large-scale ASE data. DAESC-GPU enhances computational efficiency by utilizing Graphics Processing Units (GPUs). To integrate GPUs to our software, we leveraged CuPy, an open-source Python library for GPU-accelerated computing using CUDA Toolkit libraries. CuPy pro-

vides user-friendly kernel functions which allow us to write highly specialized and optimized code directly running on GPUs.

We transformed the original R codes of DAESC into Python and employed a number of techniques to accelerate the computations by CuPy. Specifically, the original R package could only analyze one gene at one time. If we wanted to analyze multiple genes, we had to use the parallel function in R, such as parallel or foreach, or use a for loop function. Regardless, it was time-consuming to analyze large single-cell data with millions of cells and thousands of genes. In order to overcome the problem, we rewrote the original algorithm in R into Python with large matrix operations. Leveraging matrix-based computation, the new software enables concurrent analysis of thousands of genes. Although we did not change the statistical model, developing this Python version involves many challenging programming problems. The most essential one was the need to design a custom optimizer based on the BFGS algorithm which can robustly solve thousands of small optimization problems by large matrix operation. To the best of our knowledge, there is no such existing package in R and Python. Our optimizer is the first software to do such task, and it could efficiently and precisely find maximum/minimum points in real-data experiments as shown in the results section below.

Another feature of DAESC-GPU was that it exploited the computational power GPUs using CuPy. We utilized element-wise kernels in CuPy to integrate multiple mathematical operations into one single specialized function written in C++. User-defined kernels in Cupy allow researchers to directly control the GPU's parallel processing units and to define task-specific complex functions, when the pre-built CuPy functions were not enough or not efficient. For instance, in the M-step, gradient of parameters was computed through a series of steps including the sigmoid function, the difference between two digamma functions, matrix multiplication, and element-wise matrix multiplication. By employing custom kernels instead

of pre-defined functions in the CuPy library, we combined these computations into one single function implemented in C++, which significantly improved the speed of our codes. In our software, almost all essential computations were performed in such complex kernels. In addition, the unique low-level control over GPUs enabled us to simplify some computationally intensive mathematical processes. For example, during E-M iterations, kernel functions enabled the computation of difference between digamma functions using finite summations. Finally, we further enhanced the efficiency of DAESC-GPU by leveraging both kernel functions and the memory pool features in CuPy. Kernel functions enabled computations to be performed directly on the original data without requiring additional memory for latent variables. The memory pool in CuPy further reduced the overhead associated with GPU memory allocation by reclaiming memory from unused variables.

2.3 Evaluation of DAESC-GPU on endoderm differentiation data

We compared the performance of DAESC-GPU, executed on T4 and A100 on Google Colab, and the original DAESC package, executed on 8 CPUs from Google Colab (80 GB RAM) and UW Biostatistics Computing Cluster (referred to as "UW Server", 96 GB RAM), respectively. In silico experiments were conducted on single-cell ASE data from an endoderm differentiation experiment [4].

Cuomo et al [4] performed an experiment investigating endoderm differentiation using 125 induced pluripotent stem cell (iPSC) lines derived from the Human Induced Pluripotent Stem Cell Initiative (HipSci). This study profiled gene expression at 4 differentiation times points using Smart-seq2. There are in total 114 individuals whose SNP-level allele-specific read counts were known, and 105 individuals' genotypes were available. We only focused on

these 105 individuals. We removed SNPs with low mappability and monoallelic expression to reduce potential genotyping error. For each gene, aggregated SNP-level ASE counts across all SNPs in the exons of the gene and obtained two haplotype-specific counts (hap1 count and hap2 count). We further removed the genes which had non-zero ASE counts in of the cells. After preprocessing, we obtained allele-specific read counts for 4,102 genes and 30,474 cells. We conducted differential ASE analysis along pseudotime to detect dynamically regulated genes during endoderm differentiation. Pseudotime was inferred and provided by the original study. [4]

Next, we evaluated the scalability of DAESC-GPU to larger datasets by simulating additional cells based on the endoderm differentiation data. First, we randomly selected 2,000 genes. To emulate realistic read depth, we used the total read counts (TRC) from the read data and kept them fixed in the simulations. To simulate datasets with more cells, we duplicated the real TRC as TRC for the additional cells. For example, we duplicated the 2000 x 30474 TRC matrix once and created a 2000 x 60948 matrix of TRC for 60,948 cells. Larger datasets were simulated by duplicating the TRC matrix more times. Alternative allele counts were simulated using the same procedure as described in previous pepr [10]. To be specific, we assumed that there was only one eQTL driving ASE and simulated the alternative allele read counts by:

$$\begin{aligned} y_{ij}|n_{ij} &\sim BB(n_{ij}, \mu_{ij}, \phi) \\ log(\frac{\mu_{ij}}{1-\mu_{ij}}) &= z_i(\beta_0 + \beta_1 x_{ij} + \beta_1 \eta_i) + a_i \\ a_i \sim N(0, \sigma_a^2), &z_i \sim categorical([-1, 1, 0], [\pi_1, \pi_2, \pi_3]) \end{aligned} \tag{2.7}$$

Unlike DAESC-Mix where there were only two possible values for latent variable z_i, in the formula above, we set three possible value. The additional value $z_i = 0$ refers to the cases that the eQTL SNP was homozygous. We simulated the haplotype proportions π_1, π_2, π_3 by the available LD coefficient (r^2) between the eQTL and the tSNP. In this project, we only

used the data with $r^2 = 1$.

Model coefficients $(\beta_0, \beta_1, \sigma^2, \phi)$ for the simulations were derived by fitting DAESC-BB on the real data and selecting 500 genes with largest $|\beta_1|$ [10]. The number of cell observations ranged from 30,474 to 1.2 million. To accommodate larger memory requirements, we used an NVIDIA A100 GPU with 40 GB RAM for data analysis. As the number of cells increased, we partitioned the genes into more groups to accommodate the limited RAM. For cells counts below 0.35 million, DAESC-GPU could fit 2,000 genes simultaneously. For cell counts between 0.35 to 0.7 million, the genes were divided into two parts and fitted sequentially. Similarly, for cell counts between 0.7 and 1.05 million, the genes were split into three groups. For cell counts exceeding 1.05 million, they were divided into four groups, with each group processed sequentially. The reported runtime was the sum of runtime of all groups. Due to the rapidly increasing runtime, we restricted the use of the R implementation of DAESC-BB to datasets with no more than 0.3 million cell observations and restricted the use of the R implementation of DAESC-Mix to datasets with no more than 0.27 million cell observations. For comparison, we utilized the R package on the UW Biostatistics Server with 8 CPUs and 96 GB RAM, and on the Google Colab with 8CPUs and 80 GB RAM.

Chapter 3

Results

To evaluate the performance of DAESC-GPU, we conducted differential ASE analysis along pseudotime using single-cell RNA-seq data (Smart-seq2) from an endoderm differentiation experiments (See Methods for more detials) [4]. After the same data pre-processing procedure as described in our earlier paper [10], we could obtain allele-specific read counts for 4,102 genes and 30,474 cells collected from 105 individuals. First, we compared DAESC-GPU, excuted on T4 and A100 on Google Colab, and the original DAESC package, excuted on 8 CPUs from Google Colab and UW Biostatistics Computing Cluster (referred to as "UW Server") respectively. On the whole, DAESC-GPU is $30 \sim 90$ times faster than DAESC R package for the scASE analysis in this dataset. Specifically, DAESC-GPU-BB used 64 minutes on T4 and 32 minutes on A100 to analyze the all 4,102 genes, while DAESC-BB model used 1,755 minutes on Colab and 1866 minutes on UW server. When we used DAESC-GPU-Mix to analyze the data, the runtime was almost twice as before, using 120 minutes on T4 GPU and 57 minutes on A100 GPU. But the runtime of DAESC-Mix was more than 2 times of runtime of DAESC-BB model to analyze the same data, where the DAESC-Mix needed 4760 minutes on Colab and 5198 minutes on UW Server. This clearly demonstrated the computational advantage of DAESC-GPU over previous DAESC model (Firgure 2A).

Second, we showed the results from DAESC-GPU were consistent with the results from DAESC. In Figure 2B, we show the key parameters (β_0, β_1) from both of DAESC and DAESC-GPU. The more identical two results are, the closer the points will be to the diagonal line. In the Figure 2B, the consistency is perfect for the BB model, since almost all points (pink points) were on the dialogue line and the corresponding Kendall coefficients were also very high, 0.972 for β_0 and 0.986 for β_1. The estimates were also consistent between DAESC-GPU-Mix and DAESC-Mix, though the correlations are lower than those of the BB model. We can observe some discrete points a little far away from the dialogue line. One possible reason was that it was challenging to estimate the implicit haplotype switching (the latent variable z_i). Although there was minor difference, the majority of points of the Mix version are on the diagonal line and the Kendall coefficients are also over 0.9. Apart from the key parameters β_0 and β_1, we also checked the consistency for other parameters σ^2, ϕ, π_0 and π_1 from the supplement materials. The estimates of σ^2 and ϕ from DAESC abd DAESC-GPU were highly similar with Kendall's tau over 0.91. The correlation was weaker for π_0 for Mix version (supplement material, figrue 5). However, this parameter was of secondary importance and does not directly impact the interpretation of the model. We will explain it next.

Third, we applied DAESC and DAESC-Mix to analyze Cuomo data [4]. We wanted to test the association between allele-specific gene expression and pseudotime. It was equivalent to say that we wanted to test whether $\beta_1 = 0$ in the formula (equation 1, equation 2). For hypothesis testing of differential ASE, DAESC-GPU and DAESC models detect nearly identical sets of dynamic ASE genes at FDR thresholds 0.01, 0.05, 0.1 (Figure 2C). We can see that for BB version, DAESC and DAESC-GPU can find almost the same set of genes, there are just 0-5 different genes. For Mix version, there is slightly more discrepancy. But the overlap is still above 90%.

Fourth, we showed the scalability of DAESC-GPU by a simulation study (See Methods section for more details about the experiments). From Figure 2D, we observed that the runtime of DAESC increased rapidly as the number of cells increased, while that of DAESC-GPU only increased moderately and did not present major challenges for analyzing over 1 million cells and 2,000 genes. For example, DAESC-BB required 757 minutes to analyze 0.3 million cells, whereas DAESC-GPU-BB completed the analysis of 1.2 million cells in only 127 minutes. Similarly, DAESC-Mix used 2,545 minutes (1.77 days) to analyze 0.27 million cells, while DAESC-GPU-Mix needed only 457 minutes to analyze 1.2 million cells. These results showed the distinct advantage in computation efficiency of DAESC-GPU against DAESC. Given the increasing number of single-cell data, DAESC-GPU had shown great potential to facilitate more biological discoveries.

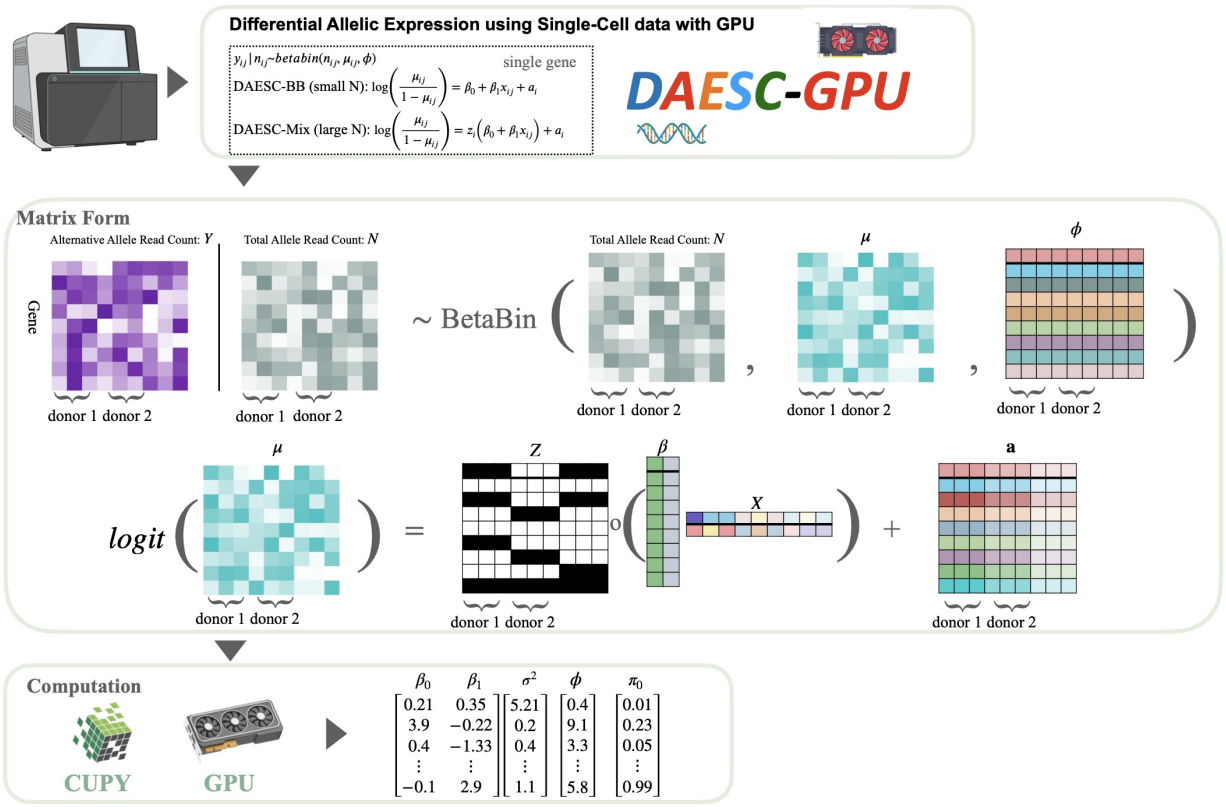

Figure 1: Schematic of DAESC-GPU

DAESC-GPU is a scalable software for single-cell differential allele-specific expression (D-ASE) analysis, leveraging large matrix computation and massive parallel computing using GPU. The mathematical sign ∘ means the element-wise matrix operation.

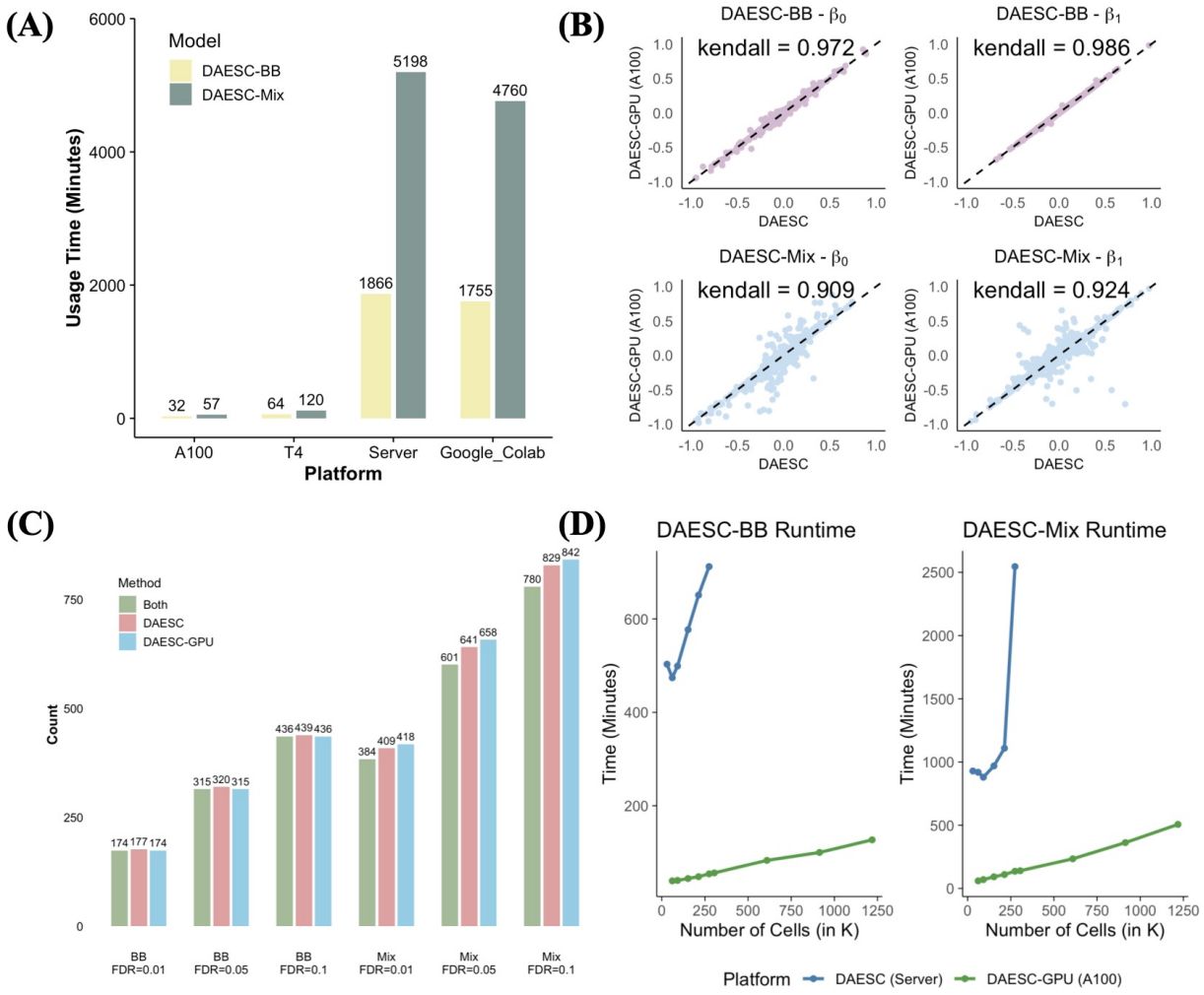

Figure 2: Performance of DAESC-GPU on endoderm differentiation data

(A) compares the runtime of DAESC-GPU (one single A100 or T4 GPU) vs DAESC R package (UW server and Google Colab, 8 CPUs); (B) compare the estimates of main model parameters between DAESC-GPU and DAESC. Each dot is a gene with DAESC estimates as x-location and DAESC-GPU estimates as y-location. Black line is the diagonal line $y = x$. Results from BB version are in red and results from Mix version are in blue. Kendall correlation of estimates between two methods is marked in the figures; (C) indicates the number of significant genes found only by DAESC, DAESC-GPU, and both. We use three common FDR thresholds 0.01, 0.05, 0.1. (D) shows the runtime of DAESC and DAESC-GPU to analyze the increasing number of simulation data.

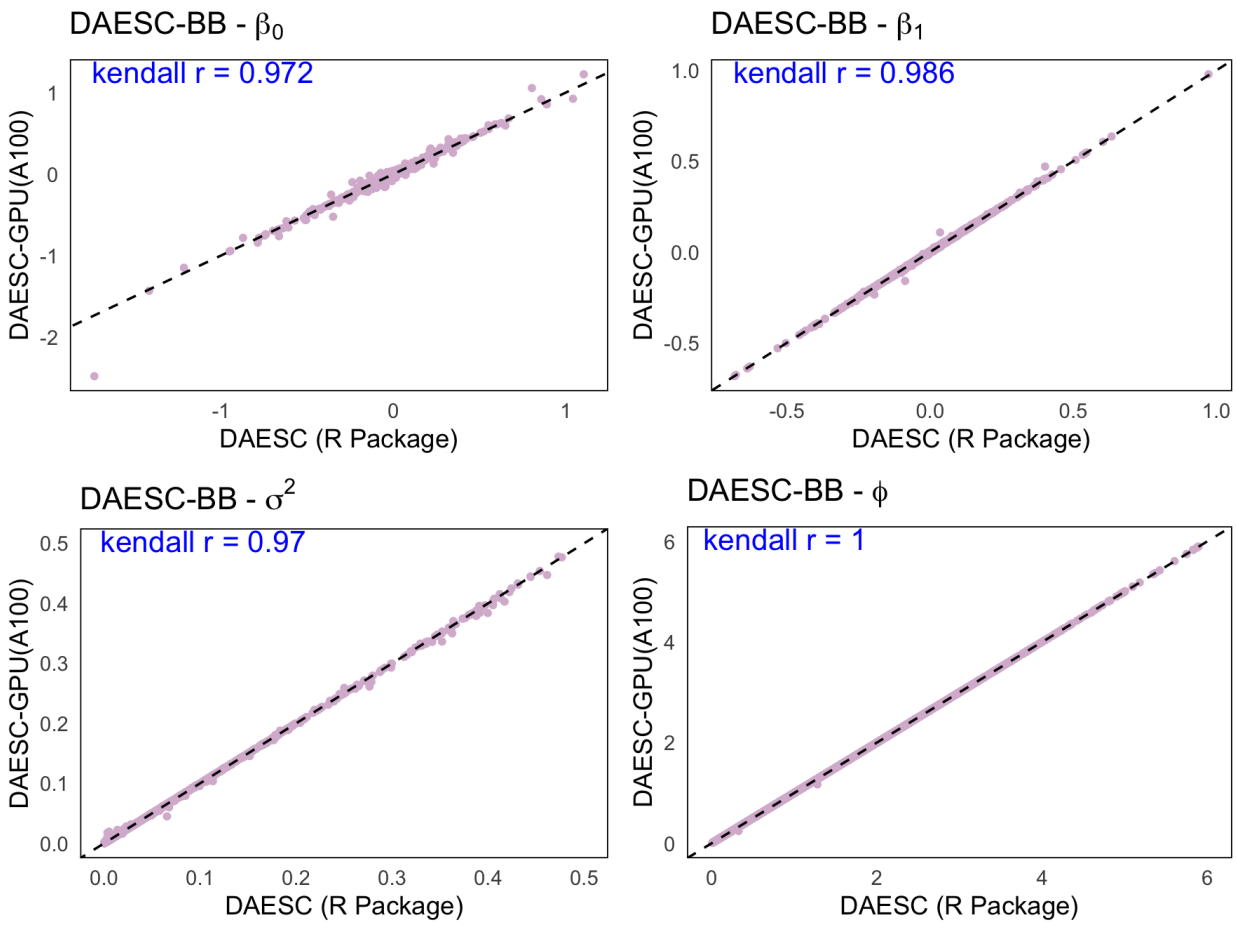

Figure 3: Compare all estimates from two softwares (BB model)

Compare the estimates of all model parameters $(\beta_0, \beta_1, \sigma^2, \phi)$ between DAESC-GPU-BB and DAESC-BB. Each dot is a gene with DAESC estimates as x-location and DAESC-GPU estimates as y-location. Black line is the diagonal line $y = x$. Kendall correlation of estimates between two methods is marked in the figures

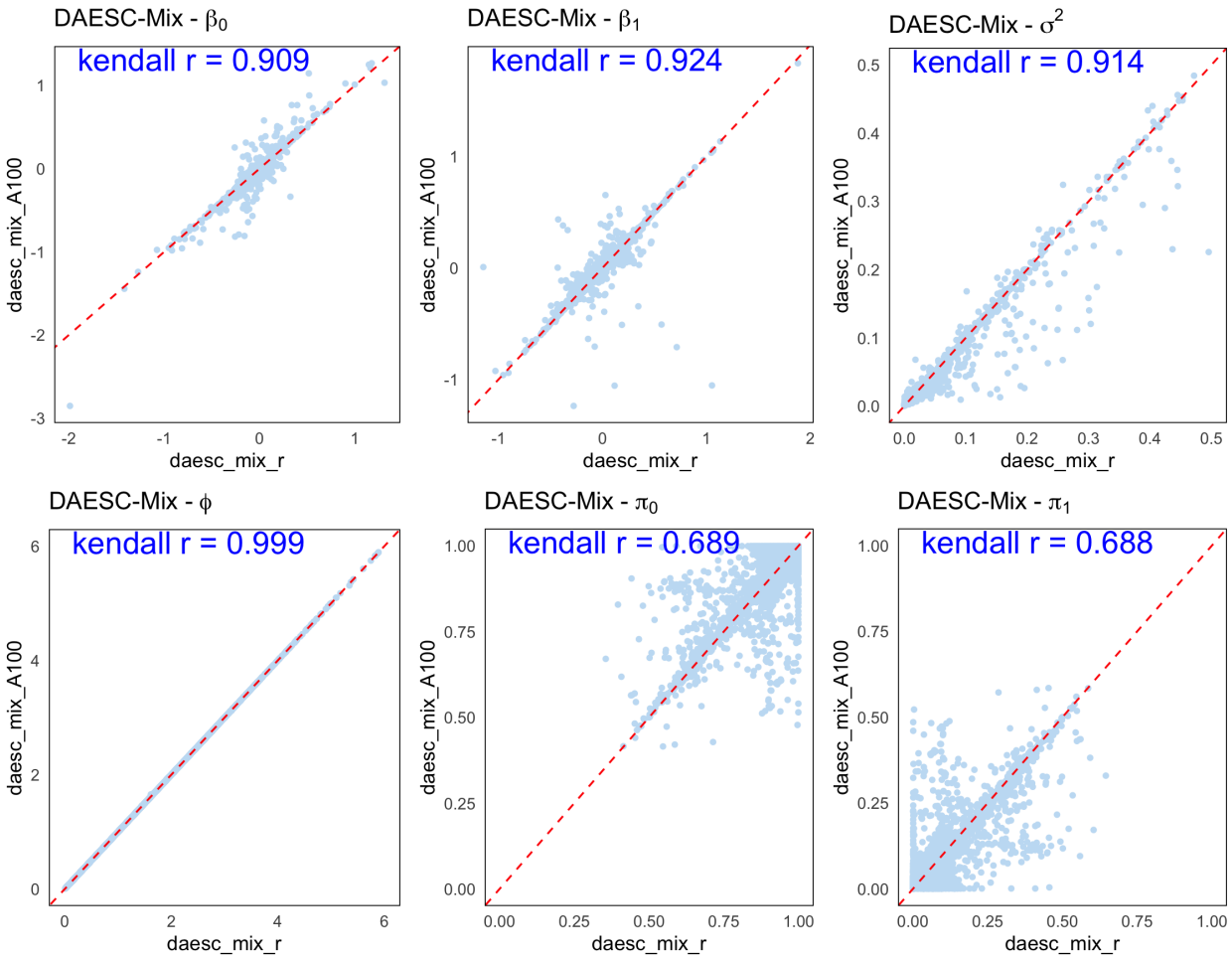

Figure 4: Compare all estimates from two softwares (Mix model)

Compare the estimates of all model parameters $(\beta_0, \beta_1, \sigma^2, \phi, \pi_0, \pi_1)$ between DAESC-GPU-Mix and DAESC-Mix. Each dot is a gene with DAESC estimates as x-location and DAESC-GPU estimates as y-location. Black line is the diagonal line $y = x$. Kendall correlation of estimates between two methods is marked in the figures

Chapter 4

Discussion

Single-cell RNA-seq (scRNA- seq) enabled the study of ASE in heterogeneous cell types within a tissue. DAESC was a powerful statistical model for the ASE analyze using scRNA-seq data from multiple individuals. DAESC-GPU was an high-performance, scalable version of DAESC. DAESC-GPU is dozens of times faster than the original R package (DAESC). The computational efficiency made it particularly advantageous for analyzing large-scale ASE data of numerous cells. In our simulation studies, we demonstrated that DAESC-GPU can analyze more than one million cells in a few hours on a single GPU. This indicated its ability to handle much larger datasets especially if multiple GPUs are available.

DAESC-GPU provided a promising and effective solution for enhancing computational efficiency across a wide array of research fields. Computational methods and software tools frequently utilized in biostatistics, computational biology, genetics, and genomics are often demanding in terms of CPU resources, thereby restricting their broader practical application. Graphics Processing Units (GPUs) emerged as an ideal alternative due to their capability to substantially reduce computational runtime compared to standard CPU-based methods. The increasing availability and affordability of GPU resources had notably popularized their usage in various research settings. Several existing GPU-based models illustrate the potential

and effectiveness of GPU acceleration. For instance, GBOOST [14] was designed to perform gene-gene interaction analyses on large genomic datasets, achieving a remarkable 40× speed-up that reduced genome-wide association study (GWAS) runtime from approximately 2.5 days to mere hours. Another notable example, Mendel-GPU [3], significantly accelerates genotype imputation tasks, cutting the required computational time from around three years to less than one week. Furthermore, FamSeq [8] leveraged GPU architecture to efficiently calculate posterior probabilities across 3^n genotype possibilities, where n represents pedigree size—a computation particularly suited to GPU parallel processing paradigms. The FamSeq method had reported a 10× speed-up, allowing variant calling from whole-genome sequencing data to be completed in approximately 36 hours, as opposed to the 16 days necessary for the equivalent CPU implementation.

A major advantage of DAESC-GPU was its compatibility with platforms such as Google Colab, which provides free and easily accessible GPU resources. Users can leverage Google Colab's complimentary NVIDIA T4 GPU, offering ten hours of free computational time—sufficient to replicate our experiments or perform various analytical tasks. Even extended usage or employment of higher-performance GPUs, such as the NVIDIA A100, was relatively affordable. The NVIDIA T4 GPU was currently priced around 0.18 USD per hour, while the NVIDIA A100 GPU costs approximately 1.18 USD per hour [6]. Although exact costs might vary slightly depending on time or location, these general rates reflected an economically viable option for researchers. Another significant benefit of using DAESC-GPU was its simplified setup and user-friendly configuration process. Many high-performance software solutions necessitated extensive and intricate environment configurations, posing a substantial barrier to users. Conversely, DAESC-GPU has been specifically designed to avoid these common environmental setup complexities, thus significantly reducing the overhead required to initiate and execute analyses.

In the subsequent phase of our work, we are developing a highly user-friendly bioinformatics pipeline designed to process raw sequencing data, such as '.sra' or '.fastq' files, and efficiently derive allele-specific gene expression. Our primary objective is to minimize the effort required by researchers, significantly enhancing the accessibility of DAESC to a broader range of single-cell datasets. The proposed end-to-end pipeline consists of three main steps. First, users are required to identify their datasets of interest from the National Center for Biotechnology Information (NCBI) and document the corresponding SRA accession numbers. These SRA accession numbers serve as input for our pipeline, which then automatically downloads all necessary datasets and supplementary resources required for single-cell allele-specific analysis, including the appropriate reference genomes and computational environments. Second, the pipeline automatically performs demultiplexing by splitting FASTQ files from the initial download and subsequently recombining reads according to individual identities. This approach addresses the common practice in current RNA sequencing experiments, where researchers often pool reads from multiple individuals into a single run and perform multiple sequencing runs. Unlike typical demultiplexing approaches that may merge data from different individuals, our method strictly aggregates reads belonging to the same individual. This design choice preserves individual-specific biological effects, crucial for downstream analyses, as discussed in further detail within the DAESC paper. Moreover, it prevents potential contamination of individual reads by reads from other individuals. Following this recombination step, the pipeline employs the Cell Ranger software to generate coordinate-sorted BAM files. Third, we adapt the SALSA pipeline, originally developed by Wilson et al. (2022), to extract single-cell allele-specific gene expression data [13]. The SALSA pipeline is a sophisticated workflow for single-cell allele-specific analysis involving multiple bioinformatics tools. Briefly, the pipeline initiates by genotyping the BAM files using GATK (version 4.2.0.0), performing base recalibration with BaseRecalibrator, and variant discovery using HaplotypeCaller. Subsequently, genotypes are phased with Shapeit, leveraging the phased 1000 Genomes reference panel for biallelic single nucleotide variants

(SNVs) and insertion-deletion (indel) variants mapped to the GRCh38 reference genome. Variant-aware realignment is then conducted using the WASP pipeline (version 0.3.4) in combination with either BWA (version 0.7.17) or STAR (version 2.5.1b). Following realignment, variant filtration is performed using bcftools. Lastly, allele-specific counts are computed through GATK's ASEReadCounter module, generating single-cell allele-specific expression profiles based on phased heterozygous SNVs [13]. Our pipeline closely adheres to this established procedure, with one notable modification: we replaced Shapeit4.2 with the latest Shapeit5 version during the genotype phasing step. Shapeit5 offers improved computational efficiency, significantly reducing runtime and memory consumption, thereby enhancing the scalability and practicality of allele-specific expression analyses.

Our work has a few limitations. DAESC-GPU accelerates computational performance but does not enhance the underlying statistical model. The primary limitation of the current model is its inability to integrate information across multiple cell types. Nonetheless, we believe that high-performance software capable of conducting basic differential ASE analyses remains highly valuable. Additionally, GPU memory constraints can restrict DAESC-GPU's performance. GPU memory is limited (e.g., only 40 GB or 80 GB on an A100 GPU), whereas CPU-based systems can utilize several hundred GBs of memory. Consequently, when analyzing large-scale scASE datasets with DAESC-GPU, not all genes can be processed simultaneously, slightly increasing runtime. However, large datasets can be partitioned into smaller segments and analyzed sequentially, still resulting in computational speeds over ten times faster than CPU-based software. Another potential approach is employing multiple GPUs concurrently, further accelerating analyses. For instance, Google Colab allows the use of three GPUs simultaneously across separate sessions for a minimal fee of approximately 10 dollars, effectively reducing total runtime by one-third. Users also have the option to rent more advanced GPUs through platforms such as Amazon EC2.

Chapter 5

Conclusion

In conclusion, DAESC-GPU is powerful software for single-cell differential ASE analysis. Its application can reveal novel biological insights from the rapidly expanding allele-specific expression datasets.

Chapter 6

Appendix

Custom Optimizer:

For parameter estimation, DAESC-GPU requires an optimizer to maximize the Q-function during each EM iteration. In the original DAESC implementation, the standard BFGS optimizer in R was used, which is capable of optimizing a single function at a time. While this approach is sufficient for the original DAESC, it is inadequate for DAESC-GPU, which needs to analyze multiple genes concurrently. Specifically, DAESC-GPU requires an optimizer capable of optimizing multiple independent objective functions simultaneously. As no such optimizer is readily available in either R or Python, we developed a custom optimizer tailored to this task.

The overall idea is straightforward. We primarily adopt the classical BFGS algorithm. However, for each variable, we introduce an additional dimension to incorporate information from all objective functions. Specifically, we aim to simultaneously optimize multiple independent functions, denoted by $\boldsymbol{f^*} = (f_1, f_2, \ldots, f_m)$, where each $f_i : \mathbb{R}^n \to \mathbb{R}$ for $i = 1, 2, \ldots, m$. For each function f_i, we use the BFGS algorithm, at iteration k:

- Get a direction $p_{k,i} = -H_{k,i}\Delta f_i(x_{k,i}), p_{k,i} \in R^n$
- Perform line search to find the appropriate step size $\alpha_{k,i} \in R^n$ along the direction $p_{k,i}$,

the step size must satisfy the Wolf condition

- Update parameter $x_{k+1,i} = x_{k,i} + s_{k,i}$ $s_{k,i} = \alpha_{k,i} p_{k,i}$
- Get the difference $y_{k,i} = \Delta f_i(x_{k+1,i}) - \Delta f_i(x_{k,i})$
- Update the approximate inverted Hessian matrix: $H_{k+1,i} = H_{K,i} + \frac{(s_{k,i}^T y_{k,i} + y_{k,i}^T H_{k,i} y_{k,i})(s_{k,i} s_{k,i}^T)}{(s_{k,i}^T y_{k,i})^2} - \frac{H_{k,i} y_{k,i} s_{k,i}^T + s_{k,i} y_{k,i}^T H_{k,i}}{(s_{k,i}^T y_{k,i})}$

We utilize three-dimensional matrices and element-wise matrix operations to ensure that each function independently undergoes the BFGS algorithm as described above. While each function is optimized independently in principle, the structured use of matrix operations enables all functions to be optimized simultaneously. This parallelization significantly reduces computational time compared to sequential optimization approaches. The defaul set of the maximum number of line search is 15 and the default set of the maximum iteration number of BFGS algorithm is 10. People can increase the number for more precise results or decrease the number for shorter runtime.

Bibliography

[1] David M Blei, Alp Kucukelbir, and Jon D McAuliffe. Variational inference: A review for statisticians. *Journal of the American statistical Association*, 112(518):859–877, 2017.

[2] Stephane E Castel, François Aguet, Pejman Mohammadi, Kristin G Ardlie, and Tuuli Lappalainen. A vast resource of allelic expression data spanning human tissues. *Genome biology*, 21:1–12, 2020.

[3] Gary K Chen, Kai Wang, Alex H Stram, Eric M Sobel, and Kenneth Lange. Mendel-gpu: haplotyping and genotype imputation on graphics processing units. *Bioinformatics*, 28(22):2979–2980, 2012.

[4] Anna SE Cuomo, Daniel D Seaton, Davis J McCarthy, Iker Martinez, Marc Jan Bonder, Jose Garcia-Bernardo, Shradha Amatya, Pedro Madrigal, Abigail Isaacson, Florian Buettner, et al. Single-cell rna-sequencing of differentiating ips cells reveals dynamic genetic effects on gene expression. *Nature communications*, 11(1):810, 2020.

[5] William J Dally, Stephen W Keckler, and David B Kirk. Evolution of the graphics processing unit (gpu). *IEEE Micro*, 41(6):42–51, 2021.

[6] Chris McCormick. Colab gpus – features and pricing, 2024. Accessed: 2025-04-07.

[7] Kyle A O'Connell, Zelaikha B Yosufzai, Ross A Campbell, Collin J Lobb, Haley T Engelken, Laura M Gorrell, Thad B Carlson, Josh J Catana, Dina Mikdadi, Vivien R

Bonazzi, et al. Accelerating genomic workflows using nvidia parabricks. *BMC bioinformatics*, 24(1):221, 2023.

[8] Gang Peng, Yu Fan, and Wenyi Wang. Famseq: a variant calling program for family-based sequencing data using graphics processing units. *PLoS computational biology*, 10(10):e1003880, 2014.

[9] Celine L St Pierre, Juan F Macias-Velasco, Jessica P Wayhart, Li Yin, Clay F Semenkovich, and Heather A Lawson. Genetic, epigenetic, and environmental mechanisms govern allele-specific gene expression. *Genome research*, 32(6):1042–1057, 2022.

[10] Guanghao Qi, Benjamin J Strober, Joshua M Popp, Rebecca Keener, Hongkai Ji, and Alexis Battle. Single-cell allele-specific expression analysis reveals dynamic and cell-type-specific regulatory effects. *Nature communications*, 14(1):6317, 2023.

[11] Stefano Rosati. Comparison of cpu and parabricks gpu enabled bioinformatics software for high throughput clinical genomic applications. Master's thesis, Marquette University, 2020.

[12] Nicole A Teran, Daniel C Nachun, Tiffany Eulalio, Nicole M Ferraro, Craig Smail, Manuel A Rivas, and Stephen B Montgomery. Nonsense-mediated decay is highly stable across individuals and tissues. *The American Journal of Human Genetics*, 108(8):1401–1408, 2021.

[13] Parker C Wilson, Yoshiharu Muto, Haojia Wu, Anil Karihaloo, Sushrut S Waikar, and Benjamin D Humphreys. Multimodal single cell sequencing implicates chromatin accessibility and genetic background in diabetic kidney disease progression. *Nature communications*, 13(1):5253, 2022.

[14] Ling Sing Yung, Can Yang, Xiang Wan, and Weichuan Yu. Gboost: a gpu-based tool for detecting gene–gene interactions in genome–wide case control studies. *Bioinformatics*, 27(9):1309–1310, 2011.