BMC Bioinformatics

SOFTWARE **Open Access**

FastQTLmapping: an ultra-fast and memory efficient package for mQTL-like analysis

Xingjian Gao[1], Jiarui Li[2], Xinxuan Liu[3], Qianqian Peng[2], Han Jing[2], Sibte Hadi[4], Andrew E. Teschendorff[3], Sijia Wang[3,5,6] and Fan Liu[4*]

*Correspondence: fliu@nauss.edu.sa

[1] National Clinical Research Center of Kidney Diseases, Jinling Hospital, Nanjing, Jiangsu, China
[2] CAS Key Laboratory of Computational Biology, Shanghai Institute of Nutrition and Health, University of Chinese Academy of Sciences, Chinese Academy of Sciences, Shanghai, China
[3] Key Laboratory of Genomic and Precision Medicine, Beijing Institute of Genomics, China National Center for Bioinformation, Chinese Academy of Sciences, Beijing, China
[4] Department of Forensic Sciences, College of Criminal Justice, Naif Arab University of Security Sciences, 11452 Riyadh, Kingdom of Saudi Arabia
[5] Taizhou Institute of Health Sciences, Fudan University, Taizhou, Jiangsu, China
[6] Center for Excellence in Animal Evolution and Genetics, Chinese Academy of Sciences, Kunming, China

Abstract

Background: FastQTLmapping addresses the need for an ultra-fast and memory-efficient solver capable of handling exhaustive multiple regression analysis with a vast number of dependent and explanatory variables, including covariates. This challenge is especially pronounced in methylation quantitative trait loci (mQTL)-like analysis, which typically involves high-dimensional genetic and epigenetic data.

Results: FastQTLmapping is a precompiled C++ software solution accelerated by Intel MKL and GSL, freely available at https://github.com/Fun-Gene/fastQTLmapping. Compared to state-of-the-art methods (MatrixEQTL, FastQTL, and TensorQTL), fastQTLmapping demonstrated an order of magnitude speed improvement, coupled with a marked reduction in peak memory usage. In a large dataset consisting of 3500 individuals, 8 million SNPs, 0.8 million CpGs, and 20 covariates, fastQTLmapping completed the entire mQTL analysis in 4.5 h with only 13.1 GB peak memory usage.

Conclusions: FastQTLmapping effectively expedites comprehensive mQTL analyses by providing a robust and generic approach that accommodates large-scale genomic datasets with covariates. This solution has the potential to streamline mQTL-like studies and inform future method development for efficient computational genomics.

Keywords: FastQTLmapping, MQTL, Multiple regression, Genomic analysis, Memory efficiency, High-dimensional data

Introduction

Methylation quantitative trait locus (mQTL) analysis evaluates associations between expansive sets of genomic variants and CpG sites across the genome. Ideally, such studies are conducted in large population samples to ensure sufficient statistical power, incorporate covariates to control confounding, and proceed exhaustively to maximize genome-wide resolution. However, the sheer scale of these analyses, often involving trillions of regressions, remains a major computational challenge. Existing tools, such as MatrixEQTL [1] and FastQTL [2], have substantially increased computational throughput on CPU-based systems, yet continue to face constraints related to efficiency, I/O speed, and memory management. More recent GPU-based solutions, including

TensorQTL [3], require large amounts of VRAM, which can be prohibitive for typical individual users.

Here, we present fastQTLmapping, a novel framework designed to perform ultra-fast, memory-efficient, and exact exhaustive multiple-regression analyses under standard CPU settings. By efficiently handling exceptionally large numbers of dependent and explanatory variables (as well as covariates), fastQTLmapping is ideally suited for mQTL studies and similarly high-dimensional genomic investigations.

Implementation

FastQTLmapping is deployed on Linux using MKL (https://software.intel.com/tools/onemkl) and GSL (http://www.gnu.org/software/gsl/) library and is run from the command line. C++ source code, a demo run, and full documentations are freely available at https://github.com/Fun-Gene/fastQTLmapping. FastQTLmapping accepts input files in text format and in PLINK binary format and outputs in the text format the test statistics of all regressions with the ability to control the volume at preset significance thresholds. Different thresholds can be specified for cis-, long cis- and trans-analysis. Z- and rank-normalizations are optionally available for pre-processing certain or all input variables. Prior to the mapping analysis, fastQTLmapping preliminarily assesses the data and provides estimates for the computational loads and recommendations for memory control.

We use a two-step design to accelerate the computation while ensuring exact results, i.e., an initial screening under a relaxed threshold followed by naïve multiple regressions. The initial screening is the most computationally burdensome and is accelerated using fine- and coarse-grained strategies. The fine-grained acceleration involves imputation of missing values by mean, calculation under single floating-point precision, and projecting dependent and explanatory variables to the orthonormal basis of covariates. The latter algorithm has long existed and is widely used, such as in MatrixEQTL, FastQTL, and TensorQTL. In brief, consider multiple regression $y = \alpha + \beta x + \boldsymbol{\gamma C} + \epsilon$, where $\boldsymbol{C}$ is the matrix of k covariates, decompose $\boldsymbol{C}$ using QR factorization $\boldsymbol{C} = \boldsymbol{QR}$, then centralize $\boldsymbol{Q}$ to $\hat{\boldsymbol{Q}}$, where $\hat{\boldsymbol{Q}}$ is the orthonormal basis. Then subtract the projections of x and y on $\hat{\boldsymbol{Q}}$ to get x' and y'. $\hat{\boldsymbol{Q}} = [q_1 q_2 \ldots q_k]$; $x' = x - \sum_{i=1}^{k} \langle x, q_i \rangle q_i$; $y' = y - \sum_{i=1}^{k} \langle y, q_i \rangle q_i$. The subsequent analyses only involve univariate regressions $y' = \alpha' + \beta' x' + \epsilon'$ to test $H_0 : \beta' = 0$ with k less degrees of freedom, which were solved using Pearson product-moment correlations involving only vectorized operations and were accelerated using the Math Kernel Library (MKL). The regressions with β' surviving a critical value that corresponds to a relaxed p-value threshold are further passed to the 2nd step naïve multiple regression analysis, where missing values are handled as is and double floating-point precision is used.

The coarse-grained acceleration was achieved using OpenMP parallelization framework. Via dynamically splitting data into chunks and fast I/O of intermediate variables, we achieved excellent memory control in such a way that the peak memory is defined only by the number of variables in the largest chunk (Figure S1). FastQTLmapping can estimate the peak memory prior to the initial screening where the max chunk size can be user-defined. We developed a parser for converting strings and plink binary files to floating-point numbers for fast I/O.

The Storey and Tibshirani False Discovery Rate (ST-FDR) [4] approach is utilized for multiple testing correction. We introduced a rapid and memory-efficient ST-FDR algorithm suitable for large-scale analysis, circumventing the *pi0estimate* function in R *qvalue* package. This is achieved by employing counters for T-statistics binning across predefined intervals while keeping memory clean. Determining which counter the current T-statistic belongs to is the speed limit step, since it is the innermost loop and the complexity is proportional to the number of counters. The speeding-up is achieved by binary search within a preloaded code register delineating counter boundaries. The estimation of pi0, the fraction of true null p-values, is refined using a C++ reimplementation of Storey and Tibshirani's bootstrap algorithm [5]. In the second step, pi0 is used to adjust the q-values in naïve multiple regression via traditional FDR methods. All FDR analyses can be separately carried out for cis-, long-cis-, and trans-analyses according to user-defined thresholds of base-pair distances.

To facilitate broader application, we provide a precompiled executable that includes statically linked MKL and GSL libraries—compatible with lower versions of gcc/glibc and minimizing external dependencies—along with source code for users who prefer to perform their own installation.

Performance

We began by evaluating FastQTLmapping against an established QTL analysis software on the GSE40364 dataset [6], which comprises genotype and gene expression data from 88 samples. After preprocessing and quality control, 6,884 genes and 268,653 SNPs remained, with a transcriptome missing rate of 0.46%. We included ten genomic PCs and ten expression PCs as covariates and applied both methods using a $P < 1 \times 10^{-6}$ significance threshold. The resulting high correlation ($R = 0.996$) between the two sets of p-values underscores the accuracy and robustness of FastQTLmapping in recapitulating findings from established QTL analysis tools when the missing rate is negligible (Fig. S2).

We evaluated the performance of fastQTLmapping, MatrixEQTL, FastQTL, and TensorQTL. To ensure comparability, MatrixEQTL was run under R with parallel MKL, and FastQTL was tested with manually split input data. The benchmark dataset was derived from the GEO database (SNP data from GSE79254 and CpG data from GSE79144), as previously described [7]. Starting from 54 individuals, 1.5 million SNPs, and 450,000 CpGs, we performed resampling of individuals and SNPs to generate three test sets of 1 billion, 10 billion, and 100 billion SNP–CpG pairs for 1,000 individuals.

All mQTL analyses were performed using 1, 4, and 16 threads without covariates, on a Xeon E5 2686 V4 CPU (18 cores, 2.3 GHz), equipped with 256 GB ECC DDR4 RAM, running CentOS Linux 7, g++4.8.5, R 4.0.3, and MKL 2021.3.0. In the absence of missing data, the four tools produced identical results across all scenarios. However, with missing data, fastQTLmapping remained accurate, whereas MatrixEQTL, FastQTL, and TensorQTL showed significant deviations at higher significance thresholds (Fig. S3).

In terms of computational speed, fastQTLmapping consistently outperformed MatrixEQTL, FastQTL, and TensorQTL under every tested condition. While all four methods exhibited linear runtime scaling with data size in serial execution, fastQTLmapping was 1.7–1.9 ×faster than MatrixEQTL in single-threaded mode, 3.6–4.0 ×faster at four threads, and 6.7–10.0 ×faster at sixteen threads. It also exceeded FastQTL by 4.7–6.2 ×and TensorQTL by 1.2–4.2 ×(Fig. 1A).

For I/O speed, fastQTLmapping was slower than MatrixEQTL and TensorQTL only when processing the largest dataset in serial mode (506.0 s vs. 216.3 s), primarily due to initial data binarization and variable normalization. In all other cases, fastQTLmapping's I/O throughput surpassed MatrixEQTL by 2.1–28.3×, FastQTL by 1.9–13.0×, and TensorQTL by 1.2–13.3 ×(Fig. 1B). Peak memory usage in fastQTLmapping scaled linearly with the degree of parallelism and the volume of significant results, but remained below that of MatrixEQTL, FastQTL, and TensorQTL under every tested condition—achieving 1.7–29.1 ×, 1.1–14.7 ×, and 3.3–168.1 ×reductions, respectively (Fig. 1C). Notably, TensorQTL's memory usage peaks at the output phase and is driven purely by dataset size, whereas fastQTLmapping minimizes memory consumption throughout input and output via data chunking (Fig. 1C).

The capabilities of fastQTLmapping were further demonstrated in a large-scale mQTL investigation [8] involving 6.14 trillion SNP–CpG associations across 7.58 million SNPs and 811,876 CpGs from 3523 Han Chinese individuals. Adjusted for age, sex, bisulfite slide number, bisulfite array position, estimated blood cell fractions (B cells, $CD4^+$ and $CD8^+$ T cells, NK cells, monocytes), and the top ten genomic principal components, the analysis—run on 64 threads—completed in 4.5 h (including I/O and ST-FDR procedures) and recorded a peak memory usage of 13.1 GB.

Discussion

In this work, we introduced FastQTLmapping to address the substantial computational demands of large-scale mQTL analyses. Although mQTL analyses are among the most computationally intensive within the broader family of QTL studies (eQTL, pQTL, sQTL, caQTL), FastQTLmapping is broadly applicable to any QTL-like dataset or pairwise regression scenario, provided the input is in standard text format (with decimal values) or PLINK's binary format for SNP data. By leveraging MKL and GNU GSL, the software delivers high-throughput data processing while maintaining a user-friendly command-line interface, comprehensive documentation, and multiple modes for exploratory analysis. A key feature is the user-defined data chunking strategy, which allows memory usage to be finely tuned according to hardware constraints. Additionally, integration of a fast, memory-efficient ST-FDR algorithm [9] enhances the software's capacity for analyzing large genomic datasets.

We benchmarked FastQTLmapping against the CPU version of TensorQTL under identical hardware and parameter configurations. The CPU-based comparison arose from limited GPU resources and the inherent difficulties in equitably assessing CPU- vs. GPU-oriented performance with respect to efficiency, power consumption, and cost. Notably, TensorQTL was challenging to install: initial attempts using Conda and YAML configuration files failed, and it was eventually installed through separate pip and Bioconductor steps (with rpy2 dependencies remaining unresolved). Under these conditions, FastQTLmapping consistently outperformed TensorQTL in terms of both speed and memory utilization across varied data scales and CPU thread counts.

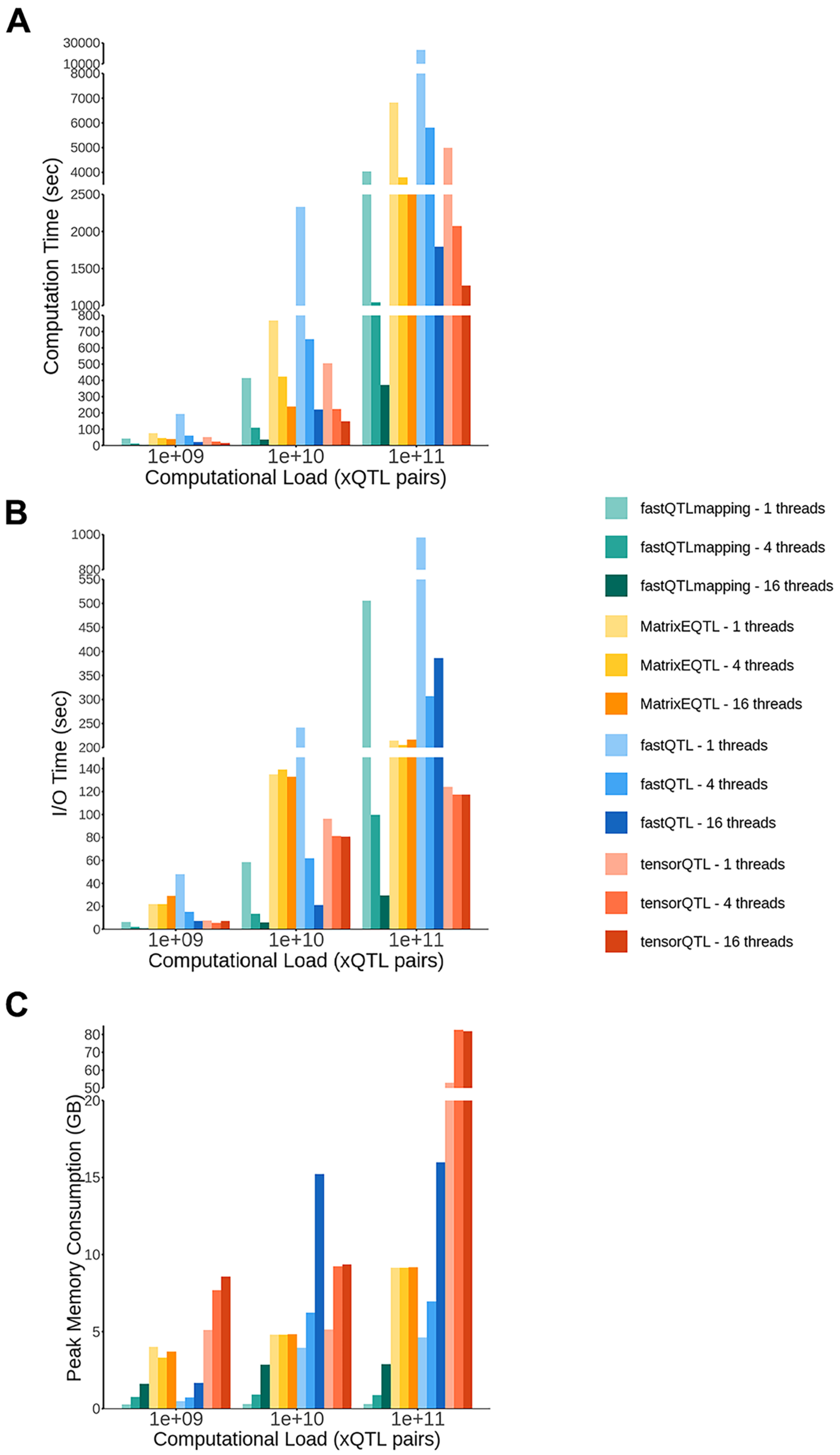

Fig. 1 Performance of fastQTLmapping and MatrixEQTL under various settings. **A** Computation time; **B** I/O time; **C**. Peak memory consumption

While TensorQTL may offer advantages for users with access to robust GPU resources, FastQTLmapping's simpler installation process and strong overall performance make it a practical choice for most mQTL analyses—particularly those involving up to tens of thousands of samples.

Despite these strengths, current development efforts for FastQTLmapping have principally centered on computational optimization, assuming well-validated datasets. Although two normalization options (Z- and rank-normalization) are provided to streamline preprocessing, more advanced data manipulation and quality control (e.g., log transformations, combined normalization approaches, outlier handling, missing-value imputation, Hardy–Weinberg equilibrium filtering, and minor allele frequency cutoffs) are not yet fully automated. Consequently, we recommend that users perform genotype imputation, remove low-frequency variants, and exclude molecular phenotypes with minimal variance prior to applying FastQTLmapping's internal normalization features.

Looking ahead, we plan to expand FastQTLmapping to better accommodate regional analyses, including the development of a window-based permutation function for automated region-specific threshold determination. We also aim to introduce customizable functions for data transformation and quality control to allow seamless integration of diverse preprocessing workflows. Notably, FastQTLmapping was recently benchmarked in a large-scale mQTL study comprising 6.14 trillion SNP–CpG associations across 3,523 samples [8], completing the entire analysis in fewer than 5 h on standard laboratory hardware. Our chunking approach ensures that both computation and I/O time grow linearly with increasing data size, while peak memory usage remains stable; for a given dataset, increasing the number of CPU threads confers near-linear speed gains with proportional rises in memory. Users can thus adjust the chunk size parameter to optimize resource allocation for diverse computational environments.

Conclusions

FastQTLmapping is ultra-fast, memory efficient, easy-to-deploy, capable for conducting pair-wise regression analysis at extraordinarily large scales on regular servers, particularly helpful for well-sized mQTL-like studies.

Availability and requirements

Project name: fastQTLmapping.

Project home page: https://github.com/Fun-Gene/fastQTLmapping

Operating system(s): Linux.

Programming language: C++

Other requirements: GLIBCXX(> = 3.4.19), CXXABI(> = 1.3.7).

License: GNU GPL-3.0.

Any restrictions to use by non-academics: There are no further restrictions on non-academic usage beyond those specified in the GNU GPL-3.0 license.

Abbreviations

mQTL	Methylation quantitative trait loci
eQTL	Expression quantitative trait loci
pQTL	Protein quantitative trait loci
sQTL	Splicing quantitative trait loci
caQTL	Chromatin accessibility quantitative trait loci

SNP	Single nucleotide polymorphism
MKL	Intel math kernel library
GSL	GNU scientific library
QR factorization	An algorithm for decomposing a matrix into an orthogonal (or unitary) matrix *Q* and an upper triangular matrix *R*
I/O	Input/output
FDR	False discovery rate
GEO	Gene expression omnibus
CPU	Central processing unit
GPU	Graphics processing unit
RAM	Random access memory
VRAM	Video random access memory
NK	Natural killer
PC	Principal component
GB	Gigabyte
gcc	GNU compiler collection
glibc	GNU C library
Yml	Yet another markup language
Pip	Pip installs packages

Supplementary Information

The online version contains supplementary material available at https://doi.org/10.1186/s12859-025-06130-3.

Additional file 1

Acknowledgements
None declared.

Author contributions
Conceptualization: FL, XG. Data curation: XG, JL, XL, QP, HJ, SH. Formal analysis: XG, JL, XL. Funding acquisition: FL, SW. Supervision: FL, SW, AET. Visualization: XG. Writing-original draft: XG, FL. Final approval: XG, JL, XL, QP, HJ, SH, AET, SW, FL.

Funding
FL was supported by Naif Arab University for Security Sciences (NAUSS-23-R18). XG was supported by Jiangsu Funding Program for Excellent Postdoctoral Talent Grant from the Department of Human Resources and Social Security of Jiangsu Province (2023ZB686). SW was supported by the National Key Research and Development Project (Grant No. 2018YFC0910403), the Max Planck-CAS Paul Gerson Unna Independent Research Group Leadership Award. This work was supported by the Strategic Priority Research Program of Chinese Academy of Sciences (Grant No. XDB38010400, XDB38010100, XDC01000000), Shanghai Municipal Science and Technology Major Project (Grant No. 2017SHZDZX01), National Natural Science Foundation of China (NSFC) (81930056, 91631307), Science and Technology Service Network Initiative of Chinese Academy of Sciences (KFJ-STS-QYZD-2021-08-001, KFJ-STS-ZDTP-079).

Availability of data and materials
FastQTLmapping source code, a demo run, and full documentations: https://github.com/Fun-Gene/fastQTLmapping. GEO data for performance testing: https://www.ncbi.nlm.nih.gov/geo/ (SNP data from GSE79254 and CpG data from GSE79144).

Declarations

Ethics approval and consent to participate
Not applicable.

Consent for publication
Not applicable.

Competing interests
The authors declare no competing interests.

Received: 19 December 2024 Accepted: 2 April 2025
Published online: 27 April 2025

References
1. Shabalin AA. Matrix eQTL: ultra fast eQTL analysis via large matrix operations. Bioinformatics. 2012;28(10):1353–8.
2. Ongen H, Buil A, Brown AA, Dermitzakis ET, Delaneau O. Fast and efficient QTL mapper for thousands of molecular phenotypes. Bioinformatics. 2016;32(10):1479–85.
3. Taylor-Weiner A, Aguet F, Haradhvala NJ, Gosai S, Anand S, Kim J, Ardlie K, Van Allen EM, Getz G. Scaling computational genomics to millions of individuals with GPUs. Genome Biol. 2019;20(1):228.

4. Storey JD. A direct approach to false discovery rates. J R Stat Soc Ser B. 2002;64(3):479–98.
5. Storey JD, Taylor JE, Siegmund D. Strong control, conservative point estimation and simultaneous conservative consistency of false discovery rates: a unified approach. J R Stat Soc Ser B. 2004;66(1):187–205.
6. Gao C, Tignor NL, Salit J, Strulovici-Barel Y, Hackett NR, Crystal RG, Mezey JG. HEFT: eQTL analysis of many thousands of expressed genes while simultaneously controlling for hidden factors. Bioinformatics. 2014;30(3):369–76.
7. Do C, Lang CF, Lin J, Darbary H, Krupska I, Gaba A, Petukhova L, Vonsattel JP, Gallagher MP, Goland RS, et al. Mechanisms and disease associations of haplotype-dependent allele-specific DNA methylation. Am J Hum Genet. 2016;98(5):934–55.
8. Peng Q, Liu X, Li W, Jing H, Li J, Gao X, Luo Q, Breeze CE, Pan S, Zheng Q, et al. Analysis of blood methylation quantitative trait loci in East Asians reveals ancestry-specific impacts on complex traits. Nat Genet. 2024;56(5):846–60.
9. Storey JD, Tibshirani R. Statistical significance for genomewide studies. Proc Natl Acad Sci USA. 2003;100(16):9440–5.

Dr. Fan Liu is a full professor at Naif Arab University for Security Sciences. Previously, he was a professor at the Beijing Institute of Genomics, Chinese Academy of Sciences, and an assistant professor at Erasmus University Medical Center. His research focuses on forensic DNA phenotyping, molecular epidemiology, and human appearance omics. He has developed pigmentation and age estimation models applied in forensic settings worldwide. Dr. Liu has published over 100 SCI papers with more than 4000 citations (H-index: 35), including over 50 lead-author articles in top journals such as Nature Genetics and Nature Communications. Over 10 of his papers attracted significant international media attention. He was included in the 2023 Stanford/Elsevier top 2% list of leading scientists in the legal medicine subfield.