MTBseq-nf: Enabling Scalable Tuberculosis Genomics "Big Data" Analysis through a User-Friendly Nextflow Wrapper for MTBseq pipeline.

Abhinav Sharma*[1], Davi Josué Marcon*[2], Johannes Loubser[1], Karla Valéria Batista Lima[2,3], Gian van der Spuy#[1], and Emilyn Costa Conceição#[1]

[1]DSI-NRF Centre of Excellence for Biomedical Tuberculosis Research; SAMRC Centre for Tuberculosis Research; Division of Molecular Biology and Human Genetics, Faculty of Medicine and Health Sciences, Stellenbosch University, Cape Town, South Africa.

[2]Universidade do Estado do Pará, Instituto de Ciências Biológicas e da Saúde, Pós-Graduação em Biologia Parasitária na Amazônia, Belém, Pará, Brazil

[3]Instituto Evandro Chagas, Seção de Bacteriologia e Micologia, Ananindeua, Pará, Brazil

April 27, 2025

Author Note

Abhinav Sharma* https://orcid.org/0000-0002-6402-6993

Davi Josué Marcon* https://orcid.org/0000-0003-1014-422X

Johannes Loubser https://orcid.org/0000-0002-2362-0187

Karla Valéria Batista Lima https://orcid.org/0000-0001-5807-0392

Gian van der Spuy# https://orcid.org/0000-0002-9067-5903

Emilyn Costa Conceição# https://orcid.org/0000-0002-7445-6620

* => Equal contributions as first authors.

# => Equal contributions as supervisors.

Author roles were classified using the Contributor Role Taxonomy (CRediT; https://credit.niso.org/) as follows: *Abhinav Sharma**: conceptualization, funding acquisition, investigation, methodology, project administration, software, validation, writing – original draft, writing – review & editing, and visualization. *Davi Josué Marcon**: writing – original draft, software, investigation, and writing – review & editing. *Johannes Loubser*: writing – review & editing and validation. *Karla Valéria Batista Lima*: validation, writing – review & editing, and resources. *Gian van der Spuy#*: visualization, conceptualization, and supervision (equal). *Emilyn Costa Conceição#*: supervision (equal), visualization, and conceptualization

Correspondence concerning this article should be addressed to Abhinav Sharma: abhi18av@outlook.com

Abstract

The MTBseq pipeline, published in 2018, was designed to address bioinformatics challenges in tuberculosis research using whole-genome sequencing data. It was the first publicly available pipeline on Github to perform full analysis of whole-genome sequencing (WGS) data for *Mycobacterium tuberculosis* encompassing quality control through mapping, variant calling for lineage classification, drug resistance prediction, and phylogenetic inference. However, the pipeline's architecture is not optimal for high-performance computing or cloud computing environments, which often require large datasets. To optimize the pipeline, a Nextflow wrapper MTBseq-nf, was created which offers shorter execution times through the parallel mode along with multiple other thematic improvements. The MTBseq-nf wrapper, as opposed to the linear batched analysis of samples in TBfull step of MTBseq pipeline, can execute multiple instances of the same step in parallel and therefore makes full use of the provided computational resources. For evaluation of scalability and reproducibility, we used 90 *M. tuberculosis* genomes (European Nucelotide Archive - ENA - accession PRJEB7727) for the benchmarking analysis on a dedicated computing server. In our experiments the execution time of MTBseq-nf parallel analysis mode is at least twice as fast as the standard MTBseq pipeline for more than 20 samples. Furthermore, the MTBseq-nf wrapper facilitates reproducibility using the nf-core, bioconda, and biocontainers projects for platform independence. The proposed MTBseq-nf wrapper pipeline is a user-friendly pipeline optimized for hardware efficiency, scalability for larger datasets, and improved reproducibility.

Keywords: Mycobacterium tuberculosis, Tuberculosis genomics, Whole-genome sequencing, Data analysis pipeline, Nexflow, MTBseq, Mycobacterium tuberculosis complex, Pipeline, Workflow

MTBseq-nf: Enabling Scalable Tuberculosis Genomics "Big Data" Analysis through a User-Friendly Nextflow Wrapper for MTBseq pipeline.

INTRODUCTION

Next-generation sequencing (NGS) has revolutionized tuberculosis (TB) research, diagnosis, and surveillance, however analyzing this increasingly voluminous amount of complex bioinformatics data, presents numerous computational challenges (Berger & Yu, 2022). The MTBseq pipeline (Kohl et al., 2018) was published in 2018 to address some of the challenges of data-analysis and improve the reproducibility of whole-genome sequencing (WGS) analysis of Mycobacterium *tuberculosis* complex (MTBC). MTBseq (MTBseq-standard) was one of the first comprehensive end-to-end pipelines and is widely used by researchers working with WGS data to generate and analyze the principal outputs namely (i) strain classification, (ii) phylogenetic tree, (iii) mapping and variant statistics, (iv) Single Nucleotide Polymorphism (SNP) distance matrix and (v) cluster groups. On the other hand, the computational infrastructure available to researchers ranges from the traditional in-house servers or high-performance computing (HPC) platforms. The available options have only been diversified in recent years with the development of (i) batch computing functionality by cloud computing vendors such as Oracle, Amazon Web Services (AWS), Google, Microsoft Azure, IBM and Alibaba etc (ii) open source job orchestrators like Kubernetes, Apache Meson and Hashicorp Nomad to analyze the exponentially increasing volume of data in an efficient manner (Stephens, 2015) . This adoption trend of NGS technologies and use of modern job orchestrators is expected to continue for foreseeable future, especially in the context of precision public health and precision medicine. The MTBseq pipeline, in it current form performs sub-optimally for these modern computing environments, therefore we aimed to enhance the MTBseq pipeline for traditional as well as modern job

orchestrators utilized in large-scale genomic analysis, while optimizing the costs and making the analysis time predictable. We developed MTBseq-nf, a wrapper built with the Nextflow workflow engine (Di Tommaso et al., 2017) - to provide an alternative to current users of MTBseq-standard with various enhancements in its (i) user-friendliness, (ii) maintainability, (iii) reproducibility and (iv) scalability, for large-scale data analysis.

MATERIALS AND METHODS

The design of MTBseq (standard) pipeline

The MTBseq pipeline by Kohl et al. (2018) (MTBseq-standard) relies upon underlying tools such as *GATK3* (McKenna et al., 2010), *PICARD* (*Picard Toolkit*, 2019), *BWA* (Li & Durbin, 2009) and *SAMTOOLS* (Li et al., 2009); and glues these tools together using the *perl5* (Wall et al., 1994) programming language as shown in Figure-1.

FIGURE-1

Its execution model is built upon core abstractions (called **steps**) such as reference mapping (*TBbwa*) and variant calling (*TBvariants*). These steps can be broadly classified into two groups based on (i) the quantity of samples analyzed per step, with sample-specific steps (*TBbwa*, *TBvariants*) and cohort-specific steps (*TBamend*, *TBgroups*) as well as (ii) composition, with primitive single-purpose steps (*TBbwa*, *TBgroups*), and composite multi-purpose step (*TBfull*), combining many of the primitive steps to facilitate routine usage of certain primitive steps, as summarized in Table-1.

TABLE-1

Implementation of MTBseq-nf wrapper

We developed *MTBseq-nf* wrapper, on top of the open-source *MTBseq* pipeline, and implemented numerous enhancements across four broad themes (i) user-friendliness (ii)

scalability (iii) reproducibility and (iv) maintainability, as summarized in Table-2. In the implementation of *MTBseq-nf* wrapper pipeline we built upon two core components. Firstly, the MTBseq-standard pipeline Kohl et al. (2018) exposes a *--step* parameter allowing users to dictate which step of the pipeline they wish to initiate. Figure-1 highlights, how the MTBseq-standard pipeline facilitates the usage of sequential sample-specific steps for routine executions, through a composite *TBfull* step which automates the execution of sample-specific primitive steps from *TBbwa* step until the *TBstrains* step. Upon completion of these steps, the users are expected to provide a TSV file, containing a list of samples and corresponding library names, that should be included in further comparative analysis steps such as *TBjoin*, *TBamed* and *TBgroup* generating the principal outputs of the pipeline as described in Table-2. Secondly, the use of Nextflow workflow manager as per nf-core best-practices pipeline template.

TABLE-2

Nextflow is a workflow management system designed to address the challenges of high-throughput NGS data at scale in a reproducible manner (Di Tommaso et al., 2017), allowing researchers to create complex workflows that integrate multiple bioinformatics tools into a single, cohesive workflow while maintaining portability across different computing infrastructure. The nf-core community further enhances the Nextflow ecosystem by bringing together Nextflow users through a Slack group, hackathons, seminars, training, and other community initiatives that foster collaboration and knowledge sharing (P. A. Ewels et al., 2020; Langer et al., 2024). Moreover, the nf-core pipeline template is constructed with rigorous standards to guarantee robustness, portability, and user-friendliness, due to its integration with other projects from the nf-core ecosystem, such as *nf-core/configs* and *nf-core/modules*.

Internally, the *MTBseq-standard* pipeline depends on the sequential analysis of input sequences thorough the *foreach* looping structures, with each sample advancing to the subsequent step only after all FASTQ samples have completed an individual step (Figure-2). On the other hand, the *MTBseq-nf* wrapper pipeline has two execution modes: (i) default and (ii) parallel mode, activated by the *--parallel* parameter. Both of these modes are functionally equivalent to the combination of *TBfull*, *TBgroups*, *TBamend* and *TBjoin* steps of the MTBseq pipeline,

The principal insights we employed to implement the *--parallel* mode was to combine the advantages of both (i) MTBseq's modular step-wise architecture exposed through the *--step* parameter to facilitate the independent execution of individual steps in the MTBseq pipeline, as well as (ii) the inherent task parallelization provided by the Nextflow workflow engine.

Consequently, when executed with the *--parallel* parameter, the MTBseq-nf wrapper pipeline employs the exact same primitive steps utilized by the *TBfull* step and optimizes the movement of intermediate files through the individual steps (Figure-2); resulting in a significant reduction in the overall execution time of the pipeline, especially when the number of samples in a cohort are large. We depend on the fact that when a sample-specific step is executed with a single sample, the *foreach* loop iterates only once, since only a single sample is accessible to that particular step. Moreover, the MTBseq-nf pipeline (in both modes) automates the generation of a TSV file when all samples undergo the initial phases of the *TBfull* step and initiates the comparative analysis stage, mirroring the behavior of the MTBseq-standard pipeline for subsequent steps, minimizing the manual intervention by users.

FIGURE-2

161 **Validation Infrastructure and Dataset**

162 For the computational studies, we selected a virtual server machine on the Oracle Cloud
163 Infrastructure, featuring 32 CPUs (16 Oracle CPUs), 64 GB of RAM, and a boot disk capacity of
164 2 TB. The essential software requirements for the experiments were installations of (i) Java
165 programming language (Arnold et al., 2005), (ii) Nextflow (Di Tommaso et al., 2017), and (iii)
166 Docker (Merkel, 2014). The MTBseq-standard pipeline v1.1.0 was installed using the bioconda
167 recipe file provided as part of Supplemental data-2. The decision to utilize a server was crucial
168 for our analysis of execution runtime, considering HPC queue systems impose an unpredictable
169 delay before a submitted job starts execution. The Docker platform was chosen for its extensive
170 utilization in cloud batch computing environments (AWS Batch, Azure Batch and Google Batch)
171 as well as contemporary container orchestration solutions like Kubernetes and Hashicorp
172 Nomad. The selection of Docker containers was influenced by the upstream biocontainer project
173 assets (Veiga Leprevost et al., 2017), which are derived from bioconda recipes (Gruning et al.,
174 2018). This approach enabled us to conduct the risk of non-reproducibility stemming from
175 infrastructural discrepancies and enabled us to conduct an isolated analysis of the primary results
176 of the pipeline.

177 We relied upon the original dataset utilized by the original study Kohl et al. (2018) to
178 evaluate the reproducibility and scalability of the MTBseq-nf wrapper in terms of (i) growth of
179 execution time versus the cohort size as well as (ii) the validity of results produced. The dataset,
180 as described by Schleusener et al. (2017) - is publicly available under the European Nucleotide
181 Archive (ENA) accession code **PRJEB7727**. Furthermore, Kohl et al. (2018) - analyzed 91
182 samples for resistance and lineage profiling in a comparative setting as part of Supplementary
183 material of the original publication. These IDs are shared in Supplemental data-1.1 .

FIGURE-3

The data submitted to ENA lists 133 paired-end FASTQ files obtained from *Mycobacterium tuberculosis* cultured samples (Supplemental data-1.2), and upon meticulous examination, we observed that (i) certain samples had several matching `experiment_accession` entries. The Supplementary-1.3 presents a frequency count of `secondary_sample_accession` in relation to `experiment_accession` ERS IDs) (ii) the samples utilized by Kohl et al. (2018) had differences compared to those reported by the ENA project `PRJEB7727`, and (iii) one sample, `ERS457325` lacked any corresponding files in the indicated ENA project. Consequently, to minimize the effect of confounding factors, we opted to deduplicate the samples instead of merging those with identical accession numbers. This was done by utilizing the very first appearance of specific `secondary_sample_accession` IDs from the files acquired from the ENA project and by excluding the missing `ERS457325` (`4730-03`) from our analysis, thereby reducing the final dataset to 90 paired-end FASTQ files. Figure-3 demonstrates our data splitting strategy for the reproducibility as well as the scalability analysis of MTBseq-nf. The conclusive samplesheet is provided in Supplemental data-1.4.

For the scalability analysis, we divided the dataset into six cohorts with an incremental number of samples namely 5, 10, 20, 40, 80 and finally 90 FASTQ paired-end files. Furthermore, the largest dataset, was used for the validation of principal outputs generated by MTBseq and MTBseq-nf (default and parallel) pipelines.

Experimental Setup for Evaluation of Scalability and Reproducibility

To minimize the possibility of regression in MTBseq-nf results (default and parallel modes) in comparison to the MTBseq-standard pipeline. We conducted a triplicated experiments (i) intra-modal analysis to assess reproducibility *within* each mode and (ii) inter-modal analysis

207 to assess reproducibility *across* distinct modes. Consequently, due to this triplicated
208 experimental design, we evaluated the results of nine distinct runs of the pipelines which are
209 highlighted in Table-3.

210 The parameters utilized for these experiments and the associated results are published as
211 part of the Zenodo repository https://doi.org/10.5281/zenodo.14678756, while the names of
212 individual runs and significant parameters are reported in Table-3. Moreover, each experiment
213 was conducted independently without employing Nextflow's resume feature to guarantee fresh
214 execution of each of the individual steps.

215 To evaluate the scalability of the pipeline, we analyzed the execution time growth for the
216 six specified cohorts using the default mode and parallel mode of MTBseq-nf on these datasets.
217 The executions of MTBSeq-nf in both default and parallel modes were monitored and visualized
218 via the Seqera Platform (Di Tommaso et al., 2025), which serves as a centralized repository for
219 tracking execution metrics and maintaining pipeline configurations.

220 To evaluate the reproducibility of the different modes of MTBseq-nf, we compared the
221 principal results of inter- and intra-modal triplicated experiments by conducting (i) 3-way diffs
222 using Araxis Merge (team, 2025), (ii) range analysis of numerical data using R language (Team,
223 2023), along with (iii) phylogenetic tree using IQTREE program (v2.3.4) (Trifinopoulos et al.,
224 2016) as summarized in Table-4.

225 We utilized a 3-way visual diff in order to assess the disparities in the principal results,
226 which are qualitative in nature. For the mapping and variant statistics produced by `TBstats`
227 (including `Mapped Reads` and `SNPs`), we utilized range statistics (min-max analysis) in R to
228 estimate the relative changes across various intra- and inter-modal studies. The scripts utilized

229 for the analysis and visualizations are available in the following repository:
230 https://github.com/abhi18av-phd-projects/mtbseq-nf-publication-analysis.

231 **RESULTS**

232 ***Thematic improvements in MTBseq-nf***

233 The MTBseq-nf wrapper pipeline offers substantial enhancements over the MTBseq-
234 standard pipeline, including an innovative parallel mode that significantly reduces overall
235 execution time for large cohorts. The implemented features span four key themes: (1) user-
236 friendliness, (2) scalability, (3) reproducibility, and (4) maintainability, as highlighted in Table-5
237 and detailed in Supplemental data-3.

238 MTBseq-nf leverages the standardized template from the nf-core community, providing
239 numerous advantages including: (i) configuration files (dotfiles) that address code quality,
240 linting, and testing requirements; (ii) integration with the nf-core/configs project for portability
241 across institutional infrastructures; (iii) access to well-tested modules from the nf-core/modules
242 project; and (iv) a graphical user interface through the nf-schema project for users less familiar
243 with command-line operations.

244 ***Reproducibility analysis of intra-modal comparison***

245 For the intra-modal comparison of the three pipeline-mode combinations, we conducted
246 triplicated experiments with identical infrastructure, dataset, and parameters as described in
247 Table-3.

248 As summarized in Table-6, the classification and SNP distance matrix results
249 demonstrated consistency across experiments, with differences limited to execution dates, which
250 was expected as successive runs could only begin after previous runs completed. Similarly, the
251 phylogenetic trees remained stable across different experiments. The cluster groups, which are

assigned labels based on the SNP matrix, showed some variation in exact labeling across different runs of the TBgroups step, attributable to the inherent nature of the agglomerative algorithm used, though the actual grouping of samples remained accurate.

The statistics report showed some differences in triplicated runs of MTBseq and MTBseq-nf default mode, as shown in Figure-4b and Figure-4c. Notably, for MTBseq-nf parallel mode, none of the numerical fields showed differences across triplicated runs, indicating complete reproducibility of results in this mode.

Furthermore, results obtained from triplicated runs of MTBseq-nf (default) and MTBseq-standard, visualized in Figure-4b and Figure-4d, demonstrated the functional equivalence of these modes.

262 ***Reproducibility analysis of inter-modal comparison***

263 **MTBseq-standard vs MTBseq-nf (default).** The samples with differences are listed in
264 Table-8. Figure-4d highlights the numerical variation in SNP counts between MTBseq-standard
265 and MTBseq-nf (default) runs across the fields of `Uncovered`, `TotalBasesUnambguous`, and
266 `TotalBases`, with negligible differences in the `GCContentUnambiguous` field. Additionally, the
267 UpSet plot in Figure-5a illustrates the variation in specific columns and corresponding sample
268 numbers.

269 **MTBseq-nf (parallel) vs MTBseq-nf (default).** When comparing MTBseq-nf (parallel)
270 with MTBseq-nf (default), we observed variations similar to the intra-modal variation of
271 MTBseq-standard triplicated runs, as shown in Table-9, Figure-5b, and Figure-5c.

272 **MTBseq-nf (parallel) vs MTBseq-standard.** MTBseq-nf (parallel) vs MTBseq-
273 standard: For a comprehensive analysis, we compared MTBseq-nf (parallel) with MTBseq-
274 standard, despite the latter being functionally equivalent to MTBseq-nf (default). This
275 comparison is presented in Table-10, Figure-5e, and Figure-5d.

276 ***Scalability analysis of MTBseq-nf (default) and MTBseq-nf (parallel)***

277 To evaluate execution time growth, we included only one specific run for each mode of
278 MTBseq-nf and excluded MTBseq-standard from the analysis due to the manual intervention
279 step required after TBfull before proceeding to TBstats, TBgroups, and TBamend steps. Since
280 MTBseq-standard and MTBseq-nf (default) are functionally equivalent, comparing total
281 execution time between MTBseq-nf (default) and MTBseq-nf (parallel) was sufficient to analyze
282 the growth curves.

283 As demonstrated in Figure-4a and Table-7, when processing more than 10 FASTQ
284 paired-end files, the execution time of MTBseq-nf (default) is at least double that of MTBseq-nf

285 (parallel), highlighting the significant performance benefits of the parallel mode optimizations.
286 This difference becomes increasingly pronounced as the sample size grows, making MTBseq-nf
287 (parallel) particularly valuable for large-scale genomic analyses.

288 **DISCUSSION**

289 The MTBseq-nf pipeline represents a significant optimization effort to modernize the
290 widely adopted MTBseq-standard pipeline for analyzing WGS data from the *Mycobacterium*
291 *tuberculosis* culture. In doing so, it addresses numerous pain points experienced by users of the
292 original MTBseq pipeline, especially in high-throughput and diverse computing environments.
293 One of the major advancements introduced by MTBseq-nf is its optional `--parallel` execution
294 mode. By decoupling sequential dependencies and capitalizing on the Nextflows inherent
295 parallelization features, MTBseq-nf significantly reduces the overall runtime of the pipeline. Our
296 empirical analysis indicated that the execution time scales much more gracefully in the parallel
297 mode, particularly as the number of samples increases — a critical feature for genomic
298 surveillance programs processing hundreds or thousands of isolates.

299 The task-level isolation introduced by `parallel` mode ensures that each primitive step
300 has sufficient memory (and CPU) during its execution and as soon as the step completes, the
301 relevant docker container exits and the docker container engine ensures that the computational
302 resources of the host system are released and made available for other queued tasks (Di
303 Tommaso et al., 2015). Whereas in the `TBfull` step within MTBseq and MTBseq-nf (default),
304 the consumed memory is not immediately released when the previous process completes as the
305 computations are done sequentially within the same container, therefore the subsequent
306 processes have slightly reduced access to memory, which has an impact upon the memory-
307 intensive computations required by `GATK`.

This improvement with MTBseq-nf *parallel* is especially beneficial for the integration of bioinformatic analysis in routine diagnostics and surveillance work especially in resource-constrained environments such as Low- and Middle-income countries (LMICs), where TB is endemic and timely analysis is vital, and where infrastructure may not be suited to long, sequential compute jobs. Furthermore, the ability to execute the MTBseq-nf pipeline across laptops, servers, HPC clusters, and cloud batch systems allows for unprecedented flexibility and broader adoption.

Reproducibility is a cornerstone of bioinformatics pipelines, particularly when used in clinical and public health contexts. MTBseq-nf leverages Nextflow's declarative parameter feature, integration with container systems (Kadri et al., 2022) such as Docker, and in-built caching mechanisms to rapidly deliver reproducible results across varied environments. The triplicated intra-modal experiments confirmed the consistency of principal outputs across different runs of MTBseq-nf, particularly in the parallel mode, which showed no measurable variation in statistics or results.

Moreover, discrepancies in output observed between sequential and parallel executions, such as minor differences in *TBstats* reports, are likely attributable to the memory-intensive behavior of tools like GATK when executed sequentially on all samples in shared containers. In contrast, the task-level isolation in the parallel mode ensures more consistent memory allocation, leading to increased stability of result across executions.

In architecting MTBseq-nf on top of MTBseq-standard, we have prioritized a straight-forward user experience. By adopting the nf-core best practices, users benefit from structured configuration files, standardized output, and simplified parameter management. Features like an explicit sample sheet, an optional Graphic User Interface (GUI) through the *nf-schema* project,

and remote execution monitoring and infrastructure management (via the Seqera platform) (Di Tommaso et al., 2025) further enhance usability, especially for users less familiar with the command line. Another valuable addition is the inclusion of modules like `FastQC` and `MultiQC` (P. Ewels et al., 2016), which offer quality control summaries that are now seamlessly integrated into the pipeline without manual intervention. In contrast, integrating such tools into MTBseq-standard would require creation, customization, and testing of Perl-wrapped modules.

The adoption of the nf-core pipeline template ensures the future viability of MTBseq-nf by allowing rapid integration of updates and modules. Unlike the traditional software architecture of the MTBseq-standard pipeline, which makes maintenance and extension daunting, especially for newcomers, to the project; the modular design of MTBseq-nf, owing to the nf-core best-practices template, aligns with modern software engineering practices, including continuous integration and delivery (CI/CD), unit testing (e.g., using nf-test), and community contributions via Github. This design ensures that MTBseq-nf can evolve in response to user needs, bug reports, and new tool integrations — such as IQTREE, which, although not currently included, can be added as an integrated module in future iterations depending on research requirements.

In terms of limitations, despite its many strengths, MTBseq-nf inherits the core computational logic of MTBseq-standard, limiting opportunities for deeper architectural optimization. The reliance on MTBseq's Perl-based modules, while preserving backward compatibility, poses a ceiling on how extensively the workflow can be modernized. Future work could involve translating these steps into native Nextflow processes or subworkflows, which would improve maintainability and support pluggable alternative tools for the individual steps of MTBseq-standard for alignment and variant calling tools.

354 Additionally, minor variations in SNP clustering labels across runs, an artifact of the
355 agglomerative clustering algorithm, do not impact downstream phylogenetic analyses but
356 underscores the importance of clear documentation and careful interpretation of clustering
357 outputs. The efficiency gains offered by MTBseq-nf reduce execution time, cost of analysis and
358 contribute to sustainability goals by lowering computational carbon footprints, an increasingly
359 relevant consideration in bioinformatics research (Grealey et al., 2022; Lannelongue et al.,
360 2023). Quantifying and comparing emissions across pipelines could be a compelling area for
361 future investigation. Furthermore, the versatility of MTBseq-nf enables its deployment in both
362 high-resource and constrained settings, supporting TB control efforts globally. By simplifying
363 complex analyses and promoting reproducibility, the pipeline aligns with the broader goals of
364 precision public health and genomic epidemiology.

365 **CONCLUSION**

366 In light of the (i) rapidly increasing WGS data for *M. tuberculosis* (ii) along with its
367 applicability in genomic surveillance and precision medicine; the MTBseq-nf offers a more
368 scalable, portable and user-friendly alternative to the MTBseq-pipeline. The MTBseq-nf wrapper
369 optimises the MTBseq-standard pipeline particularly towards an effecient usage of
370 computational resources for genomic "Big Data" analysis for. In addition, it also improves the
371 user-friendliness, reproducibility, scalability and customisablity as compared to the MTBseq-
372 standard pipeline. These enhacements in MTBseq-nf adds to the versatality of that the original
373 MTBseq-standard as a suitable alternative, not only for routine usage in resource-rich
374 environments, but particularly in resource-constraint LMIC settings where fine-grained control
375 over resource usage is necessary.

CONFLICT OF INTEREST STATEMENT

The authors have no competing interests to disclose, with the exception of Abhinav Sharma, who was employed by Seqera Labs until 2022 and has also been involved in the development of Nextflow, Seqera Platform and the MAGMA pipeline.

ACKNOWLEDGEMENTS

We extend our gratitude towards the authors of MTBseq-standard pipeline, Kohl et al. (2018), for the initiative of publishing MTBSeq pipeline with open source on Github.

DATA AVAILABILITY STATEMENT

The dataset published on Zenodo https://doi.org/10.5281/zenodo.14678756 contains results and metadata for the nine runs of MTBseq-standard, MTBseq-nf (default) and MTBseq-nf (parallel) including (i) MultiQC reports (ii) runtime metrics (iii) pipeline parameters (iv) results of all experiments.

CODE AVAILABILITY STATEMENT

MTBseq-nf pipeline https://zenodo.org/records/15234640

MTBseq-nf analysis scripts https://github.com/abhi18av-phd-projects/mtbseq-nf-publication-analysis

FUNDING STATEMENT

This study was also funded by the Brazilian National Council for Scientific and Technological Development (Conselho Nacional de Desenvolvimento Científico e Tecnológico - CNPq) grant process number 445784/2023-7 under the REVIGET N/NE project. This work was supported in part by Oracle Cloud credits (Award Number 3083687) and related resources provided by Oracle for Research. The financial assistance of the National Research Foundation

(NRF) towards this research is hereby acknowledged by AS. Opinions expressed and conclusions arrived at, are those of the author and are not necessarily to be attributed to the NRF.

GENERATIVE AI STATEMENT

We utilized the ChatGPT and QuillBot tool for grammar checks, paraphrasing and summarization.

REFERENCES

Arnold, K., Gosling, J., & Holmes, D. (2005). *The java programming language*. Addison Wesley Professional.

Berger, B., & Yu, Y. W. (2022). Navigating bottlenecks and trade-offs in genomic data analysis [Journal Article]. *Nature Reviews Genetics*, 1–16. https://doi.org/10.1038/s41576-022-00551-z

Di Tommaso, P., Chatzou, M., Floden, E. W., Barja, P. P., Palumbo, E., & Notredame, C. (2017). Nextflow enables reproducible computational workflows [Journal Article]. *Nature Biotechnology*, *35*(4), 316–319. https://doi.org/10.1038/nbt.3820

Di Tommaso, P., Floden, E. W., & team, S. L. (2025). *Seqera cloud platform*. https://seqera.io/

Di Tommaso, P., Palumbo, E., Chatzou, M., Prieto, P., Heuer, M. L., & Notredame, C. (2015). The impact of docker containers on the performance of genomic pipelines [Journal Article]. *Peerj*, *3*. https://doi.org/ARTN e1273 10.7717/peerj.1273

Ewels, P. A., Peltzer, A., Fillinger, S., Patel, H., Alneberg, J., Wilm, A., Garcia, M. U., Di Tommaso, P., & Nahnsen, S. (2020). The nf-core framework for community-curated bioinformatics pipelines [Journal Article]. *Nature Biotechnology*, *38*(3), 276–278. https://doi.org/10.1038/s41587-020-0439-x

Ewels, P., Magnusson, M., Lundin, S., & Käller, M. (2016). MultiQC: Summarize analysis results for multiple tools and samples in a single report [Journal Article]. *Bioinformatics*, *32*(19), 3047–3048. https://doi.org/10.1093/bioinformatics/btw354

Grealey, J., Lannelongue, L., Saw, W. Y., Marten, J., Meric, G., Ruiz-Carmona, S., & Inouye, M. (2022). The carbon footprint of bioinformatics [Journal Article]. *Mol Biol Evol*, *39*(3). https://doi.org/10.1093/molbev/msac034

Gruning, B., Dale, R., Sjodin, A., Chapman, B. A., Rowe, J., Tomkins-Tinch, C. H., Valieris, R., Koster, J., & Bioconda, T. (2018). Bioconda: Sustainable and comprehensive software distribution for the life sciences [Journal Article]. *Nat Methods*, *15*(7), 475–476. https://doi.org/10.1038/s41592-018-0046-7

Kadri, S., Sboner, A., Sigaras, A., & Roy, S. (2022). Containers in bioinformatics , , [Journal Article]. *Journal of Molecular Diagnostics*, *24*(5), 442–454. https://doi.org/10.1016/j.jmoldx.2022.01.006

Kohl, T. A., Utpatel, C., Schleusener, V., De Filippo, M. R., Beckert, P., Cirillo, D. M., & Niemann, S. (2018). MTBseq: A comprehensive pipeline for whole genome sequence analysis of mycobacterium tuberculosis complex isolates [Journal Article]. *PeerJ*, *6*, e5895. https://doi.org/10.7717/peerj.5895

Langer, B. E., Amaral, A., Baudement, M.-O., Bonath, F., Charles, M., Chitneedi, P. K., Clark, E. L., Di Tommaso, P., Djebali, S., Ewels, P. A., Eynard, S., Yates, J. A. F., Fischer, D., Floden, E. W., Foissac, S., Gabernet, G., Garcia, M. U., Gillard, G., Gundappa, M. K., … community., the nf-core. (2024). Empowering bioinformatics communities with nextflow and nf-core. *bioRxiv*. https://doi.org/10.1101/2024.05.10.592912

Lannelongue, L., Aronson, H. G., Bateman, A., Birney, E., Caplan, T., Juckes, M., McEntyre, J., Morris, A. D., Reilly, G., & Inouye, M. (2023). GREENER principles for environmentally sustainable computational science [Journal Article]. *Nat Comput Sci*, *3*(6), 514–521. https://doi.org/10.1038/s43588-023-00461-y

Li, H., & Durbin, R. (2009). Fast and accurate short read alignment with burrows-wheeler transform [Journal Article]. *Bioinformatics*, *25*(14), 1754–1760. https://doi.org/10.1093/bioinformatics/btp324

Li, H., Handsaker, B., Wysoker, A., Fennell, T., Ruan, J., Homer, N., Marth, G., Abecasis, G., Durbin, R., & Proc, 1000. G. P. D. (2009). The sequence alignment/map format and SAMtools [Journal Article]. *Bioinformatics*, *25*(16), 2078–2079. https://doi.org/10.1093/bioinformatics/btp352

McKenna, A., Hanna, M., Banks, E., Sivachenko, A., Cibulskis, K., Kernytsky, A., Garimella, K., Altshuler, D., Gabriel, S., Daly, M., & DePristo, M. A. (2010). The genome analysis toolkit: A MapReduce framework for analyzing next-generation DNA sequencing data [Journal Article]. *Genome Research*, *20*(9), 1297–1303. https://doi.org/10.1101/gr.107524.110

Merkel, D. (2014). Docker: Lightweight linux containers for consistent development and deployment. *Linux Journal*, *2014*(239), 2.

Picard toolkit. (2019). https://broadinstitute.github.io/picard/

Schleusener, V., Koser, C. U., Beckert, P., Niemann, S., & Feuerriegel, S. (2017). Mycobacterium tuberculosis resistance prediction and lineage classification from genome sequencing: Comparison of automated analysis tools [Journal Article]. *Sci Rep*, *7*, 46327. https://doi.org/10.1038/srep46327

Stephens, S. Y. A. F., Zachary D. AND Lee. (2015). Big data: Astronomical or genomical? *PLOS Biology*, *13*(7), 1–11. https://doi.org/10.1371/journal.pbio.1002195

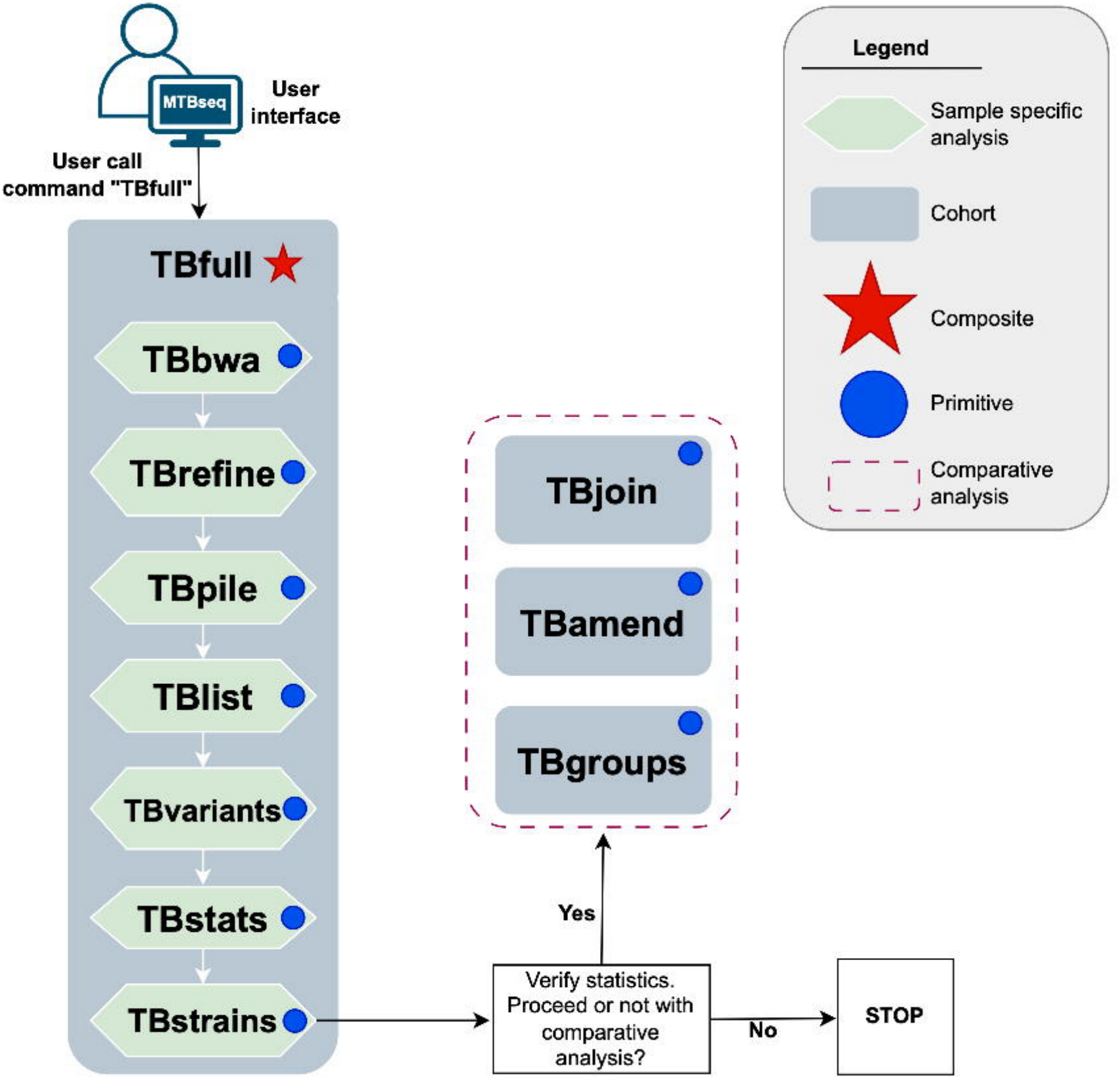
team, A. L. (2025). *Seqera cloud platform*. https://www.araxis.com/merge/index.en

Team, R. C. (2023). *R: A language and environment for statistical computing*. R Foundation for Statistical Computing. https://www.R-project.org/

Trifinopoulos, J., Nguyen, L.-T., Haeseler, A. von, & Minh, B. Q. (2016). W-IQ-TREE: A fast online phylogenetic tool for maximum likelihood analysis. *Nucleic Acids Research*, *44*(W1), W232–W235. https://doi.org/10.1093/nar/gkw256

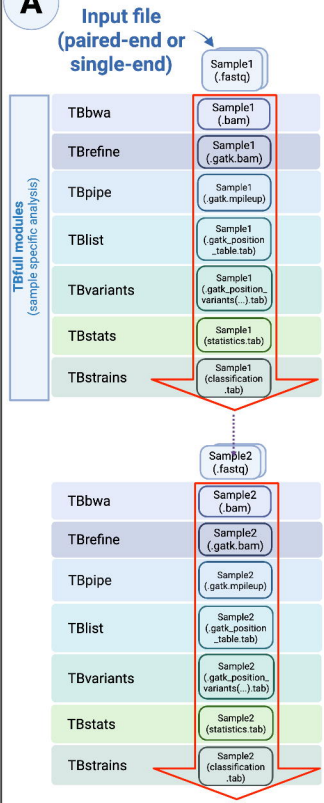
Veiga Leprevost, F. da, Gruning, B. A., Alves Aflitos, S., Rost, H. L., Uszkoreit, J., Barsnes, H., Vaudel, M., Moreno, P., Gatto, L., Weber, J., Bai, M., Jimenez, R. C., Sachsenberg, T., Pfeuffer, J., Vera Alvarez, R., Griss, J., Nesvizhskii, A. I., & Perez-Riverol, Y. (2017). BioContainers: An open-source and community-driven framework for software standardization [Journal Article]. *Bioinformatics*, *33*(16), 2580–2582. https://doi.org/10.1093/bioinformatics/btx192

Wall, L. et al. (1994). *The perl programming language*. Prentice Hall Software Series.

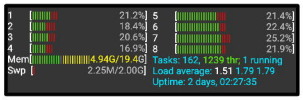

MTBseq
User interface
User call command "TBfull"
TBfull
TBbwa
TBrefine
TBpile
TBlist
TBvariants
TBstats
TBstrains
TBjoin
TBamend
TBgroups
Yes
Verify statistics. Proceed or not with comparative analysis?
No
STOP
Legend
Sample specific analysis
Cohort
Composite
Primitive
Comparative analysis

MTBseq-standard and MTBseq-nf (default)

A Standard execution model using the command "--step TBfull"



Sub-optimal Resource Usage

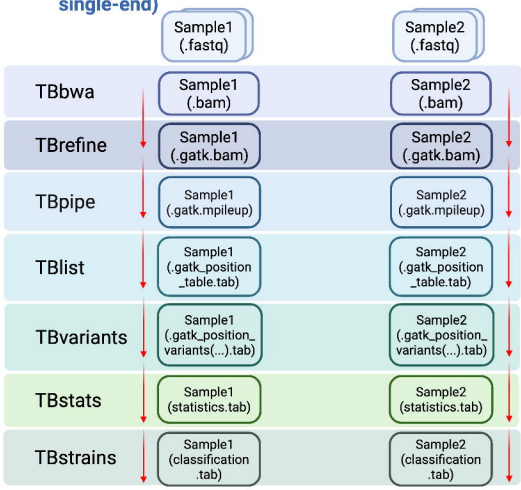


Sequential Analysis of Samples

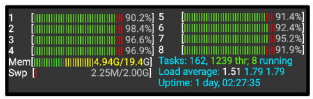
All TBfull module tasks are executed in a linear manner from TBbwa until TBstrains. The underlying implementation relies upon "foreach" loop.

MTBseq-nf (parallel)

B Parallel mode utilizes the granular "--step" commands

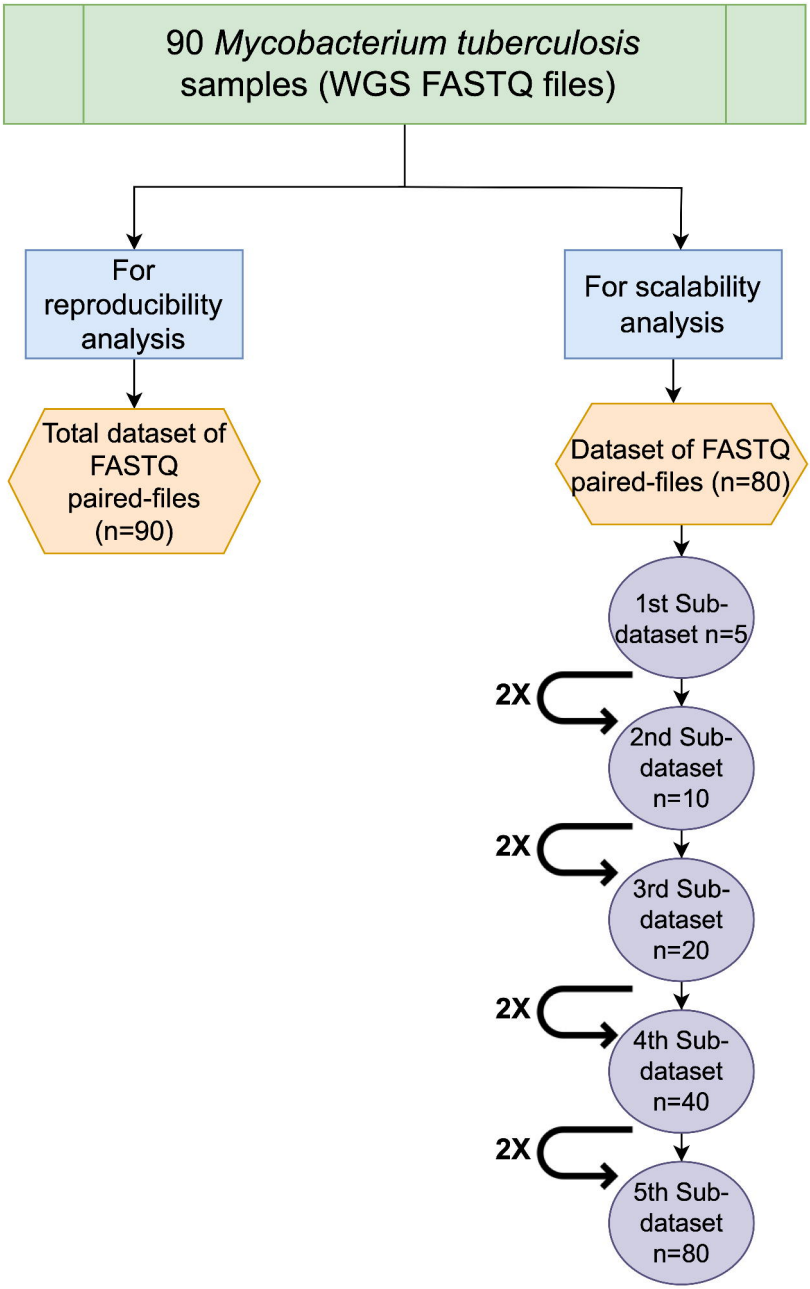


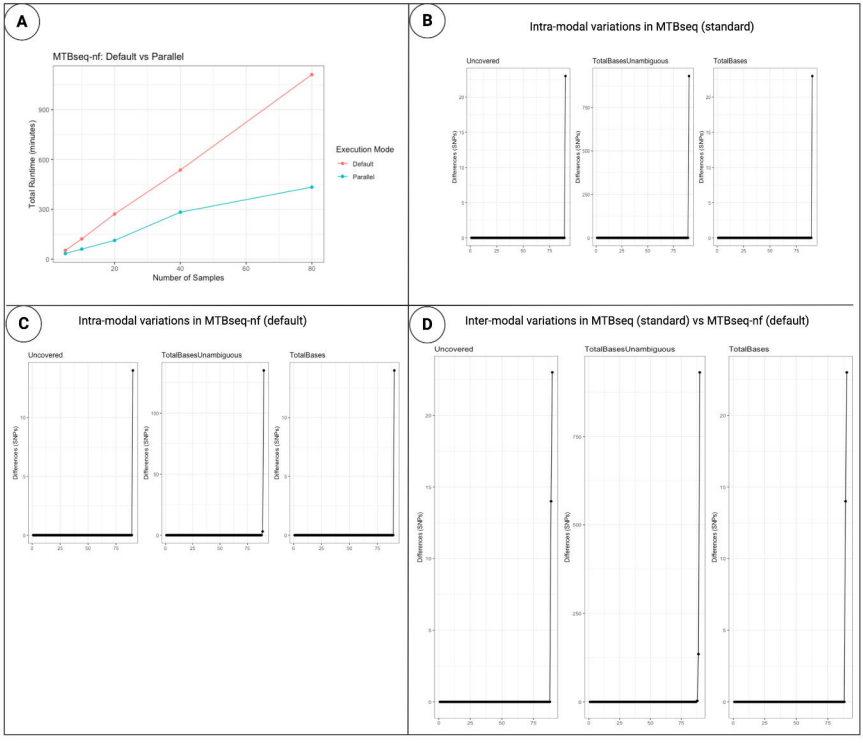
Efficient Resource Usage

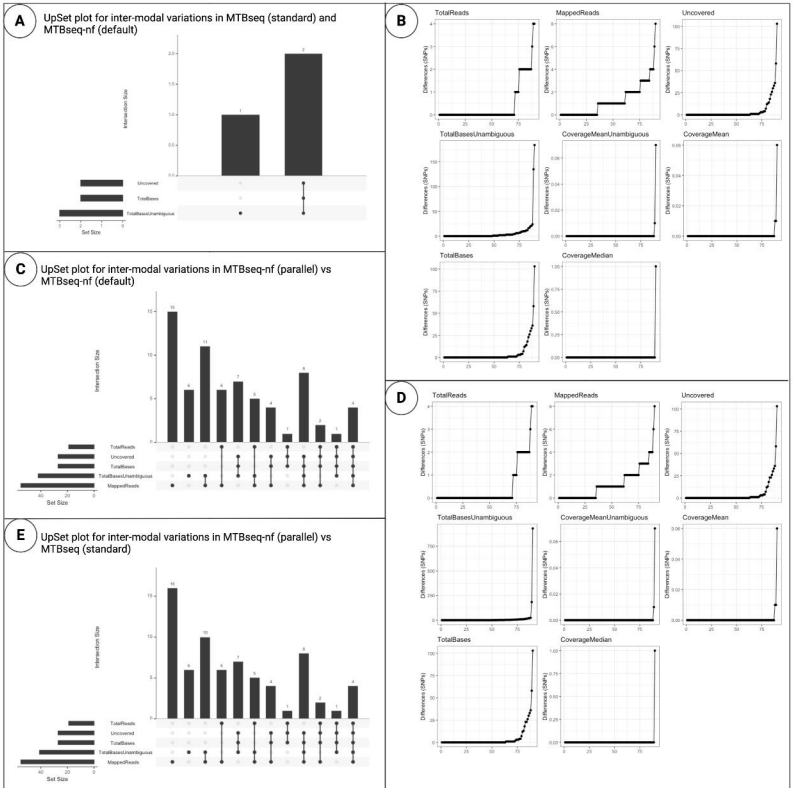


Parallelized and Independent Analysis of Samples

- All tasks are executed in an independent manner as soon as a sample is ready for a particular step. The implementation does not rely upon "foreach".
- All samples are processed in a parallel manner, across individual steps and proceed from one step to another without dependencies.


90 Mycobacterium tuberculosis samples (WGS FASTQ files)
For reproducibility analysis
For scalability analysis
Total dataset of FASTQ paired-files (n=90)
Dataset of FASTQ paired-files (n=80)
1st Sub-dataset n=5
2X
2nd Sub-dataset n=10
2X
3rd Sub-dataset n=20
2X
4th Sub-dataset n=40
2X
5th Sub-dataset n=80


A
MTBseq-nf: Default vs Parallel
Total Runtime (minutes)
Number of Samples
Execution Mode
Default
Parallel
B
Intra-modal variations in MTBseq (standard)
Uncovered
TotalBasesUnambiguous
TotalBases
Differences (SNPs)
C
Intra-modal variations in MTBseq-nf (default)
Uncovered
TotalBasesUnambiguous
TotalBases
Differences (SNPs)
D
Inter-modal variations in MTBseq (standard) vs MTBseq-nf (default)
Uncovered
TotalBasesUnambiguous
TotalBases
Differences (SNPs)


A
UpSet plot for inter-modal variations in MTBseq (standard) and MTBseq-nf (default)
Intersection Size
Set Size
Uncovered
TotalBases
TotalBasesUnambiguous
B
TotalReads
MappedReads
Uncovered
TotalBasesUnambiguous
CoverageMeanUnambiguous
CoverageMean
TotalBases
CoverageMedian
Differences (SNPs)
C
UpSet plot for inter-modal variations in MTBseq-nf (parallel) vs MTBseq-nf (default)
Intersection Size
Set Size
TotalReads
Uncovered
TotalBases
TotalBasesUnambiguous
MappedReads
D
TotalReads
MappedReads
Uncovered
TotalBasesUnambiguous
CoverageMeanUnambiguous
CoverageMean
TotalBases
CoverageMedian
Differences (SNPs)
E
UpSet plot for inter-modal variations in MTBseq-nf (parallel) vs MTBseq (standard)
Intersection Size
Set Size
TotalReads
Uncovered
TotalBases
TotalBasesUnambiguous
MappedReads