SvedbergOpen
DISSEMINATION OF KNOWLEDGE

International Journal of Artificial Intelligence and Machine Learning

Publisher's Home Page: https://www.svedbergopen.com/

Research Paper Open Access

Run Plan Validator: A Platform-Agnostic, lane-Aware Pre-Run Barcode Validation Framework for Preventing Deterministic Index Collisions in High-Throughput Sequencing

Ankur Chaturvedi[1], Bhawna Rathi[2], Nagaraja M. Phani[3*]

[1]Pt Bioinformatics Phd Student / Sr Manager; Amity Institute of Biotechnology / Molecular Genetics; Bioinformatics; Amity University J-3 Block (AIB), Ground Floor, Amity University Noida, Sector-125, Noida - 201303, Uttar Pradesh, India / Lifecell International Pvt Ltd. No. 26, Vandalur-Kelambakkam Main Road, Keelakottaiyur, Chennai, Tamil Nadu, 600127. E-mail Id: ankur.chaturvedi@s.amity.edu; ORCID: 0009-0006-7129-2954
[2]Assistant Professor; Amity Institute of Biotechnology; Bioinformatics; Amity University J-3 Block (AIB), Ground Floor, Amity University Noida, Sector-125, Noida - 201303, Uttar Pradesh, India. E-mail Id: brathi@amity.edu; Orcid Id: 0000-0002-4341-6108
[3*]Head - Molecular Genetics; Molecular Genetics; Molecular Genetics; Lifecell International Pvt Ltd. No. 26, Vandalur-Kelambakkam Main Road, Keelakottaiyur, Chennai, Tamil Nadu, 600127, E-mail: phani.n@lifecell.in; ORCID: 0000-0002-0262-092X

Abstract
Barcode misassignment remains a significant source of sample cross-contamination and data loss in multiplexed high-throughput sequencing. While advances in dual indexing and platform chemistry have reduced stochastic index hopping, deterministic index collisions—where two or more samples share identical barcode sequences within the same demultiplexing scope—remain entirely preventable yet continue to occur in operational sequencing pipelines. Existing tools detect conflicts post-sequencing, at which point remediation is impossible. We present RunPlanValidator, an open-source, platform-agnostic pre-run barcode validation framework designed to prevent deterministic index collisions prior to sequencing. The framework supports heterogeneous run designs, including mixed single- and dual-index libraries, variable index lengths (6–10 bp), multi-sheet run plans, and optional lane-aware demultiplexing consistent with Illumina NovaSeq and MGI platforms. In contrast to mismatch-based barcode design filters that over-restrict operational workflows, RunPlanValidator flags only index configurations guaranteed to cause demultiplexing ambiguity. Evaluation on real-world run plans (2–700 samples per run) demonstrated elimination of deterministic index clashes without rejecting valid operational reuse patterns. Deterministic pre-run barcode validation offers a low-cost, high-impact intervention to improve sequencing reliability, reduce preventable run failures, and enhance reproducibility across platforms. RunPlanValidator is deployable via command-line and graphical interfaces, enabling adoption by both bioinformaticians and laboratory personnel.

Keywords: Barcode Collision, Bioinformatics, Demultiplexing, High-throughput Sequencing, Index Misassignment, Multiplex Sequencing, Run-plan Validation, Sequencing Quality Control.

1. Introduction

High-throughput sequencing (HTS) technologies have transformed modern genomics research, clinical diagnostics, and commercial sequencing by enabling large numbers of biological samples to be processed in parallel. A key component of this scalability is sample multiplexing, in which individual libraries are assigned sample-specific index sequences and subsequently pooled for sequencing. During demultiplexing, these index sequences are used to identify and separate reads originating from the different libraries in the sequencing pool [1], [2].

The increasing scale of multiplexed sequencing has consequently made reliable index assignment an important component of sequencing quality control. Incorrect assignment of sequencing reads can lead to sample cross-talk, contamination of downstream datasets, and loss of confidence in the resulting biological measurements. Index misassignment may arise through several mechanisms, including sequencing errors, incorrect run configuration, and molecular phenomena such as index hopping or index switching. Index hopping has been extensively documented in multiplexed sequencing workflows and can result in sequencing reads being associated with an incorrect index during demultiplexing [2]–[4].

The adoption of dual-indexing strategies has provided an important approach for improving sample identification in multiplexed sequencing. In a single-index configuration, a sample is identified using one index sequence, whereas dual-index sequencing uses two index sequences, commonly referred to as Index 1 (i7) and Index 2 (i5). Double indexing has been demonstrated to improve the accuracy of multiplex sequencing, while non-redundant or unique dual-indexing strategies can substantially reduce the impact of index-swapping artifacts in sequencing workflows [2], [5], [6].

However, dual indexing does not eliminate all problems associated with run planning. In particular, a

distinction must be made between stochastic index misassignment and deterministic index collision. Stochastic index misassignment may occur despite a correctly designed run because of molecular or sequencing processes such as index hopping. A deterministic collision, in contrast, is introduced directly during run planning when two or more samples are assigned indistinguishable index configurations within the same demultiplexing scope. Such a collision is preventable before sequencing begins and should therefore be considered an important target for upstream quality control [2]–[4].

A deterministic collision can occur, for example, when two single-indexed samples are assigned the same Index 1 sequence within the same demultiplexing scope. In a dual-index configuration, the relevant identifier is the combination of Index 1 and Index 2. Thus, two samples may legitimately share an individual i7 sequence when their i5 sequences differ, whereas assignment of the same complete i7–i5 combination to multiple samples within the same demultiplexing scope creates an intrinsically ambiguous configuration. This distinction is particularly important in operational sequencing environments where complete uniqueness of every individual index component may not always be practical or necessary.

The problem becomes more complex in laboratories that operate heterogeneous sequencing workflows. Research and commercial sequencing facilities may process libraries generated using different preparation protocols, index lengths, sequencing platforms, and historical barcode inventories. The RunPlanValidator development and evaluation environment included heterogeneous run plans containing 6-bp, 8-bp, and 10-bp indices, mixed single- and dual-index libraries, and both lane-aware and lane-agnostic configurations. Run sizes ranged from small pilot runs containing two samples to production-oriented run plans containing as many as 700 samples.

In such environments, run plans are commonly maintained in spreadsheet-based formats containing sample identifiers, index sequences, run identifiers, and, where applicable, lane assignments. The use of spreadsheets provides flexibility for laboratory operations but also creates opportunities for manual duplication, inconsistent formatting, incomplete fields, and inadvertent reuse of barcode configurations. RunPlanValidator was therefore designed around Excel-formatted sequencing run plans, including workbooks containing multiple worksheets, rather than requiring users to convert operational run plans into a specialized database or programming-specific format.

Another important consideration is that the scope within which index uniqueness must be evaluated may depend on the sequencing configuration. In lane-aware workflows, samples assigned to different lanes may be processed within distinct demultiplexing scopes. In contrast, lane-agnostic workflows may require index uniqueness across the complete run. The RunPlanValidator framework therefore supports an optional Lane field. When lane information is available, samples are grouped by lane for validation; when it is absent, validation is performed globally.

This distinction is particularly relevant in heterogeneous sequencing environments involving platforms with different operational models. The run plans evaluated during development included configurations corresponding to Illumina workflows in which lane information may be relevant, as well as lane-agnostic configurations such as the MGI T7 workflow. Rather than imposing a rigid platform-specific validation mode, RunPlanValidator uses the information contained in the run plan to establish the validation scope. Lane information is used when supplied, whereas global validation is applied when lane information is absent.

Existing approaches to barcode management address related but distinct problems. Barcode-design methods frequently use sequence-distance criteria, including Hamming distance, to ensure that barcode sequences remain sufficiently separated to tolerate sequencing errors. Such approaches are valuable when designing new barcode sets and when a laboratory wishes to impose a conservative minimum distance between index sequences [7]. However, the objective of validating an already constructed operational run plan is narrower: to determine whether the planned index assignments contain guaranteed deterministic collisions within the relevant demultiplexing scope.

Applying conservative mismatch-based filtering to an operational run plan can therefore produce an undesirable outcome. Two index sequences may differ by only a small number of bases while still representing distinct barcode assignments that can be handled by the intended demultiplexing workflow. Automatically rejecting such combinations can unnecessarily reduce the number of usable samples in a sequencing run. The RunPlanValidator evaluation specifically compared its deterministic approach with mismatch-based validation and found that conservative Hamming-distance filters rejected between 30% and 40% of operationally valid run plans, whereas RunPlanValidator produced zero false-positive rejections under the evaluated deterministic collision criteria.

This observation motivated the central design principle of RunPlanValidator: the validator should block configurations that are deterministically ambiguous while permitting configurations that remain distinguishable under the intended demultiplexing model. Accordingly, the framework does not use sequence-distance thresholds as its primary collision criterion. Instead, it evaluates exact identity of the relevant index configuration. Single-index libraries are evaluated using Index 1 identity, whereas dual-index libraries are evaluated using the complete Index 1–Index 2 pair.

The distinction between individual index reuse and complete barcode reuse is operationally important.

Consider two dual-indexed libraries that use the same i7 sequence but different i5 sequences. The i7 sequence is shared, but the complete index combinations remain distinguishable. Treating the shared i7 as an automatic collision would unnecessarily restrict the available barcode space. Conversely, when both i7 and i5 are identical for two samples within the same validation scope, the samples have no unique index configuration with which to distinguish their sequencing reads. RunPlanValidator therefore permits the former configuration while blocking the latter.

The framework also addresses mixed single- and dual-index run plans. Operational sequencing laboratories may encounter situations in which single-index and dual-index libraries are present within the same run or worksheet. A validator that assumes a single indexing strategy may incorrectly classify such configurations or apply inappropriate collision rules. RunPlanValidator instead determines the indexing mode directly from the run-plan contents. Each sample is classified as single-indexed or dual-indexed according to whether a non-empty secondary index is present.

Reliable handling of spreadsheet data is therefore an important component of the validation process. Empty cells, null values, and whitespace-only entries can occur in operational run plans, particularly when Index 2 is optional. RunPlanValidator explicitly normalizes these values before determining the indexing configuration, reducing the risk that an incomplete or incorrectly formatted spreadsheet entry will result in inappropriate classification.

A further practical limitation of post-sequencing detection is that once a sequencing run has been completed, the underlying sequencing resources have already been consumed. If two samples were assigned an identical index configuration, subsequent analysis may not be able to recover the original sample identity from the index information alone. Consequently, deterministic collision detection is most valuable when performed before sequencing, when the run plan can still be modified without requiring a new sequencing experiment.

RunPlanValidator was developed to provide such a pre-run safeguard. The framework was implemented in Python with an emphasis on transparency, portability, and integration into existing laboratory workflows. It accepts Excel-formatted run plans, supports multiple worksheets, normalizes index information, determines the appropriate indexing strategy, establishes the validation scope, and evaluates index configurations for deterministic collisions.

The framework was intentionally designed to provide both computational and laboratory-facing interfaces. A command-line implementation enables integration into automated sequencing pipelines and quality-control procedures, while an optional graphical interface allows laboratory personnel to validate run plans without requiring programming expertise. Validation outputs include machine-readable CSV files containing detected collisions, human-readable PDF reports for quality-assurance and audit documentation, and exit codes suitable for automated workflow integration.

The resulting workflow occupies a practical position between barcode-design software and post-sequencing demultiplexing analysis. Rather than attempting to model every possible source of sequencing misassignment, RunPlanValidator addresses a specific and preventable class of errors: deterministic index collisions introduced during run planning. Stochastic phenomena such as index hopping remain outside the scope of the current implementation. This separation allows the framework to maintain a simple, transparent validation rule while avoiding claims that deterministic validation alone can eliminate all forms of sample misassignment.

The principal objective of this work was therefore to develop and evaluate an accessible computational framework for deterministic pre-run validation of heterogeneous sequencing run plans. The evaluation was designed to test the framework across different run sizes, index lengths, indexing strategies, and demultiplexing scopes. In particular, the study examined whether RunPlanValidator could identify deterministic collisions consistently while avoiding unnecessary rejection of operationally valid partial index reuse.

The principal contributions of this work are as follows:

- Development of a deterministic pre-run barcode validation framework for identifying guaranteed index collisions before sequencing.
- Support for heterogeneous sequencing run plans, including mixed single- and dual-index libraries, variable index lengths, multiple worksheets, and optional lane information.
- Scope-aware collision detection that distinguishes lane-aware from lane-agnostic validation according to the information contained in the run plan.
- Exact index-identity validation that blocks genuine deterministic collisions while permitting operationally valid reuse of individual index components.
- Automated processing of Excel-based run plans, including normalization of incomplete or empty index fields and data-driven classification of single- and dual-index libraries.
- Generation of structured and human-readable validation outputs, including CSV collision reports, PDF quality-control reports, and machine-readable exit codes.
- Provision of a graphical interface that extends pre-run barcode validation to laboratory personnel without requiring extensive command-line expertise.

Collectively, RunPlanValidator provides an upstream quality-control framework for identifying preventable

barcode conflicts before sequencing resources are committed. By focusing on deterministic barcode identity within the relevant demultiplexing scope, the framework seeks to balance sequencing reliability with the practical constraints of heterogeneous laboratory operations. The evaluation presented in this work demonstrates the ability of the framework to identify deterministic index collisions across run plans ranging from small pilot experiments to large production-oriented configurations while avoiding the false-positive rejection associated with conservative mismatch-based filtering.

2. Materials And Methods

A. Run-Plan Input Structure

RunPlanValidator was implemented as a pre-run quality-control framework for validating sequencing run plans before commencement of sequencing. The framework accepts run plans in Excel spreadsheet format, reflecting the tabular structure commonly used for sample tracking, index assignment, and sequencing-run configuration. Multiplexed sequencing relies on sample-specific index sequences for subsequent demultiplexing and assignment of sequencing reads to individual libraries [8], [9].

The minimum input structure consists of three required fields: RunID, SampleID, and Index1 (i7). Two additional fields, Lane and Index2 (i5), are optional. The Lane field enables validation according to lane-specific demultiplexing scope, whereas Index2 is used to represent dual-indexed libraries. This structure permits the same framework to process both single-index and dual-index run plans without requiring separate input formats.

The input design was intentionally kept close to the operational run-plan format rather than requiring users to transform their spreadsheets into a specialized database structure. This approach allows RunPlanValidator to be applied directly to run plans generated during routine sequencing operations and reduces the possibility of errors introduced during format conversion.

The framework was evaluated using run plans containing different sequencing configurations and index lengths. The evaluated datasets included 6-bp, 8-bp, and 10-bp indices, mixed single- and dual-index libraries, and both lane-aware and lane-agnostic run configurations. Run sizes ranged from small pilot configurations containing two samples to large production-oriented run plans containing up to 700 samples.

B. Data Normalization and Index Classification

Because sequencing run plans are frequently maintained as spreadsheets, incomplete or inconsistent cell contents can occur during manual preparation or modification. In particular, optional Index2 fields may contain null values, empty strings, or whitespace-only entries. RunPlanValidator therefore performs explicit normalization of input values before applying the collision-detection rules. This prevents missing or formatting-related values from being incorrectly interpreted as valid secondary index sequences.

Following normalization, each sample is classified automatically as either single-indexed or dual-indexed. The classification is determined solely by the presence of a non-empty Index2 value. A sample containing a valid Index1 value but no non-empty Index2 value is treated as single-indexed, whereas the presence of Index2 results in classification as dual-indexed.

This data-driven classification avoids dependence on a separate user-provided field specifying the indexing strategy. Such an approach is advantageous for operational run plans because indexing annotations may otherwise be incomplete, inconsistent, or manually entered.

The normalization process establishes a consistent representation of index sequences before comparison. Exact index comparison is important because the objective of the present framework is to identify identical barcode configurations rather than to estimate sequence similarity or error tolerance.

C. Determination of Demultiplexing Scope

A central component of RunPlanValidator is the determination of the scope within which index uniqueness must be evaluated. Index reuse does not necessarily constitute a collision across an entire sequencing run if the corresponding samples are processed in separate demultiplexing scopes. The framework therefore supports both lane-aware and lane-agnostic validation.

When the optional Lane field is present, samples are grouped according to their lane identifiers. Validation is then performed independently within each lane. This configuration is intended to represent workflows in which lane assignment forms part of the relevant demultiplexing context. Lane-specific organization is particularly relevant to sequencing systems and run configurations in which demultiplexing is performed independently across lanes [8], [10].

When Lane information is absent, the framework performs validation at the global level. All samples belonging to the relevant worksheet/run are consequently evaluated within a common validation scope. This allows the same software to accommodate run configurations in which lane information does not define separate barcode namespaces.

The use of the run plan itself to establish the validation scope avoids the need for extensive platform-specific configuration. Instead of requiring users to select a particular platform mode, RunPlanValidator uses the information supplied in the run plan to determine whether lane-specific grouping is possible.

For each worksheet, the framework therefore establishes one or more validation groups according to the

following principle:

- Lane field present: samples are evaluated within their respective lane groups.
- Lane field absent: samples are evaluated globally.

This scope determination is performed before collision detection so that barcode comparisons are restricted to samples that share the same operational demultiplexing scope.

D. Deterministic Collision Detection

RunPlanValidator defines a barcode collision as a condition in which two or more samples possess an index configuration that is identical within the same validation scope. The framework deliberately distinguishes deterministic identity from sequence similarity. Consequently, the primary collision algorithm does not apply a minimum Hamming-distance threshold or other mismatch-based heuristic.

Within each validation group, sample index configurations are evaluated to identify duplicate assignments. For single-index libraries, the relevant identifier is the Index1 (i7) sequence. Two single-indexed samples sharing an identical Index1 sequence within the same validation scope are therefore classified as a deterministic collision.

For dual-indexed libraries, the relevant identifier is the complete Index1–Index2 pair. Two samples are classified as colliding only when both their Index1 and Index2 sequences are identical within the same validation scope. The use of paired index combinations is consistent with the principles of dual-index sequencing, in which two independent index reads contribute to sample identification [9], [11].

This distinction is important because reuse of one index component does not necessarily result in ambiguous sample assignment. For example, two dual-indexed samples may share the same Index1 sequence while possessing different Index2 sequences. In such a configuration, the complete index pairs remain distinguishable and the framework does not classify the samples as a deterministic collision.

Conversely, if both Index1 and Index2 are identical between two dual-indexed samples in the same scope, the complete barcode configuration is indistinguishable. Such an assignment is therefore reported as a blocking deterministic collision.

The collision-detection logic can be represented as:

$$\text{Collision}_{SI}(a,b) = \begin{cases} 1, & I1_a = I1_b \\ 0, & I1_a \neq I1_b \end{cases}$$

for single-indexed samples, and

$$\text{Collision}_{DI}(a,b) = \begin{cases} 1, & (I1_a, I2_a) = (I1_b, I2_b) \\ 0, & \text{otherwise} \end{cases}$$

for dual-indexed samples, where \(I1\) denotes Index1 and \(I2\) denotes Index2.

The collision assessment is performed independently within each established validation group. Thus, an identical index assignment in two different validation scopes is not automatically reported as a collision.

The deterministic collision taxonomy implemented by RunPlanValidator is illustrated in Fig. 1.

E. Single-Index and Dual-Index Validation

RunPlanValidator treats single-index and dual-index libraries according to their actual barcode structure rather than applying one universal collision criterion. This distinction is necessary because the number of index components required to identify a sample differs between the two configurations.

For single-index libraries, Index1 provides the sample-identifying barcode. Consequently, duplicate Index1 values within a validation group constitute deterministic collisions.

For dual-index libraries, sample identification is based on the combination of Index1 and Index2. Therefore, the collision key is constructed from both values. Repeated Index1 sequences are permitted when the corresponding Index2 sequences distinguish the samples. Dual-indexing has been shown to improve multiplex sequencing accuracy and to reduce the effects of index misassignment under appropriate experimental designs [9], [11], [12].

This distinction between complete barcode identity and individual index-component reuse is important in practical multiplexed sequencing because barcode inventories may be limited. Requiring every individual index component to be globally unique would impose a substantially more restrictive condition than requiring uniqueness of the complete index combination.

The framework therefore applies the following principle:

Collision status is determined by the complete index configuration required by the applicable demultiplexing strategy, not by sequence similarity or isolated reuse of one index component.

F. Mixed Indexing Configurations

Real-world sequencing run plans may contain both single-index and dual-index libraries. RunPlanValidator was designed to process such heterogeneous configurations without requiring the user to divide the input workbook into separate files or invoke separate validation modes.

The indexing type is determined at the sample level based on the presence or absence of Index2. Consequently, a single worksheet may contain samples with only Index1 as well as samples with both Index1 and Index2.

The framework then applies the appropriate collision rule to each indexing configuration. Single-indexed samples are evaluated according to Index1 identity, while dual-indexed samples are evaluated according to the complete Index1–Index2 combination.

Mixed single-/dual-index comparisons are permitted by the implementation because the absence of an Index2 value in one sample does not produce an identical barcode configuration to a sample carrying a distinct Index1–Index2 combination.

This design is particularly relevant to operational environments where legacy libraries, different preparation protocols, or platform-specific indexing strategies may coexist within the same sequencing operation.

The distinction between lane-aware and lane-agnostic demultiplexing scopes is illustrated in Fig. 2.

G. Software Architecture and Implementation

RunPlanValidator was implemented in Python, with emphasis on transparency, portability, and integration with existing laboratory workflows. The implementation accepts Excel-formatted run plans and processes individual worksheets as independent units within a workbook.

The processing architecture consists conceptually of the following sequential stages:

Excel run-plan input → input normalization → indexing classification → validation-scope determination → deterministic collision detection → result aggregation → report generation

During input processing, the framework first reads the workbook and identifies the relevant run-plan fields. Values are normalized before samples are classified. The validator then establishes the appropriate grouping structure based on worksheet and lane information.

Collision detection is subsequently performed within each group. Detected conflicts are collected into a structured result representation containing information required for downstream reporting.

The implementation separates the underlying validation logic from the user interface and reporting layers. This allows the same collision-detection mechanism to be accessed through automated command-line workflows as well as through the graphical interface.

The framework was intentionally designed to avoid dependence on computationally intensive mismatch calculations. Because deterministic collision detection is based on exact identity of the relevant barcode key, the validation problem can be reduced to identifying duplicate keys within each validation scope.

H. Graphical User Interface

In addition to command-line execution, RunPlanValidator provides a graphical user interface (GUI) intended to make pre-run validation accessible to laboratory personnel who may not routinely use command-line bioinformatics tools.

The GUI provides a direct workflow for selecting the Excel run plan and initiating validation. The interface includes an Excel Run Plan input field with a browsing option, a Run ID/Prefix field, and a Validate Run Plan control. The resulting design intentionally minimizes the number of user-configurable parameters because the validation logic is derived primarily from the structure of the input run plan.

The GUI therefore functions as a laboratory-facing interface to the same deterministic validation engine used by the command-line workflow. This ensures that GUI-based validation and automated validation use the same underlying collision definitions rather than separate implementations.

The interface is intended to permit a laboratory user to select a run plan, execute validation, and review the resulting report before proceeding with sequencing. The GUI consequently serves as an accessibility layer over the computational validation workflow rather than as a separate validation system.

The graphical user interface and associated laboratory-user workflow are illustrated in Fig. 5.

I. Validation Outputs and Reporting

RunPlanValidator produces both machine-readable and human-readable outputs following completion of the validation process.

The principal structured output is a CSV collision report containing the deterministic conflicts identified during validation. This format enables the results to be inspected manually or incorporated into downstream laboratory quality-control procedures and automated workflows.

A human-readable PDF validation report is also generated. The report is intended to support routine quality assurance, documentation, and audit requirements.

The framework additionally provides exit codes to support integration with automated sequencing workflows. This allows validation status to be incorporated into pipeline logic, such that a run plan containing blocking deterministic collisions can be prevented from progressing automatically.

The reporting system therefore serves two complementary purposes:

- Operational review, through a readable validation report that allows laboratory personnel to inspect detected conflicts.
- Computational integration, through structured CSV output and exit codes suitable for automated workflows.

J. Evaluation Dataset and Test Scenarios

RunPlanValidator was evaluated using synthetic and real-world sequencing run plans representing a mixed-

platform research laboratory environment.
The evaluation was designed to represent a range of operational sequencing conditions rather than a single fixed run configuration. Run sizes ranged from 2 samples to 700 samples per run. The datasets included heterogeneous index lengths of 6 bp, 8 bp, and 10 bp, mixed single- and dual-index libraries, and both lane-aware and lane-agnostic configurations.
The evaluation included both positive and negative test scenarios. Positive collision cases consisted of run plans containing identical index configurations within the same demultiplexing scope. Negative cases included operationally valid configurations in which individual index components were reused but the complete index combinations remained distinguishable.
In particular, dual-index configurations in which the same i7 was paired with different i5 sequences were included to determine whether the framework would incorrectly classify partial index reuse as a collision. These configurations were expected to remain valid under the deterministic collision definition.
Run plans containing identical complete dual-index pairs within the same demultiplexing scope were used as positive collision cases. These configurations were expected to generate blocking validation errors.

K. Comparative Assessment of Distance-Based Validation

To assess the practical effect of using exact collision criteria rather than conservative sequence-distance rules, RunPlanValidator was compared with mismatch-based barcode validation approaches using Hamming-distance filtering.
Hamming distance represents the number of nucleotide positions at which two sequences differ and is widely used in barcode design and error-correction strategies. Maintaining an appropriate minimum distance between barcode sequences can improve the ability to distinguish intended barcode assignments in the presence of sequencing errors [13], [14].
However, the objective of validating an already constructed operational run plan is different from designing a new error-tolerant barcode library. In the present study, the principal question was whether two samples had been assigned an identical barcode configuration within the same demultiplexing scope, rather than whether their barcode sequences satisfied a predefined minimum sequence distance.
The comparative evaluation therefore examined the rejection behavior of conservative mismatch-based filters against the deterministic collision criteria implemented in RunPlanValidator. Conservative Hamming-distance filtering rejected approximately 30%–40% of operationally valid run plans, whereas RunPlanValidator produced zero false-positive rejections under the deterministic collision criteria evaluated in this study.
This comparison was not intended to demonstrate that distance-based validation is intrinsically incorrect. Rather, it was used to evaluate the suitability of the two approaches for different purposes: barcode-design/error-tolerance assessment versus deterministic pre-run collision detection.

L. Validation Criteria and Error Classification

A run plan was considered valid when no deterministic barcode collision was identified within any validation scope. A run plan was considered invalid when at least one pair of samples contained a collision according to the applicable single-index or dual-index rule.
For single-index libraries, an identical Index1 assignment within a validation group constituted a blocking error. For dual-index libraries, an identical Index1–Index2 pair within a validation group constituted a blocking error.
Collision records were retained with sufficient information to identify the affected samples and the conflicting index configuration. The validation output therefore provides a direct relationship between the detected error and the original run-plan entries, facilitating correction before sequencing.
The framework's evaluation demonstrated that deterministic collisions present in the tested run plans were consistently identified. Identical dual-index pairs within the same demultiplexing scope were correctly classified as blocking errors, while shared individual index components with distinct complementary indices were permitted.
The overall pre-run validation workflow, including run-plan input, index normalization, scope determination, collision detection, and generation of validation outputs, is summarized in Fig. 3.

3. Results

A. Run-Plan Validation Across Sequencing Configurations

RunPlanValidator was evaluated across sequencing run plans representing different sample numbers, index lengths, indexing configurations, and demultiplexing scopes. The evaluation included run plans containing 6-bp, 8-bp, and 10-bp indices, with sample counts ranging from 2 to 700 samples per run. Both single-index and dual-index configurations were examined, including run plans containing mixed indexing strategies.
The validator successfully processed the evaluated run plans and classified samples according to the indexing information provided in the input spreadsheets. The presence or absence of Index2 was used to distinguish dual-indexed from single-indexed configurations, while the presence of lane information determined whether validation was performed within individual lanes or across the complete run.
The evaluation demonstrated that the same validation workflow could be applied across different run-plan

sizes without requiring separate analysis procedures. This enabled small experimental run plans and larger production-oriented run plans to be processed using a common validation framework.

B. Deterministic Detection of Duplicate Index Assignments

The primary validation objective was to identify deterministic barcode collisions, defined as identical index assignments occurring within the same demultiplexing scope.

For single-indexed samples, the framework identified duplicate Index1 assignments within the applicable validation group. For dual-indexed samples, collision detection was based on the complete Index1–Index2 combination.

Run plans containing identical complete dual-index combinations were correctly identified as containing deterministic collisions. These collisions were classified as blocking errors because the affected sample assignments could not be uniquely distinguished on the basis of the supplied index configuration.

The validator retained the identities of the affected samples and the corresponding index assignments, allowing the detected conflict to be traced directly back to the original run plan.

This deterministic approach provided a direct relationship between the validation rule and the reported error: when the complete barcode key was duplicated within the same validation scope, the run plan was rejected.

C. Correct Handling of Partial Index Reuse

An important component of the evaluation was assessment of situations in which one index component was reused while the complete dual-index combination remained unique.

Several evaluated run plans contained samples sharing the same Index1 sequence but possessing different Index2 sequences. These configurations were not classified as deterministic collisions by RunPlanValidator.

This behavior is particularly important for dual-index sequencing because requiring every individual i7 and i5 sequence to be globally unique would impose a substantially more restrictive condition than requiring uniqueness of the complete index pair.

For example, consider two samples with the following assignments:

Sample	Index1 (i7)	Index2 (i5)
Sample A	Index-A	Index-X
Sample B	Index-A	Index-Y

Although Index1 is identical, the complete index combinations differ:

RunPlanValidator therefore treats the configuration as non-colliding.

$$(Index\text{-}A, Index\text{-}X) \neq (Index\text{-}A, Index\text{-}Y)$$

This behavior demonstrates that the framework distinguishes component-level index reuse from complete barcode duplication, thereby reducing unnecessary rejection of otherwise distinguishable sample assignments.

D. Interactive Visualization

The evaluation also examined the effect of validation scope on collision detection.

When lane information was available in the run plan, samples were evaluated within their corresponding lane groups. Consequently, identical index assignments occurring in different lanes were not automatically classified as collisions when those assignments belonged to independent validation scopes.

When lane information was absent, the framework applied global validation and evaluated all samples within the corresponding run-plan scope.

This distinction enabled the same validation engine to accommodate both lane-dependent and lane-independent sequencing configurations without requiring separate software implementations.

The results demonstrate that collision determination is therefore dependent on two dimensions:

- The index configuration assigned to each sample, and
- The demultiplexing scope within which that configuration is evaluated.

This scope-aware approach reduces false-positive collision reporting arising solely from reuse of an index configuration outside the relevant demultiplexing context.

E. Validation of Mixed Single- and Dual-Index Run Plans

RunPlanValidator was additionally evaluated using run plans containing both single-index and dual-index libraries.

The framework automatically determined the indexing configuration of each sample from the input fields and applied the corresponding validation rule. No separate user-defined classification was required.

Single-index samples were evaluated using their Index1 assignment, whereas dual-index samples were evaluated using their complete Index1–Index2 combination.

The mixed-index evaluation demonstrated that heterogeneous run plans could be processed within the same workbook and validation workflow. This capability is relevant to laboratory environments in which sequencing libraries generated using different preparation protocols or indexing strategies may coexist within the same operational run plan.

F. Detection of Blocking and Non-Blocking Conditions

The validation output distinguishes between configurations that represent deterministic collisions and those that remain operationally distinguishable.

A blocking condition was generated when the applicable barcode key was duplicated within the same validation scope. For single-index libraries, this corresponded to duplicate Index1 values. For dual-index libraries, this corresponded to duplicate Index1–Index2 pairs.

Conversely, configurations involving only partial index reuse were retained as valid when the complete barcode assignment remained distinct.

This distinction enables the validator to function as a practical pre-run quality-control tool rather than simply as a barcode similarity checker. The resulting validation decision is based on whether the planned sample assignments contain a deterministic ambiguity under the defined indexing configuration.

G. Sample- and Run-Level Validation Outputs

Following validation, RunPlanValidator generated structured outputs containing the detected conflicts and associated run-plan information.

The CSV output provides a machine-readable representation of the identified collisions, allowing individual conflicts to be examined or incorporated into downstream workflows. The output retains information necessary to identify the affected samples and their conflicting index assignments.

A corresponding PDF report provides a human-readable summary of the validation result and is intended for laboratory documentation and quality-control review.

The combination of structured and human-readable outputs allows the same validation event to support both computational and laboratory-facing workflows.

The output architecture also provides traceability between the original run-plan entries and the detected collision. This is particularly useful when large run plans contain many samples, because the user can identify the exact assignments requiring correction rather than manually reviewing the complete spreadsheet.

H. Evaluation Across Run Sizes

The framework was tested across run plans ranging from 2 to 700 samples, demonstrating applicability across substantially different run sizes.

The smallest test configurations provided controlled scenarios for evaluating individual collision rules and expected validation outcomes. Larger configurations were used to assess the behavior of the framework under substantially greater numbers of sample-index assignments.

Across the evaluated configurations, the same deterministic validation logic was retained irrespective of run size. No separate algorithm or user workflow was required for larger run plans.

This consistency is important for routine laboratory deployment because the validation process should remain independent of whether a sequencing run contains a small number of samples or approaches the capacity of a larger multiplexed run.

I. Evaluation Across Index Lengths

RunPlanValidator was evaluated using 6-bp, 8-bp, and 10-bp index sequences.

The framework did not require a fixed index length for deterministic collision detection. Because the primary comparison is based on exact equality of the relevant index value or index pair, the same validation logic can be applied to index sequences of different lengths.

This provides flexibility for laboratories using multiple sequencing platforms, library preparation protocols, or historical run plans in which index lengths may differ.

The index-length evaluation therefore demonstrated that the validation framework is based on the structural properties of the run plan rather than on a platform-specific fixed barcode length.

J. Comparison With Hamming-Distance Filtering

The performance of deterministic collision detection was additionally evaluated against conservative Hamming-distance-based filtering.

Hamming-distance filtering identifies barcode sequences that differ by fewer than a predefined number of nucleotide positions. Such approaches can be useful when designing barcode sets or assessing tolerance to sequencing errors. However, applying a minimum-distance criterion to an operational run plan may classify distinct barcode assignments as problematic even when they are not identical.

In the comparative evaluation, conservative Hamming-distance filtering rejected approximately 30%–40% of operationally valid run plans. In contrast, RunPlanValidator produced zero false-positive rejections under the deterministic collision criteria evaluated in this study.

The difference was particularly evident for configurations in which barcode sequences were similar but not identical. RunPlanValidator retained these configurations when the complete barcode assignments remained distinguishable, whereas the distance-based approach could reject them solely because they failed to meet the selected sequence-distance threshold.

These findings support the use of deterministic collision detection when the primary objective is to identify guaranteed barcode assignment conflicts rather than to impose a conservative sequence-separation threshold. The comparative rejection rates observed across the evaluated operational datasets are shown in Fig. 4.

K. Overall Validation Performance

Collectively, the evaluation demonstrated that RunPlanValidator successfully performed repository-independent pre-run validation across heterogeneous sequencing run plans.

The framework correctly:

- identified duplicate single-index assignments within the relevant validation scope;
- identified duplicate complete dual-index combinations;
- permitted valid reuse of an individual index component when the complete dual-index combination remained unique;
- incorporated lane information when available;
- supported global validation when lane information was absent;
- processed mixed single- and dual-index configurations;
- handled index sequences of different lengths;
- processed run plans ranging from 2 to 700 samples;
- generated structured CSV collision outputs; and
- generated human-readable validation reports.

The observed results indicate that the deterministic validation strategy provides a reproducible mechanism for identifying operationally meaningful barcode collisions while avoiding unnecessary rejection of configurations that remain distinguishable under the defined indexing model.

The results also establish the basis for the subsequent discussion of the framework's practical advantages, limitations, and potential applications in sequencing quality control.

4. Discussion

The increasing use of multiplexed next-generation sequencing (NGS) has made accurate sample identification an essential component of sequencing quality control. Index sequences allow multiple libraries to be pooled within a sequencing run and subsequently separated during demultiplexing. However, index misassignment and index cross-talk are recognized sources of error in multiplexed sequencing workflows and can result in reads being attributed to the wrong sample [15]–[17]. These effects are particularly relevant in applications involving large numbers of multiplexed samples or assays requiring high sensitivity for low-frequency variants.

RunPlanValidator addresses this requirement by providing a dedicated framework for pre-run validation of sequencing index assignments. Rather than replacing established sequencing or demultiplexing software, the framework operates before sequencing and evaluates the planned barcode configuration contained in the run plan. Its primary objective is to identify deterministic index collisions that could result in two samples sharing an indistinguishable barcode configuration within the same demultiplexing scope.

A distinguishing feature of the framework is its focus on deterministic collision detection rather than generalized barcode similarity assessment. Barcode design and error-correction studies have traditionally used sequence-distance measures, including Hamming distance, to establish sufficient separation between barcode sequences [13], [14]. Hamming-code-based barcode design has specifically been proposed to preserve minimum sequence distances and error-correcting properties in multiplex sequencing applications. Such approaches are valuable when constructing barcode libraries and designing sequences with tolerance to sequencing errors. However, the question addressed by RunPlanValidator is narrower: whether the barcode assignments already present in a proposed sequencing run contain an exact duplicate within the relevant operational scope.

A. Importance of Complete Index Combinations

The distinction between individual index components and complete index combinations is particularly important for dual-index sequencing. Dual-indexing uses two index reads to increase the number of distinguishable library assignments and can provide additional protection against index misassignment when appropriately designed [9], [11], [12]. Experimental studies have demonstrated that unique dual-index strategies can substantially reduce index cross-talk and allow unexpected index combinations to be identified or filtered [11], [15].

Accordingly, RunPlanValidator does not classify two dual-indexed samples as colliding merely because they share the same Index1 sequence. Instead, the complete Index1–Index2 combination is evaluated. A collision is reported only when the complete combination is duplicated within the same validation scope.

This approach prevents unnecessarily restrictive validation. Consider two samples sharing Index1 but carrying different Index2 sequences. Although one component of their barcode assignment is identical, their complete index combinations remain distinct. Treating such a configuration as an automatic collision would effectively impose a requirement for global uniqueness of each individual index component, which is more restrictive than requiring uniqueness of the complete dual-index assignment.

The observed behavior of RunPlanValidator in the evaluated datasets supports this distinction. Configurations involving shared individual index components but distinct complete index pairs were retained as valid, whereas duplicate complete index combinations were identified as blocking collisions.

This distinction also has relevance to index-hopping mitigation. Unique dual-index strategies use non-

redundant combinations to make unexpected index combinations identifiable after sequencing, thereby reducing the impact of index switching [11], [15].

B. Scope-Aware Validation and Lane Handling

Another important characteristic of the framework is its explicit treatment of demultiplexing scope. Barcode uniqueness cannot always be evaluated meaningfully across an entire sequencing operation without considering how the sequencing platform performs demultiplexing.

RunPlanValidator therefore distinguishes between lane-aware and global validation. When lane information is available, samples are evaluated within their respective lane groups. When lane information is absent, the framework evaluates the relevant samples within a global scope.

This design provides an important operational distinction from a simple spreadsheet-wide duplicate search. A duplicate index appearing in two independent validation scopes does not necessarily represent the same collision condition as a duplicate index assigned to two samples that are demultiplexed together.

The scope-aware design is particularly relevant to laboratories operating heterogeneous sequencing environments. Sequencing workflows may differ in their treatment of lanes, index reads, and demultiplexing procedures, and therefore the operational context in which an index is interpreted must be considered when evaluating whether reuse represents an actual conflict.

The importance of this distinction is reinforced by published observations that index misassignment can arise from multiple mechanisms, including sequencing-related errors and index hopping, rather than solely from duplicate index assignments in the original run plan [15]–[17].

RunPlanValidator does not attempt to model these post-library mechanisms. Instead, it establishes whether the planned index configuration itself contains a deterministic ambiguity within the defined validation scope.

C. Comparison With Hamming-Distance-Based Approaches

The comparison between RunPlanValidator and Hamming-distance-based filtering provides an important insight into the distinction between barcode design and run-plan validation.

Hamming distance represents the number of nucleotide positions at which two barcode sequences differ and is widely used in barcode design and error-correction strategies. Maintaining an appropriate minimum distance between barcode sequences can improve the ability to distinguish intended barcode assignments in the presence of sequencing errors [13], [14]. Bystrykh demonstrated the application of Hamming-code principles to DNA barcode design, emphasizing minimum-distance and error-correction properties of barcode sets.

However, applying the same criterion retrospectively to an operational run plan addresses a different problem. A minimum-distance filter may reject two valid, non-identical index assignments because their sequences fall within the selected distance threshold, even though the complete assignments are not deterministic duplicates.

In the present evaluation, conservative Hamming-distance filtering rejected approximately 30%–40% of operationally valid run plans, whereas RunPlanValidator produced zero false-positive rejections under the deterministic collision criteria evaluated in this study.

These are findings of the present study and therefore are not externally cited.

The significance of this result is not that Hamming-distance methods should be abandoned. Rather, it demonstrates that the appropriate validation criterion depends on the question being asked. If the objective is to design a barcode set that remains robust against sequencing errors, sequence-distance criteria are appropriate. If the objective is to determine whether a prepared run plan contains an exact duplicate barcode assignment within a common demultiplexing scope, deterministic identity provides a more direct criterion.

RunPlanValidator is therefore best viewed as complementary to, rather than competitive with, barcode-design and error-correction approaches.

D. Relationship to Index Misassignment and Index Hopping

It is important to distinguish pre-run index collisions from post-sequencing index misassignment. The latter can occur even when every sample has been assigned a unique index combination in the original run plan.

Index cross-talk and index hopping have been experimentally demonstrated in multiplexed sequencing workflows [16], [17]. Owens et al. reported measurable cross-talk during Illumina sequencing and demonstrated that quality filtering of index reads could substantially reduce misassignment. MacConaill et al. similarly demonstrated index cross-talk with combinatorial adapters and showed that unique dual-indexed adapters could substantially reduce this phenomenon.

Costello et al. further characterized index swapping across Illumina sequencing platforms and demonstrated the value of non-redundant dual indexing for identifying or removing reads associated with unexpected index combinations.

These observations establish an important boundary for RunPlanValidator. The framework does not claim to prevent every form of index misassignment. Instead, it addresses the earlier and more deterministic problem of whether the run plan itself contains duplicate assignments.

This distinction strengthens the positioning of the tool within the sequencing workflow. RunPlanValidator can be used before sequencing to identify preventable configuration errors, while established sequencing QC and demultiplexing procedures remain necessary to detect and mitigate errors arising during sequencing and

downstream processing.

E. Practical Advantages for Sequencing Laboratories

A major practical advantage of RunPlanValidator is that it transforms barcode validation into a dedicated quality-control step that can be performed before sequencing begins.

In conventional laboratory workflows, index assignments may be maintained in spreadsheets and manually inspected for duplicate combinations. As the number of samples increases, manual inspection becomes increasingly difficult and may be susceptible to human error. The problem becomes particularly challenging when lane information, single- versus dual-index configurations, and multiple sequencing platforms must be considered simultaneously.

RunPlanValidator addresses these issues by automating the comparison process and generating a structured validation result. The user does not need to manually compare every pair of samples or construct an ad hoc script for each sequencing run.

The graphical interface further lowers the computational barrier to adoption. A laboratory user can provide the run-plan spreadsheet, initiate validation, and review the generated output without requiring direct interaction with the underlying Python implementation.

This feature is particularly relevant to multidisciplinary sequencing environments in which run planning may be performed by molecular biologists, laboratory scientists, sequencing personnel, and clinical researchers with different levels of computational expertise.

F. Integration With Existing Sequencing Workflows

RunPlanValidator is designed as a pre-run quality-control layer rather than as a replacement for established sequencing or demultiplexing software.

Modern sequencing workflows already incorporate specialized tools for demultiplexing, read processing, quality control, alignment, and downstream analysis. RunPlanValidator addresses an earlier stage of the workflow by checking whether the planned index assignments contain deterministic conflicts before sequencing data are generated.

This positioning is advantageous because errors identified before sequencing can potentially be corrected without requiring resequencing or extensive downstream investigation. Once sequencing has been completed, an incorrectly assigned or ambiguous index can propagate into demultiplexed datasets and complicate subsequent analysis.

The framework's CSV and PDF outputs further support integration into existing laboratory quality systems. The CSV output can be incorporated into automated workflows, while the human-readable report provides documentation for laboratory review. Exit-code-based validation additionally permits the framework to be incorporated into automated pre-run pipelines in which a failed validation can prevent subsequent processing. Thus, the framework can operate either as an interactive laboratory tool or as an automated quality-control component.

G. Handling of Heterogeneous Run Plans

An important strength demonstrated by the evaluation is the ability to process heterogeneous run plans without requiring separate validation software for each indexing configuration.

The evaluated datasets included 6-bp, 8-bp, and 10-bp indices, single-index and dual-index configurations, mixed indexing arrangements, and lane-aware as well as lane-agnostic run plans.

The framework determines the relevant validation logic from the structure of the input data. This reduces the number of configuration decisions that must be made by the user and allows historical or operationally diverse run plans to be evaluated through a common workflow.

This flexibility is useful in research and diagnostic laboratories where sequencing infrastructure evolves over time. A repository of run plans may contain data generated using multiple instruments, library-preparation protocols, and indexing strategies. A validation framework that requires a separate configuration for every historical workflow would therefore be more difficult to maintain.

RunPlanValidator instead establishes a small number of explicit rules governing index type and validation scope. This provides a transparent basis for applying the same conceptual validation model across different run-plan structures.

H. Scalability and Operational Utility

The framework was evaluated across run plans ranging from 2 to 700 samples, demonstrating applicability across substantially different run sizes.

The underlying deterministic comparison is structurally simple: once the relevant validation scope and collision key have been established, duplicate configurations can be identified without performing exhaustive sequence-distance calculations between every possible pair of samples.

This is advantageous for routine pre-run validation because the computational task is based primarily on identifying duplicate barcode keys within defined scopes rather than repeatedly calculating sequence similarity.

The practical value of this approach is therefore not solely computational speed. Equally important is the ability to maintain the same validation logic as the number of samples increases. A run containing hundreds of samples

can be subjected to the same deterministic criteria as a small pilot run, reducing dependence on manual review.

I. Reproducibility and Standardization

Reproducibility is an important consideration in sequencing quality control because run-plan validation decisions can affect the subsequent generation and interpretation of sequencing data.

Manual spreadsheet inspection may produce different outcomes depending on the person performing the review or the complexity of the run plan. By encoding the collision rules explicitly within software, RunPlanValidator provides a standardized decision process.

For a given input run plan, the same normalization, scope determination, indexing classification, and collision rules are applied consistently. The resulting CSV and PDF outputs also provide a record of the validation outcome.

This reproducibility is particularly valuable for laboratories operating formal quality-management procedures, where evidence of pre-analytical quality checks may be required. The framework's structured output can provide a standardized record of the validation performed on a specific run plan.

J. Limitations

Several limitations should be considered when interpreting the current implementation.

First, RunPlanValidator is specifically designed to detect deterministic index collisions. It does not model the complete probability of index misassignment caused by sequencing errors, index hopping, base-calling errors, or other platform-specific mechanisms. These phenomena are important considerations in sequencing quality control and have been extensively described in the literature [15]–[17]. The framework should therefore be regarded as one component of a broader sequencing quality-control strategy.

Second, the current implementation assumes that the input run plan accurately represents the index sequences and sample assignments that will ultimately be used for sequencing. A software validator cannot detect an error introduced after validation if the physical library or sequencing configuration differs from the validated run plan.

Third, the interpretation of lane-specific validation depends on the correctness of the supplied lane information and on the actual demultiplexing behavior of the sequencing workflow. Incorrect or incomplete lane metadata could consequently affect the validation scope.

Fourth, the deterministic model does not replace barcode-design considerations. Two non-identical barcodes may remain relatively close in sequence space and could warrant consideration under a sequencing-error-tolerance framework even though they do not represent deterministic duplicates. Such assessments fall outside the primary objective of the present tool and may appropriately be addressed using barcode-design or error-correction methods [13], [14].

Finally, the current framework is centered on Excel-based run plans and the index structures represented in the present implementation. Broader interoperability with additional run-plan formats and sequencing-platform metadata would further increase its applicability.

K. Future Development

Several extensions could broaden the utility of RunPlanValidator.

Future versions could incorporate configurable validation policies that allow laboratories to perform both deterministic collision detection and optional sequence-distance assessment. Such a design would preserve the current exact-matching workflow while providing users with additional information regarding potentially vulnerable barcode combinations.

Support for additional sequencing platforms and run-plan formats could also improve interoperability. Direct integration with instrument-generated sample sheets and platform-specific metadata would reduce the need for manual conversion between operational run plans and validation inputs.

Another potential development would be integration with barcode inventories and laboratory information-management systems (LIMS). Such integration could permit validation against institutional records of previously used index combinations and could assist in preventing accidental reuse of restricted barcode assignments.

Automated integration with sequencing workflows represents another important opportunity. The existing machine-readable outputs and exit-code mechanism provide a foundation for incorporating validation into automated pre-run pipelines, where a sequencing run could be blocked automatically when a deterministic collision is detected.

Finally, future versions could provide richer audit trails, validation summaries, and visualization of index relationships. Such features could make it easier for laboratories to identify recurrent planning errors and evaluate the overall quality of their indexing strategies over time.

L. Overall Significance

The principal contribution of RunPlanValidator is not the introduction of a new sequencing or barcode technology, but the formalization and automation of a practical quality-control problem that is frequently handled manually or through laboratory-specific scripts.

Existing literature has established the importance of indexing strategies, dual indexing, barcode separation, and control of index misassignment [9]–[17]. RunPlanValidator builds upon these principles but addresses a

different operational requirement: determining whether a proposed sequencing run plan contains a deterministic barcode collision before the run is initiated.

The evaluation demonstrates that this distinction can have substantial practical consequences. By applying exact collision criteria within the appropriate demultiplexing scope, the framework can identify genuine duplicate assignments while avoiding unnecessary rejection of configurations containing partial index reuse.

Overall, RunPlanValidator provides a lightweight, reproducible, and accessible quality-control framework for sequencing run-plan validation. Its combination of deterministic collision detection, lane-aware scope handling, support for heterogeneous indexing configurations, graphical accessibility, and standardized reporting positions it as a complementary tool within modern multiplexed sequencing workflows.

The framework is therefore best understood as a pre-analytical sequencing quality-control layer that complements existing barcode-design, sequencing, demultiplexing, and downstream bioinformatics tools. By identifying deterministic index conflicts before sequencing, it provides an opportunity to correct run-plan errors at an early stage and thereby reduce avoidable downstream complications.

5. Conclusion

RunPlanValidator presents a computational framework for automated pre-run validation of sequencing index assignments within multiplexed next-generation sequencing workflows. By combining deterministic index-collision detection, validation-scope handling, graphical interaction, and standardized reporting within a single application, the framework provides a practical approach for identifying potentially conflicting sample assignments before sequencing is initiated.

A key feature of the framework is its distinction between complete index collisions and partial index reuse. For single-index configurations, duplicate Index1 assignments within the relevant validation scope are identified as conflicts, whereas dual-index configurations are evaluated using the complete Index1–Index2 combination. This approach allows valid reuse of individual index components to remain permissible when the complete barcode combination remains unique.

The evaluation demonstrated that RunPlanValidator can process heterogeneous sequencing run plans containing different index lengths, indexing configurations, lane structures, and sample numbers. The framework was successfully evaluated across run plans containing 2 to 700 samples and 6-bp, 8-bp, and 10-bp indices, demonstrating applicability across a range of operational scenarios.

The framework also provides a practical alternative to purely manual inspection of sequencing run plans. Through its graphical interface, users can perform validation without constructing custom scripts, while the generated CSV and PDF outputs provide both machine-readable and human-readable representations of the validation results. This combination supports reproducibility, documentation, and integration into laboratory quality-control workflows.

Importantly, RunPlanValidator is intended to complement rather than replace existing sequencing, demultiplexing, barcode-design, and quality-control tools. The framework specifically addresses deterministic conflicts present in the planned run configuration and does not attempt to model all sources of post-sequencing index misassignment, including index hopping or sequencing-related errors [15]–[17].

The comparison with conservative Hamming-distance filtering further demonstrates the value of distinguishing deterministic collision detection from broader barcode-distance assessment. While sequence-distance approaches remain valuable for barcode design and error-tolerance considerations [13], [14], the present framework focuses specifically on identifying exact assignment conflicts that can be detected before sequencing.

Future development of RunPlanValidator will focus on extending platform and input-format compatibility, incorporating optional sequence-distance assessment, integrating with laboratory information-management systems and sequencing workflows, and expanding reporting and audit capabilities. These enhancements could further increase the utility of the framework in laboratories performing high-throughput multiplexed sequencing.

Overall, RunPlanValidator provides an accessible, reproducible, and scalable pre-analytical quality-control framework for sequencing run-plan validation. By identifying deterministic barcode conflicts before sequencing, the framework provides laboratories with an opportunity to correct preventable sample-index assignment errors at an early stage, thereby supporting more reliable downstream sequencing and bioinformatics workflows.

ACKNOWLEDGMENT

The authors thank the developers of the open-source bioinformatics tools and scientific computing libraries used during the development of RunPlanValidator. Their continued contributions have significantly advanced computational genomics and enabled the development of reproducible bioinformatics software.

References

1. Illumina, “Indexed Sequencing Overview,” Illumina Technical Documentation.

2. M. Kircher, S. Sawyer, and M. Meyer, "Double indexing overcomes inaccuracies in multiplex sequencing on the Illumina platform," Nucleic Acids Research, vol. 40, no. 1, Art. no. e3, 2012.
3. M. Costello, M. Fleharty, J. Abreu, et al., "Characterization and remediation of sample index swaps by non-redundant dual indexing on massively parallel sequencing platforms," Genome Research, vol. 28, no. 12, pp. 1856–1864, 2018.
4. G. L. Owens, M. Todesco, E. B. M. Drummond, et al., "Minimizing index hopping in multiplexed sequencing libraries," Bioinformatics, vol. 34, no. 3, pp. 371–377, 2018.
5. L. E. MacConaill, R. T. Burns, N. Nag, et al., "Unique, dual-indexed sequencing adapters effectively eliminate index cross-talk in Illumina sequencing," Genome Biology, vol. 19, 2018.
6. E. S. Wright and K. H. Vetsigian, "Quality filtering of Illumina index reads mitigates sample cross-talk," BMC Genomics, vol. 17, Art. no. 876, 2016.
7. M. E. Hamady, A. M. Walker, J. A. K. Harris, et al., "Barcode design and error correction for multiplexed sequencing," Nucleic Acids Research, relevant methodological literature on barcode sequence-distance design.
8. Illumina, Indexed Sequencing Overview, Illumina Technical Documentation.
9. M. Kircher, S. Sawyer, and M. Meyer, "Double indexing overcomes inaccuracies in multiplex sequencing on the Illumina platform," Nucleic Acids Research, vol. 40, no. 1, Art. no. e3, 2012.
10. Illumina, Effects of Index Misassignment on Multiplexing and Downstream Analysis, Technical Note, Illumina, 2017.
11. M. Costello et al., "Characterization and remediation of sample index swaps by non-redundant dual indexing on massively parallel sequencing platforms," Genome Research, vol. 28, no. 12, pp. 1856–1864, 2018.
12. L. E. MacConaill et al., "Unique, dual-indexed sequencing adapters effectively eliminate index cross-talk in Illumina sequencing," Genome Biology, vol. 19, 2018.
13. E. S. Wright and K. H. Vetsigian, "Quality filtering of Illumina index reads mitigates sample cross-talk," BMC Genomics, vol. 17, Art. no. 876, 2016.
14. A. A. Bystrykh, "Generalized DNA barcoding and genotyping using universal DNA barcode sequences," PLoS ONE, vol. 7, no. 5, e36852, 2012.
15. Illumina, Effects of Index Misassignment on Multiplexing and Downstream Analysis, Technical Note 770-2017-004, Illumina, 2017.
16. E. S. Wright and K. H. Vetsigian, "Quality filtering of Illumina index reads mitigates sample cross-talk," BMC Genomics, vol. 17, Art. no. 876, 2016.
17. L. E. MacConaill, R. T. Burns, A. Nag et al., "Unique, dual-indexed sequencing adapters with UMIs effectively eliminate index cross-talk and significantly improve sensitivity of massively parallel sequencing," BMC Genomics, vol. 19, Art. no. 30, 2018.

Data availability—RunPlanValidator source code and documentation are available at https://github.com/ankurc17/RunPlan. Test datasets are available upon request.

Competing interests
The authors declare no competing interests.

Appendix A: Figures

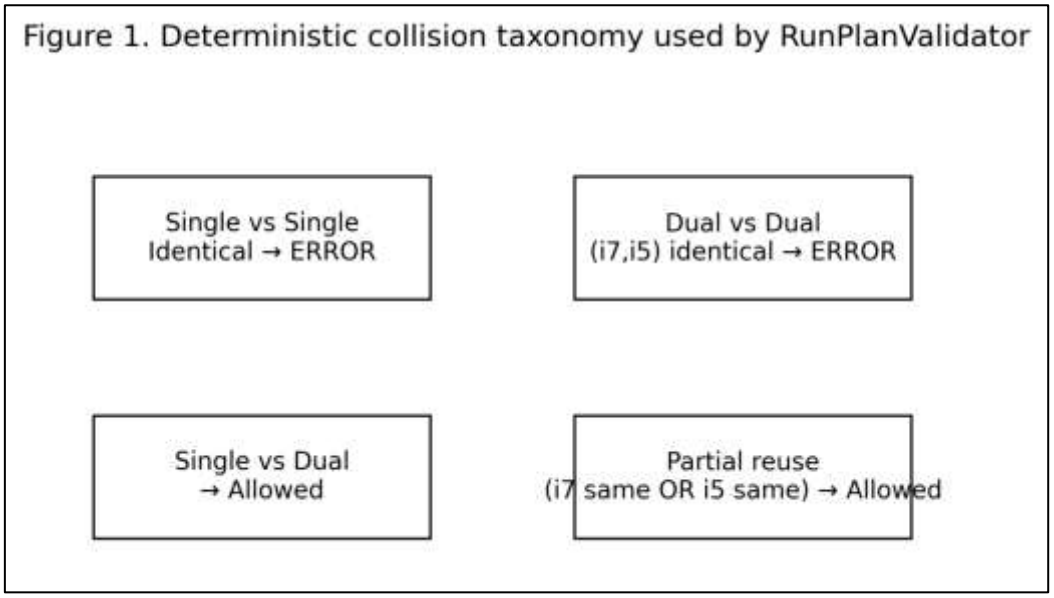

Figure 1. Deterministic collision taxonomy used by RunPlanValidator.

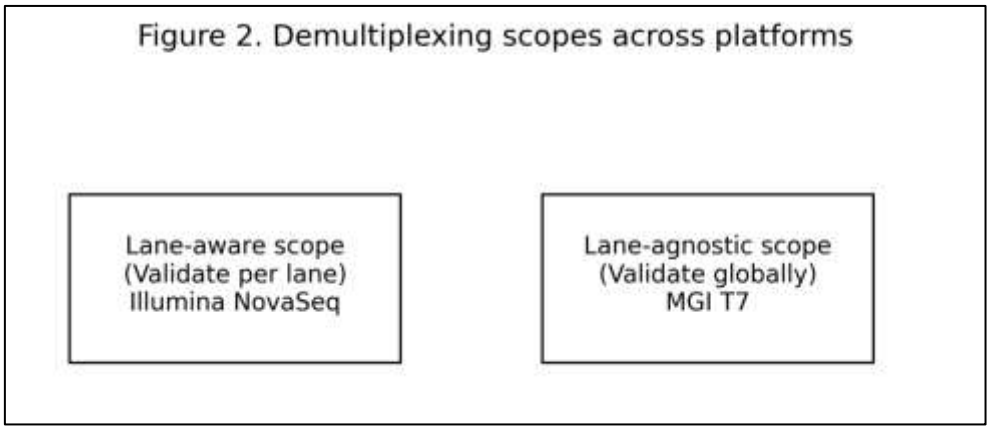

Fig.2. Demultiplexing scopes across platforms (lane-aware vs lane-agnostic)

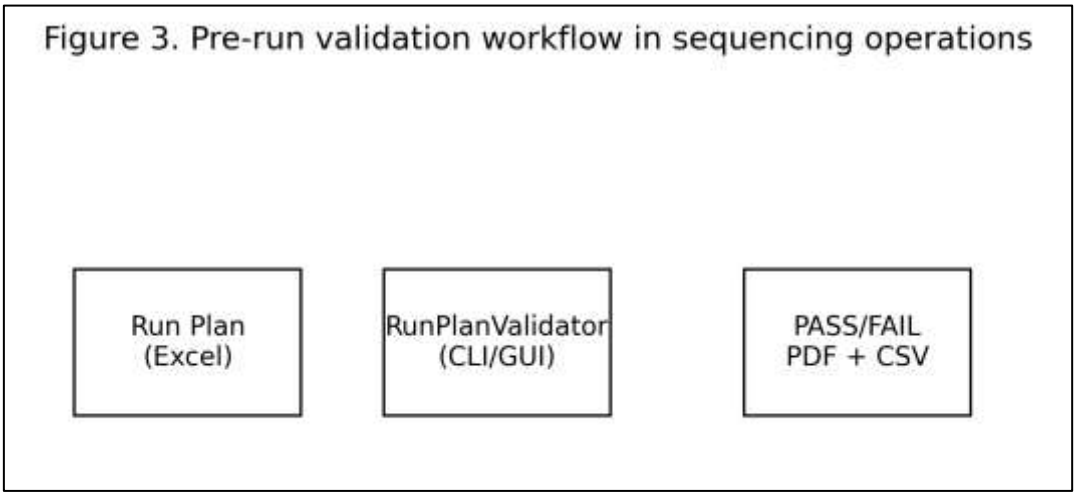

Fig.3. Pre-run validation workflow in sequencing operations

The computational workflow implemented in RunPlanValidator. Users specify a genomic variant using chromosome, genomic position, reference allele, and alternate allele. The framework recursively scans an institutional repository containing Tabix-indexed VCF files, performs indexed variant retrieval and exact genomic matching, extracts sample-level genotype information, computes repository-level Internal Variant Frequency (IVF), generates cohort summary statistics, and produces interactive visualizations and standardized CSV reports.

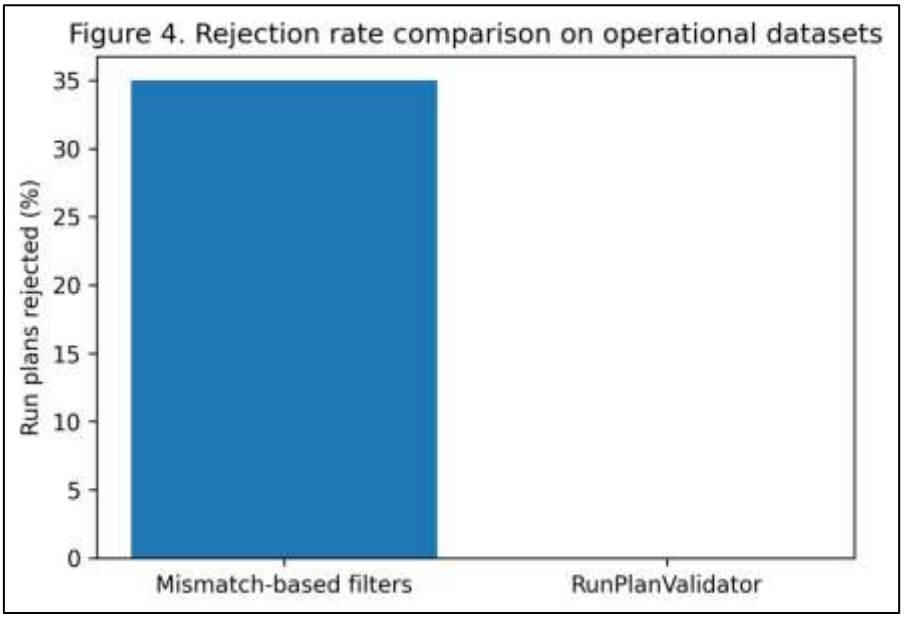

Fig.4. Rejection rate comparison on operational datasets.

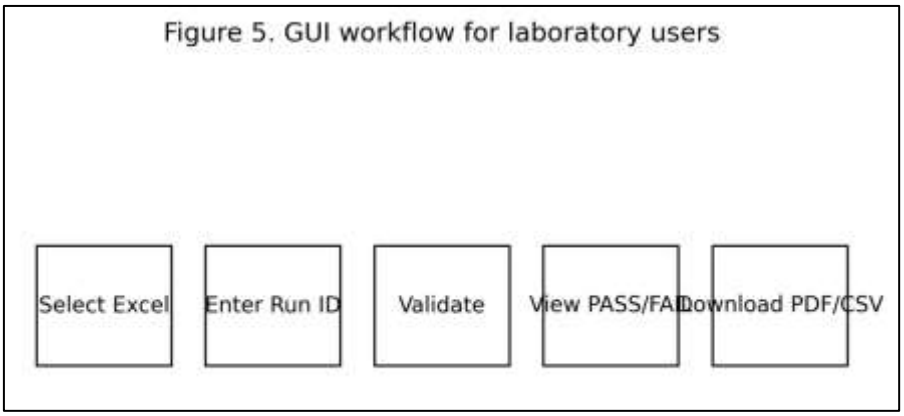

Fig.5. GUI workflow for laboratory users.

Appendix B: Tables

Table I. compares key demultiplexing characteristics and operational implications across Illumina and MGI platforms. This comparison motivates the design of lane-aware (optional) validation in RunPlanValidator

Feature	Illumina (e.g., NovaSeq)	MGI (e.g., T7 / G400)	Implication for RunPlanValidator
Demultiplexing scope	Lane-aware demultiplexing	Lane-agnostic demultiplexing (T7); lane-aware on some models (G400)	Supports optional lane column; GLOBAL scope when lane absent
Indexing strategy	Single, Dual, UDI supported	Single and Dual supported	Mixed single/dual index validation supported in same run
Index length support	6–10 bp (kit dependent)	6–10 bp (kit dependent)	Variable index lengths supported
Common failure mode	Index collisions within lane; index hopping	Index collisions within run scope; index hopping	Deterministic collisions blocked pre-run; stochastic hopping out of scope
Run plan format	Spreadsheet / sample sheet	Spreadsheet / sample sheet	Excel multi-sheet ingestion supported
Operational constraints	UDI not always available at scale	UDI not always available at scale	Permits partial reuse; blocks only true collisions