

EFFICIENT DATA STRUCTURES FOR CROSS-SAMPLE INFERENCES ON GENOMIC DATA

POLITECNICO DI MILANO

Dipartimento di Elettronica, Informazione e Bioingegneria



Vahid JALILI

Thesis advisors:

Prof. Matteo Matteucci

Prof. Marco Masseroli

Thesis supervisor:

Prof. Stefano Ceri

Milan, Italy

Politecnico di Milano 2016

*Wings are a constraint that
makes it possible to fly.*
Robert Bringham

To my wife, son, and parents . . .

Acknowledgements

Though only my name appears on the cover of this dissertation as the author, a great many people have contributed to its production. I owe my gratitude to all those people who have made this dissertation possible and because of whom my graduate experience has been one that I will cherish forever.

Foremost, my deepest gratitude is to my advisor Prof. Matteo Matteucci for the continuous support of my PhD study and related research, for his patience, motivation, enthusiasm, and immense knowledge. His guidance helped me in all the time of research and writing of this thesis. I could not have imagined having a better advisor and mentor for my PhD study. I have been amazingly fortunate to have an advisor who gave me the freedom to explore on my own, and at the same time the guidance to recover when my steps faltered. Prof. Matteucci taught me how to question thoughts and express ideas, he taught me going after my dreams, do not loose sight of them, fight for them, and materialize them. His patience and support helped me overcome many crisis situations and finish this dissertation. I am also thankful to him for encouraging the use of correct grammar and consistent notation in my writings and for carefully reading and commenting on countless revisions of all the manuscripts I wrote during my graduate experience. I hope that one day I would become as good an advisor to my students as Prof. Matteucci has been to me.

My sincere thanks also goes to Prof. Stefano Ceri for insightful comments and constructive criticisms at different stages of my research were thought-provoking and they helped me focus my ideas. Additionally, I am thankful to him for giving me the opportunity to join his talented research group. I am indeed thankful to prof. Marco Masseroli for all his kind support during my graduate experience, for letting me join their lab, and particularly for his careful revisions and constructive critic comments that improved our publications.

Besides my advisors and supervisor, I would like to thank my thesis reviewer Prof. Stefano Rizzi for his insightful comments and encouragement, but also for the hard question which incented me to widen my research from various perspectives.

I thank my fellow labmates, the Genome computing group at Politecnico di Milano, for all the fun we have had in the last three years. In particular, I am grateful to Fernando Palluzzi

Acknowledgements

for enlightening me the glance of Genomics. I am thankful to Francesco Venco, Pietro Pinoli, Arif Canakoglu, Abdulrahman kaitoua, Yuriy Vaskin, and Stefano Perna for the stimulating discussions. I am indeed thankful to Marzia Angela Cremona, Yashar Deldjoo, Mehdi Elahi, and Massimo Quadrana for their kindest and insightful support in statistical and analytical challenges of present dissertation.

Last but not the least, I would like to thank my family: my wife, my parents, and to my sister for supporting me spiritually throughout my graduate experience writing. None of this would have been possible without the love and patience of my family. My immediate family to whom this dissertation is dedicated to, has been a constant source of love, concern, support and strength all these years. I would like to express my heart-felt gratitude to my family.

V. J.

Abstract

Advances in next generation sequencing (NGS), also known as high-throughput sequencing, ubiquitize DNA sequencing as a flexible tool for genome exploration. NGS has opened the possibility of a comprehensive characterization of the genomic and epigenomic landscapes, giving answers to fundamental questions for biological and clinical research, e.g., how DNA-protein interactions and chromatin structure affect gene activity, how cancer develops, how much complex diseases such as diabetes or cancer depend on personal (epi)genomic traits. This is opening the road to personalized and precision medicine.

A distinguished aspect of NGS-based experiments is the large amount of data they produce. The generated data are broadly applicable and facilitate various functional analysis, including studies about DNA-protein interaction or histone modification (using Chromatin immunoprecipitation followed by massively parallel DNA sequencing (ChIP-seq)), transcriptional regulation (using RNA-seq), long range chromatin interactions explained by *de novo* spatial structure of genome (using Hi-C6). Recent studies combine these studies into larger assays for in-depth interpretations of sequencing data. Yet such interpretations, and *making sense* of data, demand complex computation and large scale data retrieval systems. The present dissertation has focused on *sense-making*, e.g., discovering how heterogeneous DNA regions concur to determine particular biological processes or phenotypes. Towards such discovery, characteristic operations to be performed on region data regard identifying co-occurrences of regions, from different biological tests and/or of distinct semantic types, possibly within a certain distance from each others and/or from DNA regions with known structural or functional properties.

The manuscript explains Di4 (1D interval incremental inverted index) and its predecessor Di3 (1D interval inverted index). Di4 and Di3 are single-dimension (1D) multi-resolution indexing frameworks, designed to be comprehensive, generic, extensible, and scalable back-end data structures for information retrieval on NGS interval-based data. Di4 and Di3 are defined at data access layer, agnostic to data, business logic, and presentation layers; this design makes them adaptable to any underlying persistence technology based on key-value pairs, spanning from classical B+tree to LevelDB and Apache HBase, and it makes them suitable for different business logic and presentation layer scenarios. Benchmarking Di4 and Di3 on real and simulated datasets and a comparison with common tools in bioinformatics realm, demonstrate the effectiveness of Di4 and Di3 as a back-end for general purpose genomic region manipulation.

Acknowledgements

The applicability of Di4 and Di3 to different business logic and presentation scenarios, and extensibility to application-specific functions, is assessed in comparative evaluation of ChIP-seq samples. The ChIP-seq technology identifies protein-DNA interactions using enriched regions on DNA when the significance measure (p-value) is below a stringency threshold. Replicated samples are expected to have a degree of repeated evidence, which can locally lower the minimum significance required to accept an enriched region. The present dissertation discusses a method for the joint analysis of ChIP-seq replicates, which confirms overlapping enriched regions if their comparative evaluation complies a set of user-defined thresholds. The method is implemented using Di3 to demonstrate the extensibility of the frameworks. Additionally, Di3 is used as back-end data structure to implement common ChIP-seq data assessment methods, such as functional analysis, correlation assessment, nearest feature distance distribution, chromosome-wide statistics, and genome browser.

Keywords: Genomic computing; indexing framework, domain-specific data indexing; region-based operations and calculus; data integration; comparative analysis.

Sommario

Grazie agli sviluppi nel sequenziamento di nuova generazione (NGS), anche noto come High-Throughput Sequencing, il sequenziamento del DNA si è diffuso ovunque come strumento flessibile per l'esplorazione del genoma. Le tecniche di NGS hanno reso possibile una caratterizzazione globale del panorama genomico ed epigenomico, rispondendo a domande fondamentali nell'ambito della ricerca biologica e chimica quali l'effetto sull'attività genica delle interazioni tra proteine e DNA, nonché della struttura della cromatina, le modalità di sviluppo di diverse forme di cancro e la dipendenza di malattie complesse come diabete dai tratti (epi)genomici dell'individuo. A loro volta questi sviluppi stanno aprendo la strada alla cosiddetta medicina di precisione, o personalizzata.

Una caratteristica peculiare degli esperimenti basati sull'NGS è la grande quantità di dati da essi prodotti. Questi dati sono applicabili in diversi contesti e facilitano diversi tipi di analisi funzionali, ad esempio gli studi sull'interazione DNA-proteine o le modificazioni degli istoni (usando l'immunoprecipitazione della cromatina seguita dal sequenziamento massivo e parallelo del DNA – il ChIP-Seq), la regolazione della trascrizione (RNA-Seq) oppure l'interazione a lungo raggio della cromatina spiegata attraverso una struttura spaziale del genoma de novo (usando Hi-C6). Lavori più recenti hanno combinato queste ricerche in studi più estesi sull'interpretazione approfondita dei dati di sequenziamento; tuttavia, sia queste interpretazioni, sia l'attività stessa di estrapolare un significato dai dati, richiedono calcoli onerosi e sistemi di estrazione dati all'altezza della quantità di dati disponibile. Questa tesi si concentra sul sense-making, ovvero sull'indagare come regioni eterogenee del DNA concorrano nel determinare particolari processi biologici o fenotipi. Al fine di effettuare tali indagini, alcune operazioni di base devono essere effettuate su dati di regioni provenienti da diversi test biologici e/o diversi tipi di dato semantico, al fine di identificare eventuali co-occorrenze e, possibilmente, a quale distanza ciascuna di esse si manifesti rispetto alle altre e/o ad altre regioni del DNA la cui struttura o le cui proprietà funzionali siano già note.

Il manoscritto descrive Di4 (un indice inverso incrementale per intervalli monodimensionali) ed il suo predecessore non incrementale Di3. Di4 e Di3 sono framework di indicizzazione monodimensionali (1D) multi-risoluzione, pensati per essere strutture dati generiche, estensibili, globali e scalabili per effettuare ricerca e recupero di informazioni su dati NGS di natura intervallare. Di4 e Di3 sono definiti a livello di accesso dati e sono agnostici al dato, alla logica

Acknowledgements

dei processi che lo elaborano e al livello di presentazione dello stesso; questa definizione permette loro di essere adattabili a qualsiasi pre-esistente tecnologia di salvataggio su disco che sia basata su coppie chiave-valore, a partire dal classico B-tree fino a LevelDB ed a Apache Hbase e questo li rende adatti per diversi scenari di utilizzo. Esperimenti di benchmark su Di4 e Di3 effettuati sia su dataset reali sia sintetici ed un confronto con strumenti già esistenti nel campo della bioinformatica hanno mostrato l'efficacia di questi indici come back-end per la manipolazione, a qualunque fine, delle regioni genomiche.

L'applicabilità di Di4 e Di3 a diversi scenari di utilizzo e la loro estendibilità a funzioni specifiche sono state valutate tramite indagini comparative di campioni di ChIP-Seq. La tecnologia ChIP-Seq identifica le interazioni proteina-DNA usando regioni arricchite sul DNA con un valore di significatività (pvalue) al di sotto di una certa soglia critica. Ci si aspetta che campioni duplicati abbiano un certo grado di replicazione nei risultati, cosa che può ridurre a livello locale la minima significatività richiesta per accettare una regione arricchita. In questa tesi viene presentato un metodo per l'analisi incrociata dei duplicati di ChIP-Seq al fine di validare in modo più significativo regioni arricchite sovrapposte. Il metodo è stato implementato usando Di3 al fine di dimostrare l'estendibilità del framework. In aggiunta, Di3 viene utilizzato come struttura dati di back-end per implementare altri metodi più comuni di valutazione del dato ChIP-Seq, quali analisi funzionali, valutazioni di correlazione, distribuzione della distanza delle feature più vicine, statistiche a livello di cromosoma e visualizzatori di genoma.

Contents

Acknowledgements	i
Abstract	iii
List of figures	xi
List of tables	xiii
1 Introduction	1
1.1 Genomics primer	1
1.2 Problem Statement	8
1.3 Thesis Contribution	9
1.4 Structure of the Thesis	11
2 Background and State of the Art	13
2.1 State of Art Tools in Genomics for Region Calculus	14
2.2 Related Work from Classical Data Structures	15
2.3 Related Work from Temporal Databases	16
3 Di3: 1D Intervals Inverted Index	21
3.1 Di3 Design	21
3.1.1 Snapshots keys organization	23
3.1.2 Snapshots values organization	24
3.2 Operations supported by Di3	26
3.3 Di3 Intervals	28
3.3.1 First Resolution	30
3.3.2 Second Resolution	31
3.4 Retrievals	34
4 Di4: 1D Intervals Incremental Inverted Index	37
4.1 Interval indexing beyond Di3	37
4.2 Di4 Objectives	39
4.3 Di4 Design	42
4.4 Notation	43
4.5 Di4 First Resolution	46

Contents

4.6	Di4 Second Resolution	48
4.6.1	Grouping	48
4.6.2	Aggregation	49
4.7	Di4 Indexing Algorithms	49
4.7.1	Bookmarking intervals on First Resolution	50
4.8	Information Retrieval based on Intervals	57
4.8.1	Data Structure Reconstruction	60
4.8.2	Cover	67
4.8.3	Summit	71
4.8.4	Map	72
5	Di4 and Di3 Performance Evaluation and Comparison	75
5.1	Performance Evaluation Setup	75
5.2	Comparison between inverted index and incremental inverted index	78
5.2.1	Comparison of indexing speed	79
5.2.2	Comparison of index file size	80
5.2.3	Comparison of retrievals	81
5.2.4	Performance evaluation based on different degrees of parallelization	84
5.2.5	Performance evaluation based on different persistence setup	86
5.3	Comparison of Di3 and Di4 performance with common bioinformatics tools	86
6	Di3 Application in Comparative Analysis of ChIP-seq Replicates	93
6.1	Introduction	93
6.2	Related Works	96
6.2.1	Binary analysis	97
6.2.2	Alignment read Merging	97
6.2.3	Irreproducibility Discovery Rate	98
6.2.4	joint Analysis of Multiple ChIP-seq Datasets	98
6.3	Definitions	99
6.4	Combining Replicates	100
6.4.1	Authenticity of Combining Replicates	100
6.4.2	Combining Test Statistics	102
6.4.3	Method of Combining Replicates	104
6.4.4	Combining replicates using Di3	107
6.4.5	Threshold Automatic Validation	107
6.4.6	Example	109
6.5	Functional Annotation and Analysis of Enriched Regions	112
6.5.1	Motivation	112
6.5.2	Algorithm	112
6.5.3	Graphical User Interface	113
6.6	Nearest Neighbor Distance Distribution	113
6.6.1	Motivation	113
6.6.2	Algorithm	114

6.6.3	Graphical User Interface	114
6.7	Global Correlation Assessment	115
6.7.1	Motivation	115
6.7.2	Algorithm	119
6.8	Genome Browser	119
6.8.1	Algorithm	120
6.9	Results	120
6.9.1	Simulated Technical Replicates	122
6.9.2	Evaluation of Technical Replicates	123
6.9.3	Evaluation of Biological Replicates	126
7	Toward Google-Style Search in Genomics	133
7.1	Introduction	133
7.2	Pattern Finding Queries on Di4	134
7.2.1	Sample Pattern Rank	136
7.2.2	Region Pattern Rank	139
	Conclusion and Future Works	146
	Bibliography	162
	Publication: Using combined evidence from replicates to evaluate ChIP-seq peaks	163
	Publication: MuSERA: Multiple Sample Enriched Region Assessment	172
	Publication: Indexing Next Generation Sequencing Data [SUBMITTED]	187

List of Figures

1.1	DNA structure	2
1.2	Sequence analysis	3
1.3	ChIP-seq workflow	4
1.4	ChIP-seq peak calling	5
1.5	DNA sequencing cost	7
1.6	Di4 Schematic Design	10
3.1	Di3 data model	23
3.2	Di3 key organization	25
3.3	Di3 value organization	26
3.4	An example of COVER function	28
3.5	An example of SUMMIT function	29
3.6	An example of MAP function	29
3.7	An example of ACCHIS function	29
3.8	The first and second resolutions of Di3	31
3.9	An illustration of single-pass compared to double-pass indexing	32
3.10	A comparison between single-pass and double-pass indexing	33
4.1	Design differences between Di3 and Di4	38
4.2	Di4 Schematic Design	40
4.3	Di4 Architecture	41
4.4	Di4 data structure	45
4.5	Single-pass vs. Double-pass indexing	51
4.6	Example of functions COVER, SUMMIT, MAP, and ACCHIS	59
4.7	A sample region for decomposition	61
4.8	Candidate, Selected, and Reconstruction regions	62
4.9	Reconstruction of intervals of different types	62
5.1	Application design of Di4	76
5.2	Interval/region accumulation distribution in datasets.	78
5.3	First pass indexing speed of Di3 and Di4 on four datasets	79
5.4	Comparison of Di3 and Di4 indexing elapsed time and index file size	81
5.5	Benchmark of retrieval functions	83
5.6	Performance evaluation based on different degrees of parallelization	85

List of Figures

5.7	Di4 operations performance correlation with persistence setup	89
5.8	Di4 index size on correlation with persistence setup	90
5.9	Benchmark: On-the-fly processing scenario	90
5.10	Benchmark: Personal repository scenario	91
6.1	Schematic view of comparative analysis of ChIP-seq replicates.	95
6.2	MuSERA architecture on extending Di3	96
6.3	An example of grouping intervals in different sets.	101
6.4	Possible overlapping conditions with three replicates.	104
6.5	The flowchart of combining replicates	106
6.6	Automatic confirmation of intersecting peaks	109
6.7	MuSERA features: functional annotation and analysis	114
6.8	MuSERA features: Nearest Neighbor Distance Distribution	116
6.9	Correlation assessment hierarchy	117
6.10	Sample portion on genome	118
6.11	The Calculated similarity of Figure 6.10 example	119
6.12	Integrated Genome Browser	121
6.13	Technical replicates results	125
6.14	Biological replicates results	127
6.15	Sequence logos for the Position Weight Matrices	130
6.16	Biological replicates results ($C = 2$)	131
7.1	Sample Pattern Rank, Example 1.	137
7.2	Sample Pattern Rank, Example 2.	138
7.3	Sample Pattern Rank, Example 3.	139
7.4	Sample Pattern Rank, Example 4.	140
7.5	Region Pattern Rank, Example 1.	141

List of Tables

2.1	Functionality comparison of classical data structures	16
4.1	Taxonomy of Di4 functions	59
5.1	Di3BCLI commands.	77
5.2	Datasets used for Di3 and Di4 benchmarking.	77
5.3	Specification of machines used for performance evaluation	77
6.1	Groups of ERs based on p-values	99
6.2	Datasets used for evaluation of comparative analysis	122
6.3	Statistics of Simulated Technical Replicates.	123
6.4	Statistics of alternative simulated technical replicates.	124
6.5	P-values for the enrichment of the E-box in technical replicates.	124
6.6	Percentages of ERs overlapping with DNase-seq data in technical replicates . .	126
6.7	Percentages of ERs overlapping with DNase-seq data in biological replicates ($C = 1$)	128
6.8	P-values for the enrichment of the E-box in biological replicates ($C = 1$)	129
6.9	Percentages of ERs overlapping with DNase-seq data in biological replicates ($C = 2$)	129
6.10	P-values for the enrichment of the E-box in biological replicates ($C = 2$)	130

1 Introduction

Next generation sequencing; an embraced technology by life science realm that integrates ideologies, analysis practices and toolkits for encyclopedic studies such as personalized medicine. It's data production cost is rapidly degrading, and is a ubiquitized standard; yet generated data size outpace analytical and computational capacities. Present chapter discuss pressing challenges in analysis, retrieval, interpretation, accessibility, and reproducibility — in general, *sense-making* — on NGS data, and our contribution to tackle rising challenges.

1.1 Genomics primer

Genomics propelled biology for the last hundred years by sparking scientific quest to decipher the nature of genetic material. 1865-1900: Mendelian inheritance and Boveri-Sutton theory defines cellular basis of heredity; the chromosomes. 1930s : biochemistry, genetics and other biological and physical disciplines converged to establish the molecular basis of heredity; the DNA double helix structure. 1970s : determination of precise order of DNA building blocks and discovery of biological mechanism for DNA-to-protein translation unlocked information basis of heredity; DNA sequencing. Present: an incessant exploratory to decipher genes, genetic regulatory networks, and ultimately entire genome spawning the field of Genomics.

The genome encodes how an organism evolves, functions and reproduces. It consists of DNA (Deoxyribonucleic acid) which is a double-stranded molecule with double helix structure (see Figure 1.1 ¹). Each DNA strand is a very long sugar-phosphate backbone of four nucleotides Adenine (A), Guanine (G), Cytosine (C) and Thymine (T) bound together by covalent bonds. The two DNA strands store same biological information, are anti-parallel, and are held together by hydrogen bounds between complementary nucleotides (i.e., A with T, and C with G). Genes are special subsequence of nucleotides and are molecular units of heredity; they encode ribonucleic acid (RNA) and proteins (gene expression); and proteins perform functions in the organism. Regulation of gene expression controls the quantity of RNA and proteins produced by specific gene. Gene expression regulation is a function of variety of factors including cell

¹Source: <https://en.wikipedia.org/wiki/DNA>

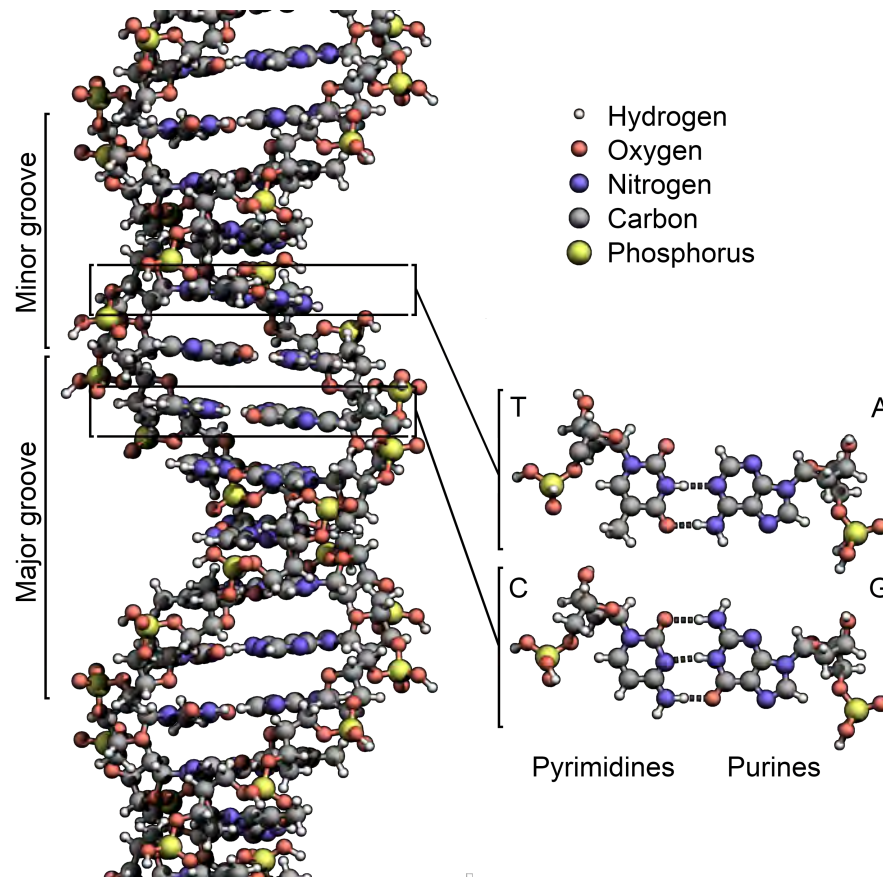


Figure 1.1: DNA structure.

type and determines how a cell functions. For instance, genes responsible for producing the oxygen-carrying protein hemoglobin are relatively up-regulated in blood cells as opposed to other cell types.

The procedure of studying a biological phenomena of an organism commonly starts by sequencing the DNA (primary analysis), follows by assembling the sequences (secondary analysis), and ends by analysis of the constructed representation of the chromosome (tertiary analysis) (see Figure 1.2). The DNA sequencing is the process of determining the precise order of four nucleotides. There has been various sequencing technologies developed since 1977 when Maxam-Gilbert published first sequencing technique known as chemical sequencing method [1]; and since the beginning of twenty-first century Next-Generation Sequencing (NGS) became the standard DNA sequencing technology. However, neither of current sequencing technologies can read DNA as one piece, rather they read it as a collection of short fragments² of nucleotides (e.g., AACGTACCG) – referred-to as nucleic acid sequences, or sequence reads, or raw reads.

²The length of the fragments varies from few tens of nucleotides to tens-of-thousands depending on the technology and the machine in use (e.g., NGS fragments span few hundreds of nucleotides).

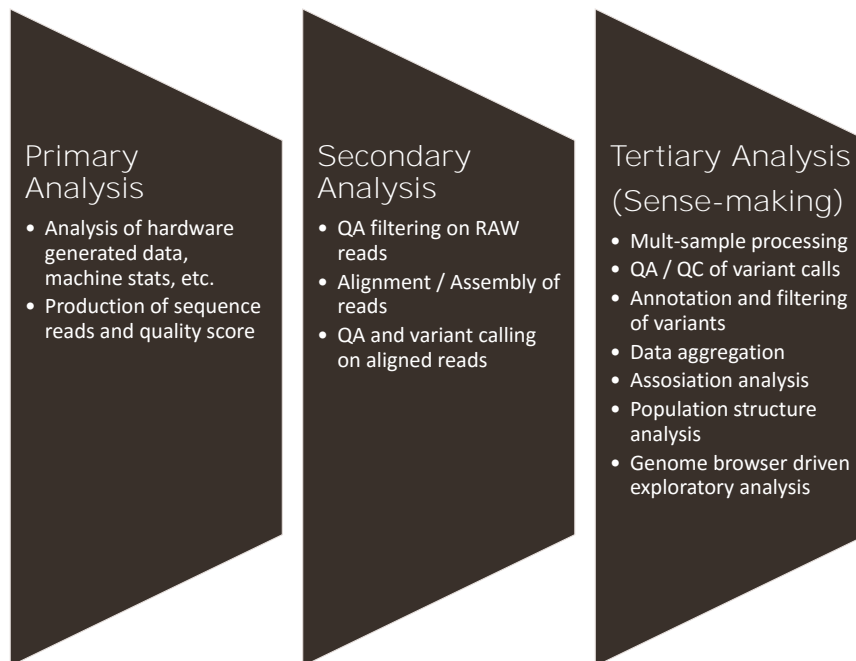


Figure 1.2: Sequence analysis.

The procedure of assembling DNA fragments depends on the objectives of the DNA sequencing; the two common objectives are: (1) *full genome sequencing* to determine a complete representation of whole genome of the organism by merging together the DNA fragments using *sequence assembly* procedures; (2) study of functional and structural characteristics of the genome by mapping DNA fragments to a position on a full genome sequence as the reference using *sequence alignment* procedures. Ultimately, the assembled DNA fragments are subjected to various analysis spanning from structural and functional analysis to genome-wide association studies, variant annotation and motif analysis.

Analysis of the assembled/aligned fragments tend to get the attention as a common procedure to disclose biological phenomena of a study (sense making). The NGS technology propelled the analysis by offering ChIP-seq (chromatin immunoprecipitation followed by massively parallel DNA sequencing) (see Figure 1.3 ³) to study complex biological characteristics such as DNA-protein-antibody interactions; and RNA-seq technology for studying features such as gene expression, single nucleotide variation, or fusion gene detection. The NGS pipeline explains biological phenomena of an experiment by two types of data: (1) raw reads, (2) genomic intervals which are relatively enriched positions on DNA with respect to the background signal that highlight the phenomena of the conducted experiment. The representations mainly differ in the level of abstraction a biological phenomena is disclosed, and the significant disparity in the data sizes. Therefore, the data management and analysis challenges are different.

³Source: <https://www.bnl.gov/newsroom/news.php?a=11351>

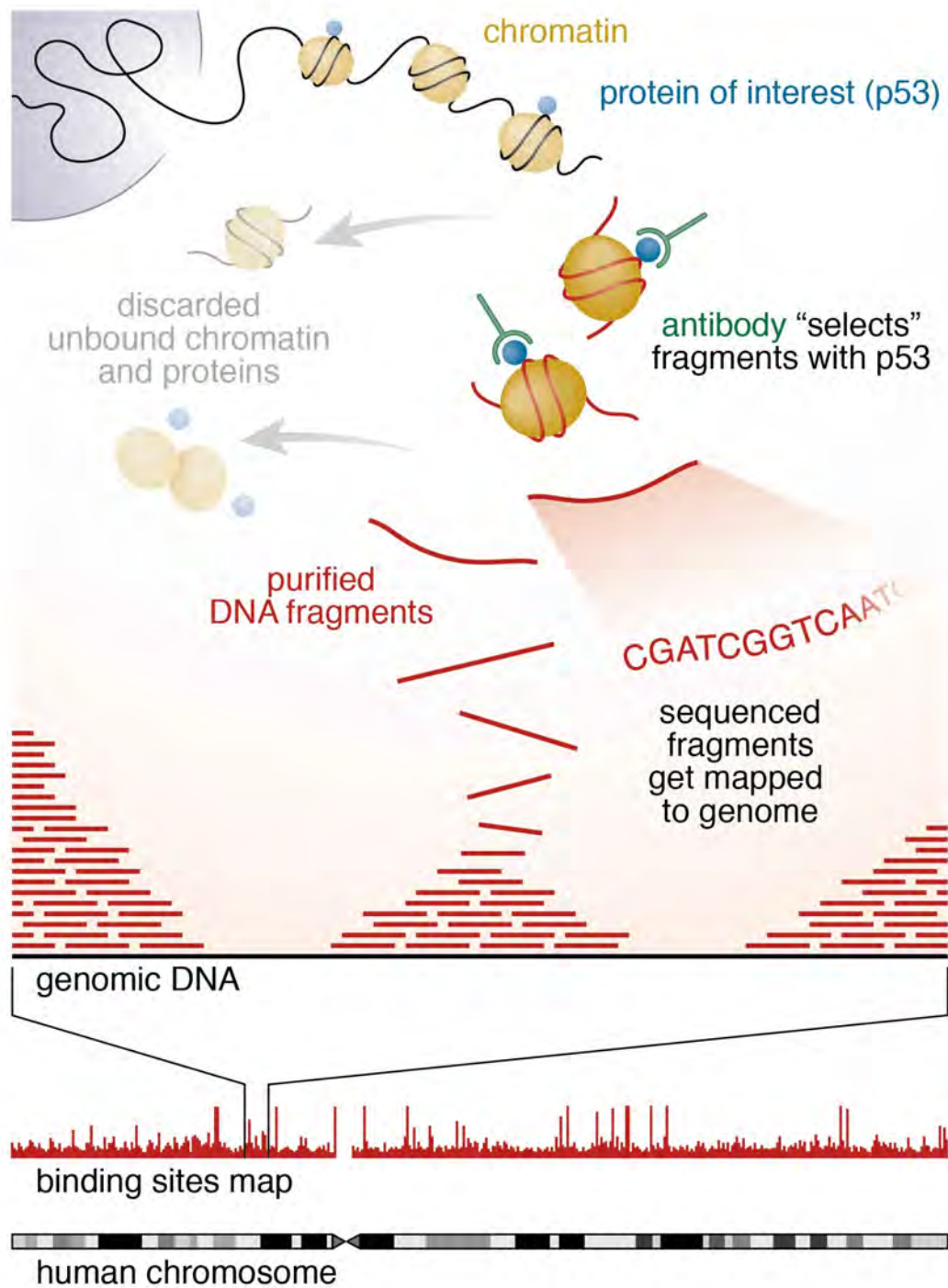


Figure 1.3: ChIP-seq workflow.

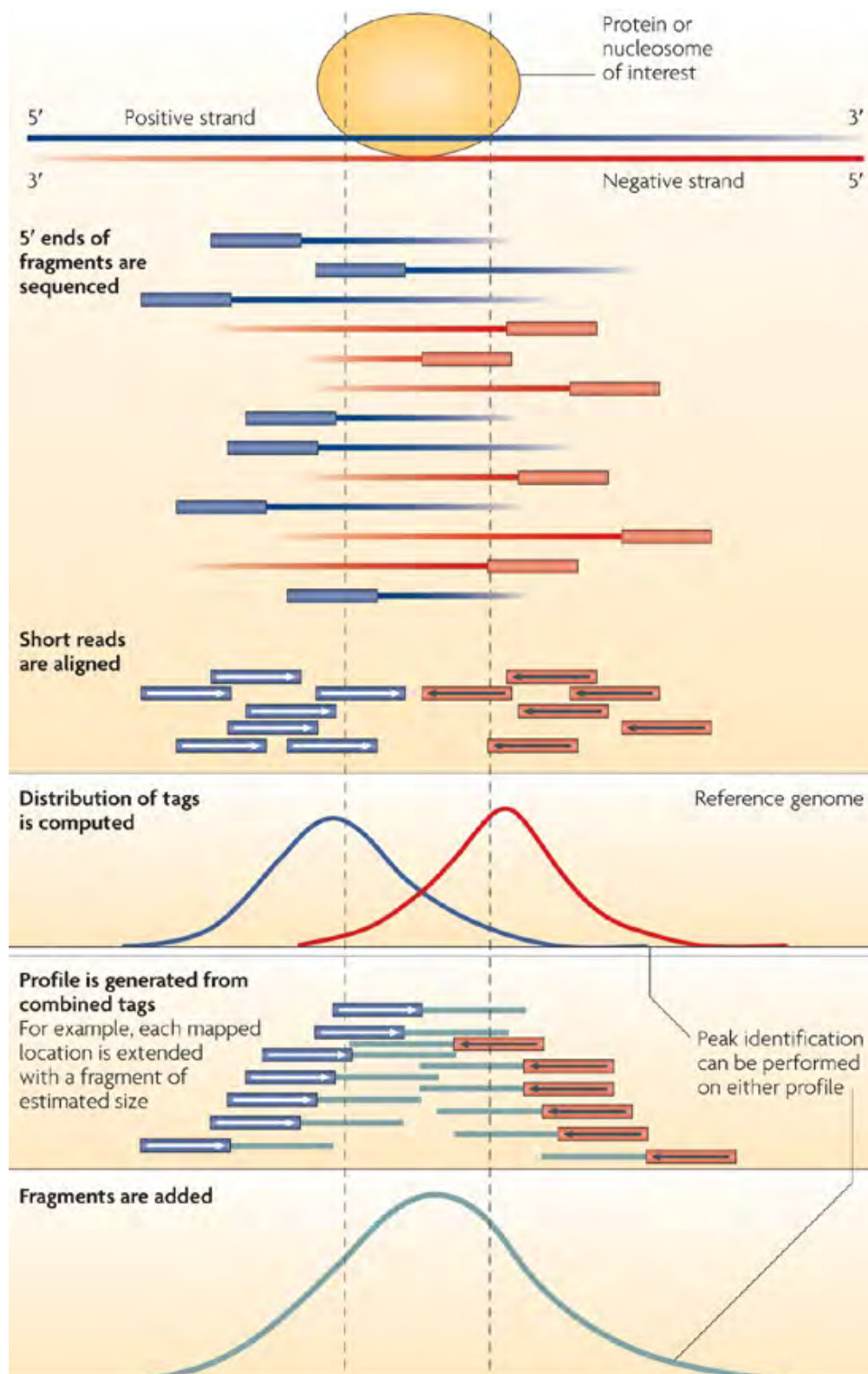


Figure 1.4: ChIP-seq peak calling.

The ChIP-seq and RNA-seq technologies generalize complex events such as DNA-protein-antibody interactions in terms of intervals, commonly known as binding sites or enriched regions on the DNA domain [2][3] (see Figure 1.4; source: [4]). The intervals of an experiment are determined by various processes such as *peak calling* [5][6] and are collected in tab-delimited files commonly referred-to as *samples*. The genomic intervals delegate various DNA characteristics such as (dis)regulations of DNA biological function that express particular genes. For instance, studying Transcription Factor (TF) proteins — presented as binding sites — intervals can disclose a TFs' binding profile and accordingly transcription regulatory networks (e.g., [7] [8]). Chromosome Confirmation Capture (3C) [9] is a technique that explains long-range chromatin interactions in terms of *de novo* 3D spatial organization of DNA using intervals (e.g., Chromatin Interaction Analysis by Paired-End Tag Sequencing (ChIA-PET) [10] [11]).

Interest in genomics data is skyrocketing as the DNA sequencing cost is rapidly degrading (from \$100M in 2001 to few thousand dollars in 2014 per genome [12], see Figure 1.5⁴) and produced data size is growing exponentially [13]. For instance, the National Center for Biotechnology Information (NCBI) expanded the Gene Expression Omnibus (GEO) repository from few hundreds of samples in 2000 to 550K in 2010 [14]. Additionally, a distinguished aspect of NGS pipeline is the large amount of data it produces for each experiment that multiplies rapid data production by larger data size for each sample. Ultimately, profound biological questions are generally compiled into challenging Information Retrieval (IR) ⁵ (e.g., [16] [17]) and data analysis tasks, which composed with data expansion rate become the genomics big data problem [18] [19]. Let consider the query “find transcriptional repressor CTCF in ChIP-seq samples of human cell lines that are at 100kb (kilo base-pair) range from given transcription start site”, it is a challenging query given the large domain size; and jMOSAICS [20] –an ad-hoc application that jointly assesses ChIP-seq enriched regions of a sample across multiple replicates– employs 30min on a single-CPU machine with Intel®Xeon®3.0 GHz processor and 64-bit architecture for processing four replicates. It would be challenging to extend the algorithm for similar inference applied on a larger number of samples. Therefore, genomics tend to get the attention of various disciples to tackle the genomics big data challenge.

Studying a biological phenomena is a traverse on primary, secondary, and tertiary analysis, with a fundamental difference between the steps in the data production cost, generated data sizes, and execution time; and accordingly different challenges arise. The primary analysis commonly starts by experimental design, continues by wet-lab preparation and ends at machine execution which in overall is an intrinsic time consuming process (e.g., Illumina HiSeq 2500 sequencing machine requires 3-12 days in standard mode to sequence each provided sample) and commonly outputs data in *FASTQ* file format with average size of 180GB

⁴Source: <https://www.genome.gov/sequencingcosts/>

⁵From the perspective of either search criteria, domain, or both. For instance, from criteria perspective “find all binding sites in provided sample that co-localize with human genes of hg19 assembly where provided TFs' regulating these genes are enriched to a given extend”, or domain perspective “determine average expression rate of all ChIP-seq peaks intersecting promoter regions of hg19 on ENCODE [15] for related cell lines”.

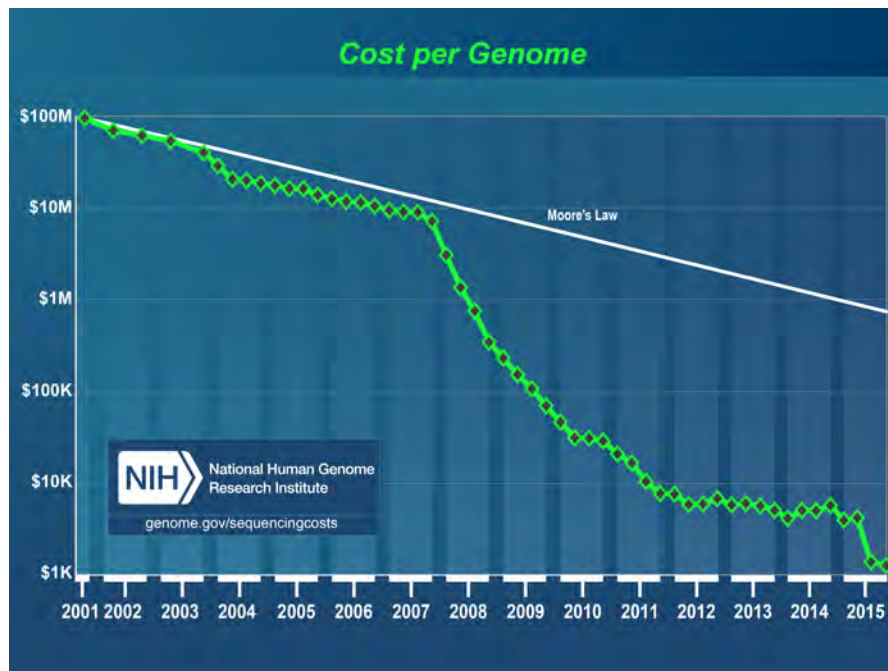


Figure 1.5: DNA sequencing cost.

for human genome.⁶ The secondary analysis operates upon data produced by primary analysis for sequence alignment/assembly and variant calling. The procedures are relatively faster than primary analysis and generated data are commonly in *SAM* (Sequence Alignment Map), *BAM* (Binary Alignment Map) and *VCF* (Variant Call Format) format which are of the same scale as primary analysis output. The tertiary analysis spans a wide range of inferences initiating from the output of secondary analysis such as variant annotation or peak calling. The tertiary analysis procedures are considerably faster and produced data size are smaller in order of magnitude.

On the other hand, the diversity of inferences grows exponentially from primary to tertiary analysis, which motivates one-to-many relation between the data of two consecutive analysis steps. For instance, suppose a study that defines an experiment and accordingly a FASTQ file is generated and the fragments are aligned in a BAM file, and the targeted inference is obtained by executing a particular pipeline of tertiary analysis. Another study with similar experimental setup may leverage on the same BAM file as a second replicate for joint tertiary analysis. Additionally, another study may apply a different pipeline on the same BAM file to infer from a different aspect of the experiment. Therefore, the cardinality and diversity of tertiary analysis data is growing exponentially as different labs executing different pipelines on in-house or publicly available primary and secondary analysis data.

⁶Human genome has 3G base-pairs (i.e., 3×10^9), and each base-pair takes 2Bytes (1Byte for base-pair and 1Byte for quality score), therefore with sequencing coverage of 30x, one full human genome sequence in FASTQ format takes $3G \times 2 \times 30 = 180G$ Bytes. The FASTQ format is commonly compressed to 25% of original size.

The data production cost and generated data size rapidly degrades from primary to tertiary analysis. In contrast, the cardinality of generated data grows exponentially from primary to tertiary analysis. Additionally, while genomic aspects are generally abstracted as sequences of nucleotides at primary and secondary analysis, tertiary analysis has interval-based representation. Therefore, the differences between the analysis steps necessitate different strategies for data management and processing. The interest in tackling the challenges logically follows data production pipeline and hence primary and secondary analysis got relatively more attention than tertiary analysis so far. While various ad-hoc and systematic solutions for optimal data management and analysis are developed for sequence reads, genomic intervals has only recently drawn the attention. Henceforth, this manuscript concentrates on genomic intervals.

1.2 Problem Statement

The sequencing machines encapsulate DNA fragments into files and the pipelines processing fragments export data in *samples* (files) which motivates genomic data management research to mainly focus on file systems. File systems coordinate physical access to data and are specifically agnostic, such that, do not commonly enforce any structure on data. This characterization has performance and generality advantages for wide-range of data types. However, it is not strictly a good practice for querying, parallelization, data integrity and security on data that intrinsically hold a structure. In addition, file system based management permits poor practices such as duplications (i.e., personal copies) and encoding meta-data in filename.

In contrast, database management systems (DBMS) orchestrate both physical and logical access to data, and maintains intrinsic and logical structure of data. In addition to addressing file system based data management challenges (e.g., redundancy control), a DBMS offers features for an improved access to data. In general, the features are as follows: *concurrent access* that in addition to allowing multiple users and processes concurrent access to data, it improves data sharing, *efficient querying* that allows fast retrieval on data by indexing pre-determined attributes, *extensibility and scalability* that are key features for todays application scenarios (e.g., genomics), which are well supported by DBMS, *transaction support* that via atomic operations guarantees data accuracy and integrity, and allows data recovery having abnormal termination of application.

Analysis of genomic intervals is the *sense-making* step which is commonly conducted by various manipulations and trial-and-errors either individually, or jointly or cross-referenced by supplementary in-house or publicly available data; therefore systematic solutions for optimal retrievals, data management and analysis are highly demanded.

Additionally, similar to many other disciplines, genomics is the science of exploration [21], and, to support that, different technologies are adapted including decision support systems (DSS), knowledge-based DSS (KDSS) (DSS and KDSS in computational biology are reviewed in [22]), and workflow management systems (WFMS) (integrating local and online resources spanning data and software, e.g., Taverna [23]) that render transparent technical details. Tertiary analysis

is a composition of manipulations spanning different disciplines such as data retrieval and mining, statistical analysis, and linear algebraic manipulations. Current DSS, KDSS, and WFMS are not fully comprehensive such that “sense-making” requirements commonly force scientists to roll their proprietary solutions, typically by integrating existing “building blocks”, which possibly end up with a solution that is difficult to scale, maintain and reuse.

Exploration is conceived of analytical and computational challenges. Remarkable efforts are done for analytical aspects, e.g., SciDB [24]. SciDB is an analytical database mainly focused on statistical and linear algebra operations such as matrix multiply, covariance, inverse, best-fit linear equation solution, and singular value decomposition. SciDB has a multidimensional array data structure with petabyte scalability, and performs significantly faster than related works ([25]: benchmarking SciDB vs Postgres, Hadoop (where data management is in Hive, and analytical operations are in Mahout), and column-based storage systems). However, beside the analytical challenges, a computational challenge that affects the applicability, scalability, and performance of sense-making systems is the underlying storage system. Present dissertation is focused on a model for genomic data and a back-end data structure for comprehensive, extensible, scalable, and efficient retrieval.

1.3 Thesis Contribution

The present dissertation is focused on algorithms and data structures for efficient retrieval from NGS data. It explains *1D intervals inverted index* (Di3) and *1D intervals incremental inverted index* (Di4); these are single-dimension multi-resolution generic indexing frameworks over interval-based data to support query execution and coordinate-oriented retrievals. This dissertation is genomic-centered, and explains Di3 and Di4 in favor of “sense-making”. The data model maps positions on domain (e.g., Genome) to observed events (e.g., genomic activities), by indexing primary attribute of intervals — coordinates: *left* and *right* — in a NoSQL key-value pair paradigm. The proposed indexes excel retrieval and data mining operations on occurrence and co-occurrence of intervals by rendering transparent the complexities of efficient, scalable, extensible, and comprehensive information retrieval system.

Intervals of input data are organized in a multi-resolution paradigm. The first resolution provides detailed information about the coordinates of intervals and has two versions: inverted index (Di3), and incremental inverted index (Di4). The second resolution aggregates first resolutions to optimize retrieval on on-indexed attributes. See Figure 1.6.

In general, Di3 and Di4 aims at efficient execution of the following operations on genomic data:

- **Similarity search:** given a set of samples, find most similar set of samples in terms of co-occurrence of intervals.
- **Range queries:** for instance, find regions on domain where a specific number of intervals co-occur.

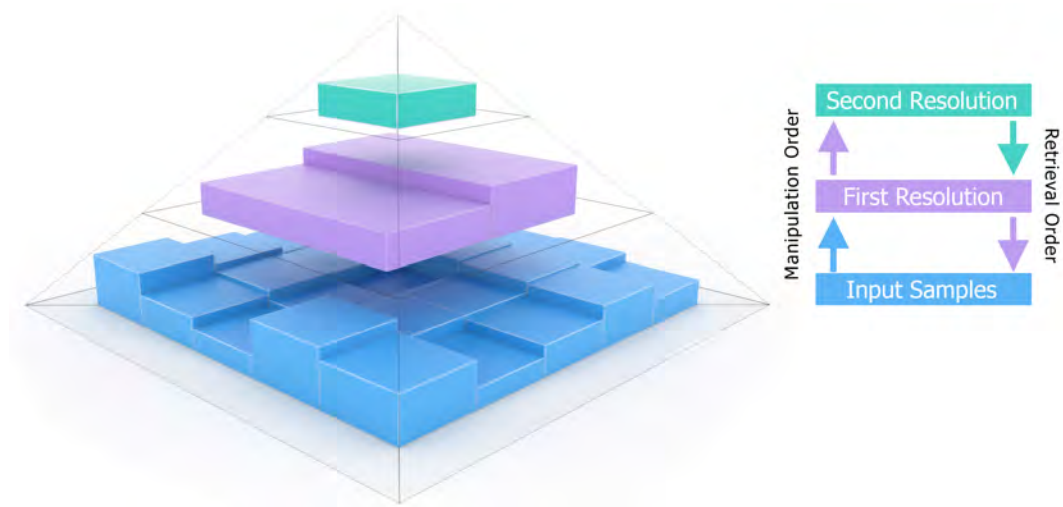


Figure 1.6: Di3 and Di4 schematic design; the coordinates of intervals from input samples are indexed in a double-resolution paradigm. First resolution indexes the coordinates in inverted, and incremental inverted structures. Second resolution aggregates the non-indexed attributes of groups of intervals. The size of persisted information decreases from input samples to the second resolution, with inverted index of first resolution being relatively larger in size compared to incremental inverted index. The manipulation of data structures logically starts from input samples and goes to second resolution through first resolution. However, the retrieval is conducted in reverse order.

- **Region calculus:** e.g., given an interval, find all intervals overlapping with it.
- **Data mining:** co-occurrence patterns, dependency detection (i.e., given a dataset, determine the positions on which the regions usually co-occur - with respect to the data in repository), deviation detection (i.e., given a dataset, find positions on which the regions do not commonly co-occur - based on the data in repository).

The proposed indexes are generic and designed for re-usability and extensibility over any domain with (homogeneous and heterogeneous) interval-based data, which is attained by two design decisions. First, the framework is defined at data access layer (DAL), independent from data layer, with maximum flexibility to business logic layer, while both layers are agnostic of the data model. This design makes Di3 and Di4 adaptable to any underlying key-value pair persistence technology, business logic, and presentation layer scenarios (e.g., a graphical user interface). The persistence technologies span built-in in-memory key-value pair collections of most programming languages (e.g., “Dictionary” and “ConcurrentDictionary” in C# and “map” in C++) and classical data structures such as B+tree to cloud-oriented technologies (e.g., LevelDB inspired by Google’s BigTable technology, Apache Accumulo, Apache Cassandra, Apache HBase, Kyoto Cabinet, Berkeley DB, or Symas Lightning Memory-Mapped Database (LMDB) (NoSQL databases are surveyed in [26] and [27])).

Second, the diversity of domains, data types, and semantics of intervals supplemented by application-driven operations, motivates implicit definition of “sense-making” functions, such that, in contrast to **what** functions to be implemented, Di3 and Di4 define **how** such

functions to be applied. In this regard the proposed models accept user-defined functions (UDF) through behavioral design patterns such as *strategy pattern* [28]. In general, Di3 and Di4 models hold the following characteristics:

- Relative and absolute knowledge of the occurrence point on the domain.
- Retrieve relative ordering of intervals for conjoint evaluation (e.g., range queries, or nearest neighbors).
- Persisted index leveraging on a data layer technology.
- Mainly static intervals such that commonly preserve their states and do not necessitate to frequently update occurrence points of indexed intervals.
- Do not map sense of “now”.
- Accepts user-defined functions (UDF) over primitive Di4 operations.

1.4 Structure of the Thesis

The dissertation is organized as follows. Chapter 2, we review state-of-the-art for researches conducted in genomics for efficient retrieval from NGS data. We review classical data structures such as interval tree that can be used for retrieval and organization of genomic intervals. Then we review the state-of-the-art of temporal databases for efficient event storage and retrieval methods that can be applied or inspire genomic interval manipulation.

In Chapter 3 describes Di3. We explain the Di3 data model, and intervals organization on the model, and the kind of operations the model supports and excels. We then describe how intervals are indexed on this data structure, to which we provide single-pass and multi-pass indexing, and the explain the scenarios where one is superior to the other, and we provide pseudo codes for each indexing algorithm. Finally, we explain the functionality of Di3 in three layers of *physical*, *logical*, and *semantic*; and we formally described the information retrieval functions of each layer and provide pseudo code for each.

In Chapter 4, we describe Di4, which provides Di3 functionalities but with a different model. We explain Di4 independent from Di3 in model, algorithms and data structure. We explain Di4 design, model, and operations. We provide pseudo code for it's manipulation and retrieval. Similar to Di3, we explain the functions of Di4 in three layers, and for information retrieval functions in each layer we provide pseudo code and formal description. We explain the differences between Di3 and Di4. In Chapter 5, we assess the performance of Di3 vs. Di4. Then we compare Di4 performance we common genomic tools.

In Chapter 6, we explain Di4 in a comparative analysis setup, which highlights Di4 applicability to different business logic and presentation layer scenarios, in addition to extensive analytical user-defined function. The setup provides means of comparative evaluation of ChIP-seq

Chapter 1. Introduction

enriched regions using replicated experiments. This setup describes two applications: share business logic layer design, and differ in presentation layer (a command line interface (named: MSPC (Multiple Sample Peak Calling)) as opposed to graphical user interface (named: MuSERA (Multiple Sample Enriched Region Assessment))). In addition, MuSERA uses Di4 for common “sense-making” procedures on ChIP-seq replicates such as *(i)* genome browser, *(ii)* functional analysis, *(iii)* nearest neighbor search, and *(iv)* correlation assessment.

Finally, in Chapter 7, we described Google-style search, which is an extensive usage of Di4 for similarity search.

2 Background and State of the Art

Chromatin immunoprecipitation (ChIP) followed by massively parallel DNA sequencing (ChIP-seq) is rapidly emerging as a standard for investigating protein–DNA interactions. Commonly, a comprehensive understanding of how various histone modifications, chromatin structures, transcription factor bindings, and other DNA regulatory elements orchestrate a particular transcriptional profile, necessitate a comparative consideration of multiple experiments. For instance, various chromatin factors cooperate in complex manner to orchestrate transcription, which necessitate robust similarity metric (e.g., insensitive to non-biological variation such as peak width) on ChIP-seq data to quantify chromatin interactions [29]. Additionally, Given the intrinsic noise of the ChIP-seq protocol, it is recommended to repeat every experiment at least twice [30] which produces biological and technical replicates.

The ChIP-seq studies can thoroughly characterize genome-wide location of various protein bindings [31] and histone modifications [32] with high precision. However, in addition to actual location of individual elements, a profound study considers also interaction between the regulatory elements. A common initial step in building genome-wide understanding of such characteristics is to assess the co-occurrence of interactions from a dataset of homogeneous ChIP-seq experiments [33] [31] [34] [29].

Sense making is an exploration task, and its functional requirements span basic region calculus to complicated analysis-specific operations. Therefore, an ideal tool in this category is required to offer quadruple aspects: functionality, extensibility, reusability, and scalability. Present chapter discusses tools offered by bioinformatics community for interval-based genomic data manipulation, and evaluates each for the quadruple aspects. Additionally, manipulation of interval-based data has historical interest in the field of databases and data structures which induced two distinct deviations, respectively *temporal databases* and *spatial data structures*; both aiming at tackling interval-based data manipulation challenge from efficient storage/retrieval and manifold operations point of view. Presents chapter also reviews related studies in these fields.

2.1 State of Art Tools in Genomics for Region Calculus

The sequencing machines encapsulate DNA fragments into files, and pipelines processing fragments export data in files (*samples*) which together motivates genomic data management research to mainly focus on file systems. Bioinformatics community offers a rich collection of ad-hoc applications (e.g., BEDTools [35], BEDOPS [36]) and systematic solutions (e.g., UCSC [37] and Galaxy [38]) to perform prominent IR tasks. These tools mainly leverage on file system; beside the poor practices file system triggers such as duplications (i.e., personal copies) and encoding meta-data in filename, it also unnecessarily escalates the challenges for variety of applications such as rapid retrieval, extendibility, data sharing, and analysis. Present section discusses the challenges in details.

The cornerstone of interval-based data manipulation in genomics is *interval intersection*, “given a query interval, find all reference intervals that overlap it” [39] [40]. Interval intersection is fundamental for retrieval on annotation data and alignment databases, and integration of diverse biological information using a reference genome [41]. Queries to retrieve features overlapping specific regions are frequently used when examining genomic features (e.g., on a genome browser). Such queries might be executed by linearly scanning entire file, that might be reasonable for few retrieval. However, reading the entire file is an inefficient strategy in long run. A trivial solution would be adopting database systems [37] [42]; however, the limited functionality and poor performance practices [41] of generic database system, in addition to the complexity of setting up and designing schemas of the database for an average end user, discourage adoption of such systems. Therefore, a wide range of research is focused on efficient algorithms for interval intersection, and an extensive set of applications implement it. In general, interval intersection algorithms are trinary, described as follows.

- i. **Tree-based**, e.g., BITS [43] (binary tree), and Segtor [44] (segment tree). The UCSC Genome Browser [37], BEDTools [35], and SAMTOOLS [45] uses R-trees [46]; such that intervals of a sample are partitioned into hierarchical “bins”, and query intervals are compared with “bins” to narrow the search for interval intersection to a focused portion of the genome. The algorithm is inefficient, as it necessitates a linear-sweep on all the intervals in each candidate bin for those overlapping the reference. Additionally, such algorithms are poor candidates for parallelism, because non-uniformly distributed intervals (which is common for ChIP-seq, RNA-seq, and exome sequencing) unbalance bin loads; consequently, some bins takes considerably longer time to be processed than others.
- ii. **Plane-sweep-based** on pre-sorted samples, e.g., FJoint [47] which concurrently scans pre-sorted query sample and reference intervals to find overlaps. Some tools use plane-sweep-based algorithms on pre-sorted samples [48] [49] [50]; for instance, recent versions of BEDTools and BEDOPS use plane-sweep as complement to tree data structure. Theoretically, plane-sweep-based algorithms are optimal; except for parallelization considering “sweep invariant” challenge [51]: “given a sweep line and its associated split, all

intersections and event points are known between the sweep line and the beginning of the split.”. Plane-sweep algorithms are parallelized by partitioning the input statically or dynamically, such that each partition can be swept without violating “sweep invariant” [52] [53] ([51] discusses a partitioning method without violating “sweep invariant”). However, parallelization degree of this method is limited to the number of partitions, and execution load of partitions is prone to non-uniformly distributed data.

- iii. **Index-based**, e.g., NC-lists [41]. Efficient retrieval requires a dedicated data structure. For instance indexed flat file such as BAM file [45], or Tabix [54] that adapts BAM indexing techniques for generic tab-delimited files, convert a file of sequential access to its equivalent random access file. Such data structures are commonly built during pre-processing steps, and are persisted for further references. Building such data structures may take a considerably long time; however, as the processing time is significantly reduced, the pre-processing and processing balance is encouraging. The observation leads to the conception of BAM format. A common method is Nested Containment lists (NC-lists) [41] (an extension to NC-lists [55]) which is mainly designed for: “given a query interval, find all reference intervals that overlap it”.

Commonly algorithms and tools offered by bioinformatics community are for pairwise intersection of genomic intervals. Extension to beyond pairwise is commonly provide by custom scripts and algorithms do not offer such extensibility. Hence, custom scripts suffer scalability and orthogonality, which makes it difficult to perform a comparative analysis on a collection of homogeneous sample. Identifying intersecting intervals of multiple samples is an important aspect to study variety of biological characteristics. To address the challenge, “slice-then-sweep” algorithm [56] for N-Way interval intersection is developed.

2.2 Related Work from Classical Data Structures

Classical search trees such as interval trees [57], segment trees [58], range trees [59], or Fenwick trees [60] are optimal solutions each for particular interval-based retrievals (see Table 2.1). For instance, IR queries such as “find all the intervals intersecting given interval” can be determined in $O(\log_2 n)$ using interval trees; while queries such as “find non-overlapping neighbor intervals at d -distance for a given interval”, or “find n -th closest interval” requires re-mastering interval trees. Additionally, some operations despite of introducing challenges may require multiple of such data structures to be utilized concurrently, which might be considered suboptimal. For example, a query such as “given a point/interval determine n -th closest region where a particular number of intervals are accumulated” requires segment and Fenwick trees complementation. Moreover, such collections are mainly designed as in-memory data structures; and when persisted, cause the burden of IO.

Table 2.1: Functionality comparison of classical data structures.

	Feature	Interval Trees	Segment Trees	Range Trees	Fenwick Trees
Performance	Preprocessing	$O(n \log_2 n)$	$O(n \log_2 n)$	$O(n \log_2 n)$	$O(n \log_2 n)$
	Query Time	$O(k + \log_2 n)$	$O(k + \log_2 n)$	$O(k + \log_2 n)$	$O(\log_2 n)$
	Space	$O(\log_2 n)$	$O(n \log_2 n)$	$O(n)$	$O(n)$
	Modification	$O(\log_2 n)$	$O(\log_2 n)$	$O(\log_2 n)$	$O(n)$
Application	Storage	Interval	Interval	Point	Accumulation
	Query	Point/Interval	Point	Interval	Interval
	Output	Interval	Interval	Point	Accumulation

2.3 Related Work from Temporal Databases

Informally, an event describes a phenomena that is happened, is happening, or is expected to happen at certain time and/or at specific location. Events are central elements in the representation of data in various domains spanning multimedia [61], geography [62], and cultural heritage [63].

Vast majority of events can be generalized in terms of intervals on various domains, where their manipulations can reveal extensive aspects of such events. For instance, Flight schedules as events occurring on an airport can be modeled as intervals on temporal domain, which then can be used for various scheduling task such as aircraft landing (e.g., [64], [65]), gates (e.g., [66], [67]), crew (e.g., [68], [69]), baggage (e.g., [70]) or aircraft maintenance (e.g., [71], [71]). Various scheduling problems [72] are also defined in terms of intervals; domain specific scheduling such as parallel machine scheduling (e.g., [73]), sport game scheduling (e.g., [74], [75]), flow-shop scheduling (e.g., [76], [77]), or general scheduling tasks (e.g., [78]).

Despite of scheduling problems, intervals have wide range of applications. Different versions of elements[79] of spatial data each valid during a specific timespan are defined in terms of intervals. Intervals are used in estimating rounding error of calculations in numerical analysis (e.g., [80], [81]). Some computer-assisted proofs manipulates intervals for problems such as proving Kepler [82] and double bubble [83] conjectures, or chaos in the Lorenz equations (e.g., [84]). Global optimization is another domain of interval application (e.g., Rastering function [85, 86]).

In this dissertation, events are considered as real-world occurrences that unfold over any domain, and is focused on events that have happened. In general, according to a classification proposed in [87], an event could be:

- **structure operation**, an event is triggered by an operation on a data structure. For instance, insert an item into a table.
- **behavior invocation**, an event is triggered by execution of a user-defined function. For instance, a message *display* is sent to an object of type *widget*.
- **transaction**, an event is triggered by executing transaction commands. For instance, abort, commit.

- **abstract** or **user-defined**, an event is triggered by a programming mechanism that signals occurrence of an event explicitly. For instance, response to information entered by a user.
- **exception**, an event is triggered by an exception occurrence. For instance, incorrect input.
- **clock**, an event is triggered at specific point in time. For instance, Feb 16, 2016 15:05PM.
- **external**, an event is triggered by an action outer to the system. For instance, fire alarm is triggered.

Additionally, [87] generalizes events in two types:

- **primitive**: an event is triggered by an occurrence described in pre-defined categories. For instance, insert an item into a table.
- **composite**: an event is triggered by a primitive or a composite occurrences, where the composition rule is defined an algebraic expression. Rich event algebras have been proposed for a range of systems including [88] and [89].

A standard spatiotemporal database uses the basic model that all data objects have a locational attribute and a temporal attribute that record the spatial and the valid time extent associated with the object. Many spatiotemporal information systems develop novel access methods for spatiotemporal queries. In some cases like Geographic Information Systems, the primary data object can be a spatial location which has other non-spatial and temporal properties associated with it. An event-oriented spatiotemporal database is a variant, which describes spatiotemporal data objects with “events” as a data organizing attribute. In the event-based approach introduced by [90] for efficiency reasons, an event is interpreted as a change in data property. Such systems stores the time associated with each change in increasing order from the initial “world state” at time t_0 to the latest recorded change at time t_n . [62] have developed an object-based event model which is designed to answer queries like the following:

- What are all the events related to object X ?
- What are the objects that are related to event Y ?
- Has any instance of event type Y happened without the participation of object X ?
- What are all the events that are related to event Y ?

More specialized data models and access methods have been developed for specific event related problems in spatiotemporal databases. In addition, considering the event modeling in sensor networks (e.g., [91], [92], [93]), and multimedia information systems (a survey: [94]), there exist a wide diversity in event modeling research. Present section considers modeling events in temporal databases.

In general, three interpretation of *time* is considered in temporal databased: (i) *valid time* during which a fact is true in the environment, (ii) *transaction time* during which a fact is stored in the databased, and (iii) *bitemporal* where both valid and transaction time are considered. Given that the present dissertation is focused on genomics as application scenario, only temporal databases with *valid time* interpretation of *time* are considered in this section.

A classical organization of events in temporal databases fall in four categories as follows: [95]

- i. **State-space approaches** [96] [97] where the state is an instantaneous point in the domain, and actions are mapping between states.
- ii. **Date line systems** [98] [99] [100] events are organized by dates while their temporal order is preserved, this method is mainly used for time points rather than time intervals.
- iii. **Before/after chaining** [98]; motivated by relative temporal information. Although efficient for relative ordering queries, but has intrinsic drawbacks of linear access for n -th item.
- iv. **Formal models** [101] which are essentially point-based representations, where intervals are constructed out of consecutive points; with points forming the foundation of reasoning system.

A temporal database model may incorporate multiple of concepts in one model. For instance, event model of Bertino et al., [102] inspired *state occurrence* and *state transition* as views on temporal data. Additionally, recent temporal data models introduce novel cloud-based and modern hardware architecture based technologies (e.g., [103] [104], or [105] a recent book on temporal data models and indexing). For instance, the *Simple Event Model* (SEM) [106] which is created to model events in various domains, without making assumptions about the domain-specific vocabularies. SEM is designed with a minimum of semantic commitment to guarantee maximal interoperability. In addition, Checkpoint INTerval Index Array (CINTIA) [107] is a recent indexing framework on temporal data. CINTIA is a data structure to store and query on interval data, with main objective of high memory locality and query execution performance on big collection of intervals. Regardless of the taxonomy, temporal databases commonly define five conceptual operations on events described as follows:

- i. `GetStart(event)` and `GetStop(event)` operations that respectively determine start and end of the event.
- ii. `Lifespan(event)` operation that returns an interval of time during which the event is valid.
- iii. `Snapshot-state(event, time)` operation that returns the *state* of the event at time, while *state* could be *true* or *false* indicating whether the event is valid at time or not. A different design of this function returns a record of all attributes of the event at time, if the event is valid, and returns null if the event is not valid at time.

- iv. `Snapshot-value(event, attribute, time)` operation returns *value* of attribute of event at time.
- v. `Historical-state(event, StartTime, EndTime)` operation returns a ordered (based on time) and timestamped record of all the attributes of event on given time interval.

Interval intersection query, as described in Section 2.1, is an active challenge in temporal databases. An enormous amount of study is focused on optimal interval intersection queries (e.g., [108] [109] [110] [111] [112] [113]), four technical goals that includes [41]:

- i. **Database scalability**; for n interval in database, a method should ideally require $O(\log n)$ for interval intersection, with memory cost of $O(\log n)$, while storage cost is $O(n)$.
- ii. **Query scalability**; for n intervals overlapping a query interval, a method should ideally require $O(n)$ to determine n intervals, while being linear ($O(n)$) or constant ($O(1)$, depending on the scenario) in memory requirement.
- iii. **Construction scalability**; time complexity for creating an indexed database that supports interval intersection should be of the same scale of an standard database and does not exceed $O(n \log n)$ for n intervals, and memory complexity be bound to $O(n)$.
- iv. **Update scalability**; a database should support and be agile on updates, such that an update on n items should not exceed $O(n \log n)$ in time and memory complexity.
- v. **Practicality**; such a database is required to be executable on a typical platform; data size, execution environment and runtime should be of reasonable balance.

However, although classical data structures such as R-tree [46], segment tree [58], MV3R-Tree [114], and Overlapping B+trees [115] are widely appreciated in the bioinformatics community, but achievements of temporal database realm got relatively less attention by the community.

3 Di3: 1D Intervals Inverted Index

Present chapter explains *1D intervals inverted index* (Di3), a general-purpose index structure which provides fast access to intervals; therefore, it applies to several domains, including genomics (any genomic region is a linear interval defined by the genomic coordinates of the region ends). Di3 main strengths are its ability to adapt to domain needs, thanks to the native support for user-defined functions (UDFs), and its portability to several implementation technologies, thanks to its high-level, layered design that abstracts from implementation details. Di3 models homogeneous and heterogeneous intervals on a domain; related events are collected in sets, which collectively constitute a *sample*. For instance, a genomic data sample may contain regions (i.e., intervals) of DNA-protein interactions occurring on a genome under an experimental setup; each interval can be associated with values, e.g., a significance score.

Present chapter explains Di3 data structure at higher-level. It is mainly focused on describing the indexing model, and how the elements of the model (*snapshots*) are used for bookkeeping intervals. Intentionally present chapter keeps formalism at lower-level to focus on higher-level concepts of indexing model. Next chapter, Chapter 4, explains the Di3 model and its elements in details, discusses possible challenges that may raise under particular scenarios, and provides 1D intervals incremental inverted index (Di4) that tackles the challenges of Di3 under the scenarios.

Accordingly, the chapter is organized as follows: Section 3.1 explains the model of Di3 and its elements, Section 3.2 explains operations defined on Di3 model, Section 3.3 explains how intervals are organized in Di3 model, and finally Section 3.4 provides pseudocode for operations describing how Di3 model is used for each function.

3.1 Di3 Design

Genome exploration studies are grounded on the efficient execution of operations for the composition and comparison of (epi)genomic regions, and their associated attributes. Region-based operations to be performed towards these goals include the identification of co-occurrences

or accumulations of regions, possibly from different biological tests and/or of distinct semantic types, within the same area of the DNA, sometimes within a certain distance from each others and/or from DNA regions with known structural or functional properties (e.g., describing particular DNA sequence motifs, genes involved in certain biochemical pathways, or regulatory regions of gene transcription activity). Nowadays, such complex operations are only partially supported by existing tools, e.g., BEDTools [35], BEDOPS [36], GROK [116]; these tools typically support only algebraic operations based on the genomic coordinates of the regions within a single data sample or a pair of samples at the time, requiring the use of scripts to perform complex operations on multiple data samples.

To cope efficiently with complex region calculus, we have developed the Di3, a multi-resolution single-dimension data structure. Di3 is defined at the data access layer, and it is independent from data layer, business logic layer, and presentation layer. This design decision has two significant advantages; firstly, being independent from the data layer, Di3 is adaptable to any key-value pair persistence technology. These may range from Apache Cassandra, LevelDB, Kyoto Cabinet, and Berkeley DB for persisted large scale data (NoSQL databases are surveyed in [26] and [27]), to simple in-memory key-value collections implemented by most of modern programming languages (e.g., “Dictionary” in C# and “map” in C++). Secondly, Di3 design can support different business logic and presentation layer scenarios, which are complemented by user-defined functions (UDF) provided via behavioral design patterns such as *strategy pattern* [28].

In general, let $S = \{S_1, \dots, S_j, \dots, S_J\}$ denote the available samples, where each sample is a set of intervals $S_j = \{I_1^j, \dots, I_i^j, \dots, I_{|S_j|}^j\}$; each interval $I = [l, \bar{l}]$ is included within its lower (left) and upper (right) bounds (ends). Di3 organizes intervals by means of *snapshots* (see Figure 3.1 panel A); ¹ each snapshot corresponds to a point on the domain, and it is associated with all the intervals overlapping that point. More precisely, each snapshot B_b is a key-value pair element, where the key (e_b) is the coordinate of the snapshot on the domain, and the value (λ_b) is a set of pointers to the descriptive metadata of all the intervals overlapping the e_b coordinate. For instance, two “Flight” events can be modeled using four snapshots as follows:

- i. 9:00AM: Flight-A departures.
- ii. 9:30AM: Flight-B departures, and Flight-A is flying, and was flying all the time between 9:00AM and 9:30AM.
- iii. 10:00AM: Flight-A lands, and Flight-B is flying, and all the time between 9:30AM and 10:00AM both flights were flying.
- iv. 10:30AM: Flight-B lands, and all the time between 10:00AM and 10:30AM Flight-B was flying.

¹The term *snapshot* is inspired by *snapshot databases* [117] where queries reconstruct database state at a given time in past; in other words, in snapshot databases the focus is mainly on the queries that ask for the state of a database at a given time.

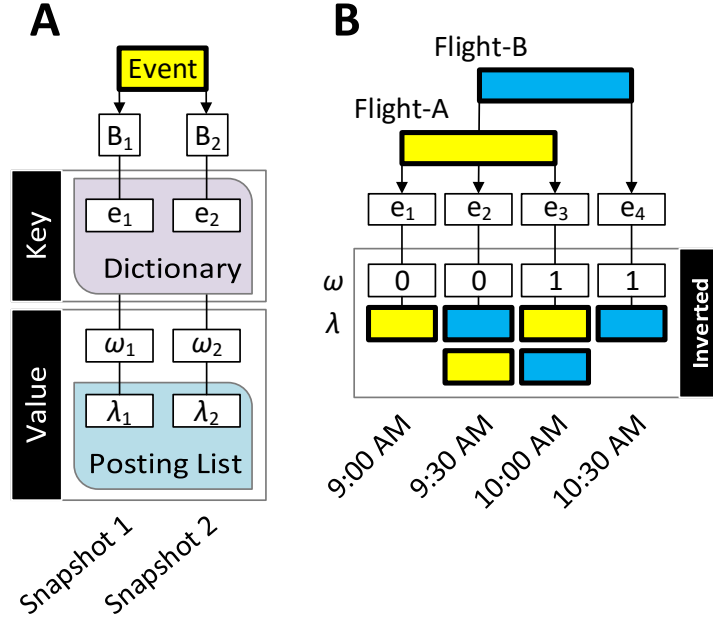


Figure 3.1: Di3 data model. (A) Depicts the key-value pair design decision, and dictionary entries (i.e., positions on the domain) and posting list (i.e., observed events on that position on the domain). (B) Illustrates the “Flights” example and the four snapshots Di3 creates for the two flights.

In general, a snapshot on a domain has two essential attributes; first, a snapshot has at least one causal event. For instance, the causal event of the snapshot at 9:30AM is the *departure of Flight-B*. Second, a snapshot captures the state of events happening at a position on domain. For instance, the snapshot of 9:30AM captures both *departure of Flight-B* (the causal event) and *Flight-A is flying*, where both events are at a degree of happening. According to these attributes, the organization of events by snapshots in key-value pair paradigm is described as follows for *key* and *value* individually.

3.1.1 Snapshots keys organization

The procedure of assigning snapshots to each end of intervals is described step-by-step in present subsection with reference to Figure 3.2 and highlighting main attributes of snapshot key organization of the Di3 model as follows:

- **One snapshot for each end of an interval** (see panel A on Figure 3.2)

An interval I_i^j is indexed using exactly two snapshots, one for the left-end (I_i^j) and one for the right-end (\bar{I}_i^j). For instance, referring to panel A on Figure 3.2, interval I_1^1 is indexed by two snapshots B_1 and B_2 . Each snapshot refers to a position on domain by its key. For instance, snapshots B_1 and B_2 on panel A of Figure 3.2 refer to positions $e_1 = I_1^1$ and $e_2 = \bar{I}_1^1$ on the genome.

- **Snapshots are sorted** (see panel B on Figure 3.2)

Snapshots are organized sorted based on snapshot key, and any new snapshot that is inserted in Di3 maintains the order. For instance, referring to panel B on Figure 3.2, interval I_1^2 is indexed by two snapshots B_1 and B_2 where $e_1 = \underline{I}_1^2$ and $e_2 = \bar{I}_1^2$. Since, \underline{I}_1^2 and \bar{I}_1^2 occur earlier than \underline{I}_1^1 and \bar{I}_1^1 respectively, the snapshots are ordered accordingly.

- **One snapshot for multiple causal intervals** (see panel C on Figure 3.2)

A snapshot refers to at least one interval, and multiple intervals with one equal end can be indexed using single snapshot. In other words, there exist at least one causal interval for each snapshot. For instance, referring to panel C on Figure 3.2, a new interval I_1^3 is inserted. A snapshot B_3 is assigned to its left-end (i.e., $e_3 = \underline{I}_1^3$). However, the right-end of I_1^3 equals the right-end of previously organized interval I_1^2 (i.e., $\bar{I}_1^2 = \bar{I}_1^3$), therefore, the snapshot that was assigned to right-end of I_1^2 can be used to index the right-end ($e_4 = \bar{I}_1^2 = \bar{I}_1^3$).

- **Gap** (see panel D on Figure 3.2)

Two consecutive snapshots are not necessarily indexing contiguous intersection. In other words, intervals indexed by two consecutive snapshots are not necessarily overlapping. For instance, referring to panel D on Figure 3.2, two consecutive snapshots B_5 and B_6 are indexing I_1^1 and I_2^3 respectively, while the two intervals are not overlapping. Therefore, there is a *gap* between two consecutive snapshots if none of the indexed intervals overlap.

- **Adjacent bounds** (see panel D on Figure 3.2)

Two adjacent snapshots are indexing two consecutive causals bounds. For instance, referring to panel B on Figure 3.1, the snapshots at 9:00AM and 9:30AM are indexing two consecutive flight departures (i.e., “Flight-A” and “Flight-B”). Therefore, the lower and upper bound of an interval are not necessarily indexed by two adjacent snapshots. The two bounds of an interval are indexed by two adjacent snapshots if are the two consecutive causals bounds. For instance, referring to Figure 3.2, the lower and upper bound of intervals I_1^3 and I_2^3 are indexed by two adjacent snapshots B_3 - B_4 and B_6 - B_7 respectively. In contrast, the two bounds of interval I_1^1 are indexed by non-adjacent snapshots B_2 - B_5 , because the two bounds are not consecutive causalities.

3.1.2 Snapshots *values* organization

While the *key* of each snapshot informs the position on domain where an interval starts or stops, the value of snapshots reconstructs the intervals overlapping the position. The procedure of populating snapshots values by each interval insertion is described step-by-step and highlighting main attributes of snapshot *values* is described as follows:

- **Indexing an interval by a snapshot** (see panel A on Figure 3.3)

Each snapshot stores a tuple per each interval it overlaps with. This tuple describes for

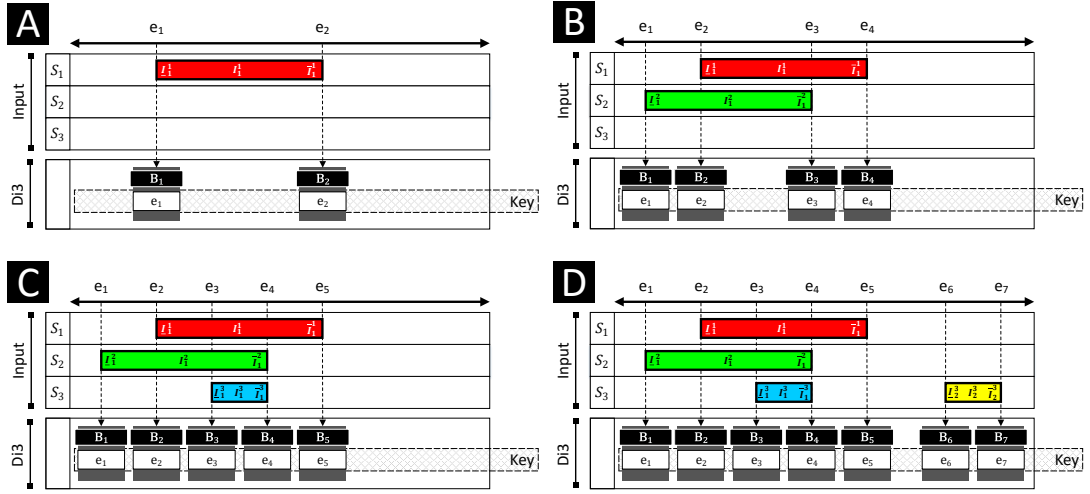


Figure 3.2: Di3 key organization explained step-by-step. S_j : sample j ; I_i : interval i ; B_b : snapshot b .

each interval the intersection condition, and it has a pointer to the interval descriptive metadata. For instance, referring to panel A on Figure 3.3, the snapshots B_1 and B_2 have one tuple for interval I_1^1 that overlaps with them. The tuples are $(L, @I_1^1)$ and $(R, @I_1^1)$ as values of snapshots B_1 and B_2 respectively. Letters L and R describe the intersection type, such that L tells that the left-end of the interval is overlapping the snapshot, and R tells that right-end of the interval is overlapping the snapshot.

- **Overlapping intervals** (see panel B on Figure 3.3)

If two interval overlap, then there exist at least one snapshot that has intersection condition M . For instance, referring to panel B on Figure 3.3, the newly inserted interval I_1^2 overlaps previously inserted interval I_1^1 . The left-end of interval I_1^1 overlaps I_1^2 , and right-end of interval I_1^2 overlaps I_1^1 , therefore, the related snapshots describe this condition by adding a tuple per each interval with the intersection condition M . In this case, tuple $(M, @I_1^2)$ is added to B_2 and tuple $(M, @I_1^1)$ is added to B_3 .

- **A snapshot has at least one tuple with intersection condition L or R** (see panel C on Figure 3.3)

A snapshot is assigned per each end of intervals, therefore, the value of a snapshot has a tuple to at least one causal interval. In other words, there exist at least one tuple with intersection condition L or R , and any number of tuples with intersection condition M . For instance, referring to panel C on Figure 3.3, the newly inserted interval I_1^3 has the same right-end as the previously inserted interval I_1^2 , hence the snapshot B_4 refers to both causal intervals by having $(R, @I_1^2)$ and $(R, @I_1^3)$.

- **Gap** (see panel D on Figure 3.3)

A gap between two consecutive snapshots is determined by single snapshot value. If all the tuples of a snapshot has intersection condition R , then there is a gap between the

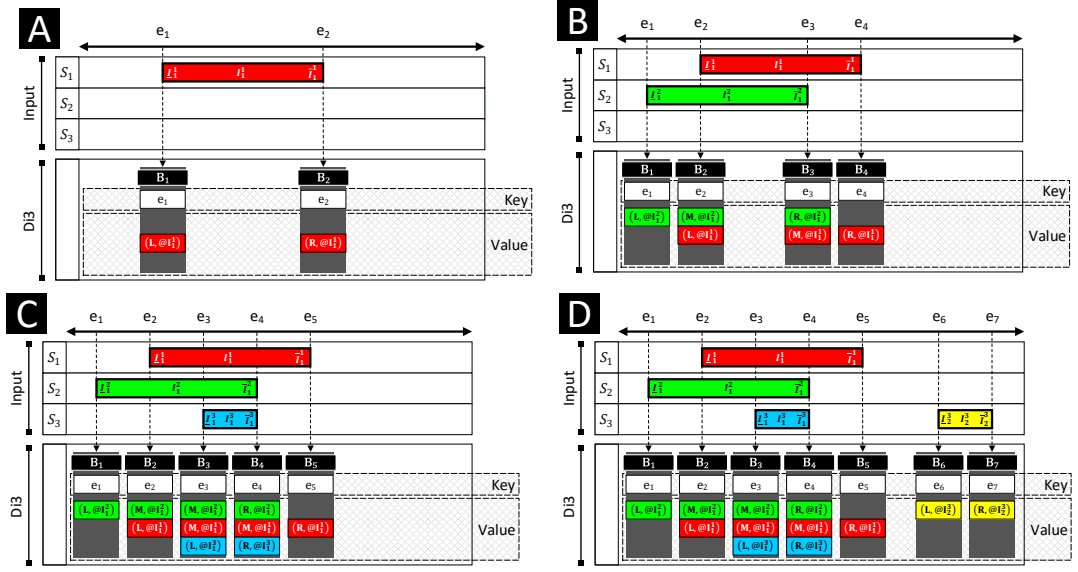


Figure 3.3: Di3 value organization explained step-by-step. S_j : sample j ; I_i : interval i ; B_b : snapshot b .

snapshot and its preceding snapshot. For instance, referring to panel D on Figure 3.3, the snapshot B_5 has one tuple and its intersection condition is R , therefore, there is a gap between B_5 and B_6 . Alternatively, if all the tuples of a snapshot have intersection condition L , there is a gap between the snapshot and its former snapshot. For instance, referring to panel D on Figure 3.3, the only tuple of snapshot B_6 has intersection condition L , hence, there exists a gap between B_6 and B_5 .

The number of intervals overlapping a snapshot presents the *accumulation* of intervals at that snapshot. For instance, referring to panel D on Figure 3.3, accumulation at B_1 is 1, because the snapshot refers to one interval I_1^1 ; accumulation at B_2 is 2, because the snapshot refers to two intervals I_1^1 and I_1^2 ; and accumulation at snapshot B_3 is 3, because the snapshot refers to three intervals I_1^1 , I_1^2 , and I_1^3 . To calculate accumulation, it is required to count the number of all the intervals overlapping the position on domain to which the snapshot refers. These are the intervals to which the snapshot has a pointer. However, the intervals are right-open which means accumulation at B_7 (panel D on Figure 3.3) is zero. Because B_7 is referring to the right-end of interval I_2^3 and right-end is exclusive at the position on domain to which the snapshot refers to. Therefore, when counting the set of pointers in a snapshot for accumulation, only the pointers with intersection condition L and M are considered.

3.2 Operations supported by Di3

Present section discusses data retrieval based on coordinate-oriented model of Di3 model. The Di3 design provides a hierarchy of functions to facilitate retrieval and extensibility. Therefore, data retrieval is defined in three levels: *Physical*, *Logical*, and *Semantic*. The former bridges the

Di3 data model to the data layer, using some key-value pair persistence technology. Operations at the physical layer include *Create*, *Read*, *Update*, *Deleted* (CRUD), and *Enumerate*. These operations create and manipulate the snapshots and organize them in a key-value pair storage, by translating input intervals to snapshots and retrieving intervals from snapshots. They are internal to Di3 and accordingly do not incorporate UDFs.

Logical level functions leverage on physical level operations, and they provide the essential elements for region calculus. These functions cover classical region calculus that benefits from the information of a single snapshot, e.g., “given a point on the domain, find intervals overlapping with it” (similar to queries on segment trees [58]), or that leverage on information provided by a set of consecutive snapshots, e.g., “given an interval, find all intervals overlapping with it” (similar to queries on interval trees [57]). Logical level functions leverage on snapshots to optimally retrieve co-occurrences of intervals, or co-occurrence histograms and distributions; some of them define the Di3 public application programming interface (API), whereas other functions can be user-defined functions (UDF).

Upon physical level operations and logical level functions, Di3 builds semantic level functions. The goal of these functions is to facilitate both high-level reasoning on data that include coordinate-attribute criteria, and UDFs creation for extensibility to application requirements. These functions are: *similarity search* (see Chapter 7), which finds samples that best match the criteria defined in a query; *co-occurrence patterns*, which searches for density-based co-occurrence patterns; *dependency detection*, which determines the positions on the domain where query regions co-occur; and *deviation detection*, which finds positions on the domain where the regions of a given set do not commonly co-occur, based on the information stored in the Di3 model.

The dissertation is focused on the logical layer, being this the *de-facto* Di3 API, whereas the physical layer operations are strictly related to the specific persistence technology used for the Di3 implementation, and the semantic layer functions are application dependent (some of them are used by MuSERA and are described in Chapter 6). Concerning the logical layer, the following are the operations natively supported by Di3:

- **Cover**; the COVER function applies to snapshots and reconstruct a single sample from intervals constituting the snapshots by taking into account interval intersections and a UDF. Each resulting interval I is the contiguous intersection of at least `minAcc` (minimum accumulation) and at most `maxAcc` (maximum accumulation) of intervals. See Figure 3.4 as an example of the COVER function. For each resulting interval I , the COVER function determines contributing intervals, which then are passed to the UDF for further evaluation. A UDF may assign to a resulting region I any user-defined value type, spanning from “list of all contributing intervals” or “cardinality of contributing intervals”, to analytical evaluations such as “custom aggregation of significance values”. In COVER, the use of UDFs is thus supported, being `count` the default aggregation function.

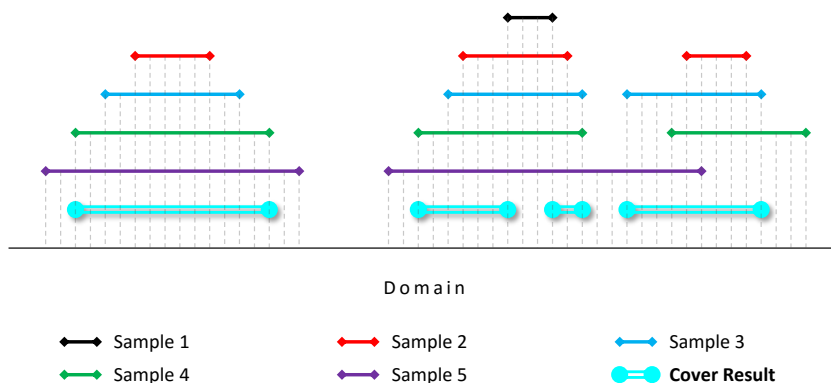


Figure 3.4: An example of *COVER* function with $\text{minAcc}=2$ and $\text{maxAcc}=4$ on 5 samples.

- **Summit**; the SUMMIT function is a variation of the COVER function; similarly, it takes minAcc , maxAcc and a UDF, and reports the local intersections of COVER. In other words, SUMMIT returns regions that start from a position where the number of intersecting regions is between minAcc and maxAcc and does not increase afterwards, and stop at a position where either the number of intersecting regions decreases, or it violates the maxAcc parameter (see Figure 3.5). Similar to COVER, SUMMIT determines contributing intervals and pass them to the UDF, which can aggregate any property of intervals and return any user-defined aggregated value type.
- **Map**; given a reference interval I , the MAP function determines all the intervals indexed in Di3 that overlap with I (similar to classical interval-tree operation). In addition to a reference, the function takes a UDF and passes the determined intervals to the UDF (see Figure 3.6). The output of the UDF is then reported back as an attribute of I . For instance, a UDF may take all the intervals overlapping I and return their cardinality.
- **Accumulation histogram/distribution**; the functions ACCHIS and ACCDIS compute the genome-wide accumulation of intervals, and respectively report a histogram or distribution of the calculated information (see Figure 3.7).
- **Nearest neighbor**; given a reference interval I , the *Nearest Neighbor* function determines the nearest neighbor indexed interval, that is either an overlapping interval, or the closest up-stream or down-stream non-overlapping interval.

3.3 Di3 Intervals

The design of Di3 is motivated by giving the best possible support to logic operations; in particular, Di3 supports MAP and Nearest neighbour operations in logarithmic time in the number of regions, while it supports COVER, SUMMIT and Accumulation histogram opera-

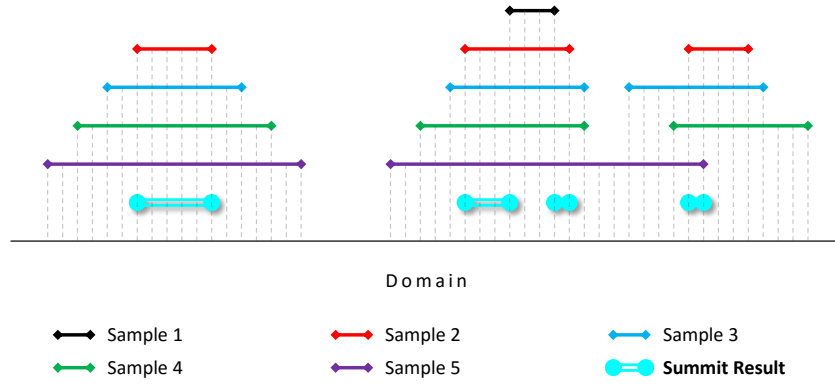


Figure 3.5: An example of SUMMIT function with $\min Acc=2$ and $\max Acc=4$ on 5 samples.

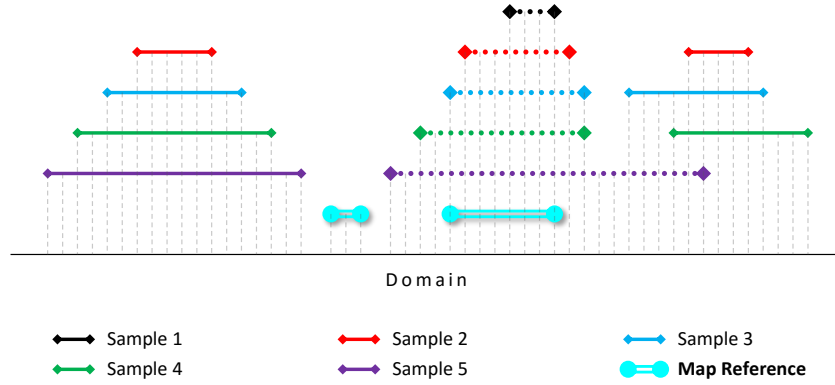


Figure 3.6: An example of MAP function on 5 samples. The dotted intervals are the intervals MAP function finds overlapping reference intervals.

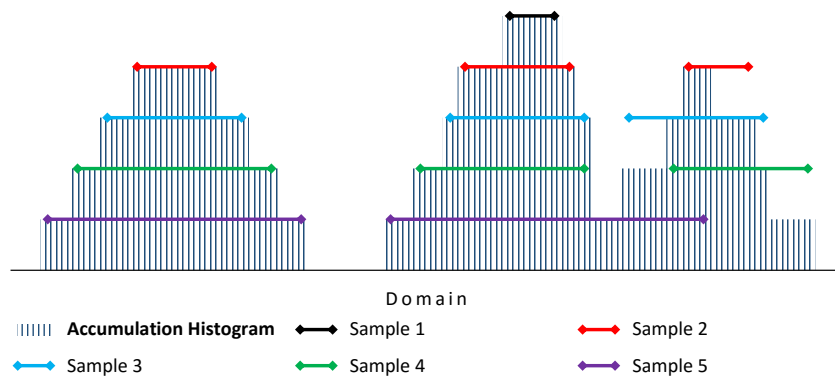


Figure 3.7: An example of Accumulation Histogram function on 5 samples.

tions in linear time (indeed, second resolution indexing supports sub-linear time search, as it will be later explained).

As have seen, Di3 adopts a *single-dimension* paradigm, and targets the interval's coordinates; however, it also adopts a *double-resolution* paradigm. The *first resolution* organizes intervals based on their coordinates through snapshots and provides logarithmic access on the coordinates; the *second resolution* builds groups of snapshots based on their coordinates and computes suitable aggregation functions for each group, using all the attributes (and not just the coordinates). These groups act as secondary key for improved access based on specific attributes. In the following sections we describe each resolution in details.

3.3.1 First Resolution

Snapshots summarize information about the coordinates where a *variation* occurs in intervals, i.e., when an interval starts or ends; a snapshot at e_b exists *iff* at least one interval introduced such variation. Hence, a finite set of snapshots can be used to index a finite set of intervals on both discrete and contiguous domains. Note that multiple intervals with the same starts or ends are possible.

Each snapshot holds a list of the IDs of all the intervals overlapping with its position; having a pointer to each interval overlapping a point on the domain is advantageous mainly for queries that target specific or relative positions. For instance, “given an interval, find all intervals overlapping with it” requires $O(\log_2 n)$ to find the snapshot which has a pointer to all intersecting intervals; or “given an interval, find all its neighbors at 200 base-pair (unit of the genomic domain) distance” requires $O(\log_2 n)$ to find the snapshot at overlapping position, plus few additional siblings of the determined snapshot, and then to join the intervals represented by the snapshots (which asymptotically is still $O(\log_2 n)$).²

This organization is shown in Figure 3.8; the upper part of the figure describes 4 input intervals; the lower part of the figure describes the Di3 index, where each snapshot B_b includes a key-value pair, the key is e_i , and the value is the list of intervals which have an intersection with B_b . Precisely, B_b can be at the left (L), at the right (R), or in the middle (M) of an interval; the type of positioning (i.e., L , R or M) is included in each entry of the interval list, which also contains a pointer to descriptive metadata for the interval. For instance, the third snapshot, B_3 , intersects with three intervals, being in the middle of the S_1 and S_2 intervals and at the left of the S_3 interval.

The first resolution index is created using *batch indexing*, a method which includes input intervals one after the other; such method is preferred over *bulk indexing* or *range indexing*, which require the sorting of intervals prior to index creation. These methods are related to the physical layer being implemented through a B+tree and they are related to classical methods

² Computational complexities are based on a B+tree data structure for the implementation of the physical layer operations.

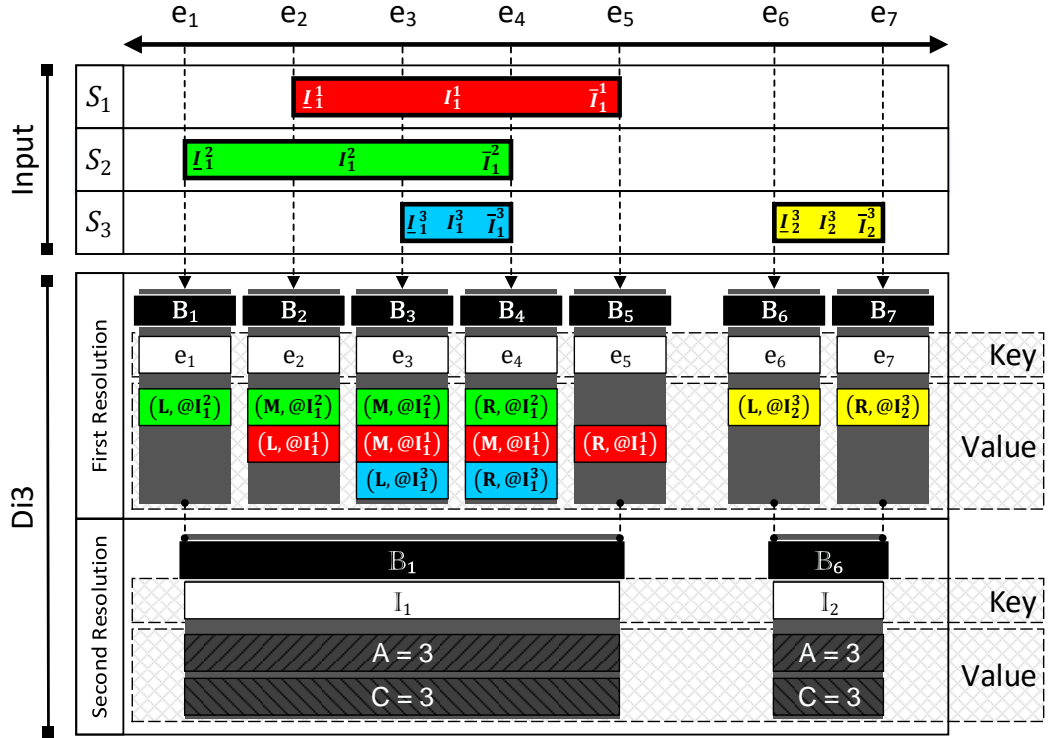


Figure 3.8: The first and second resolutions of Di3. The blocks of second resolution are composed of two aggregations: maximum accumulation (A), and interval count (C).

for B+tree data insertion [118], [119]. To perform batch indexing, we contrasted two methods, respectively called *single pass indexing* and *double pass indexing*. The former one considers each interval in input and updates the data structure in a single pass; the latter one at the first pass orders the snapshots of new intervals with respect to all existing intervals, and at the second pass updates all the snapshots with the information about the list of the intersecting intervals (see Figure 3.9 as an example).

The two methods are compared in Figure 3.10, the comparison clearly shows that no method dominates over the other one. In general, single pass indexing is superior with a small number of new intervals, while double-pass indexing is superior for a large number of new intervals. Based on such analysis, the initial loading of the index in Di3 is performed by using the double-pass indexing, while index update is performed by single pass indexing.

3.3.2 Second Resolution

The second resolution is built from the first resolution as of Figure 3.8, by grouping a set of snapshots and aggregating the relative information. The elements of the second resolution of Di3 are collections of consecutive snapshots grouped together, called *blocks*. Each block

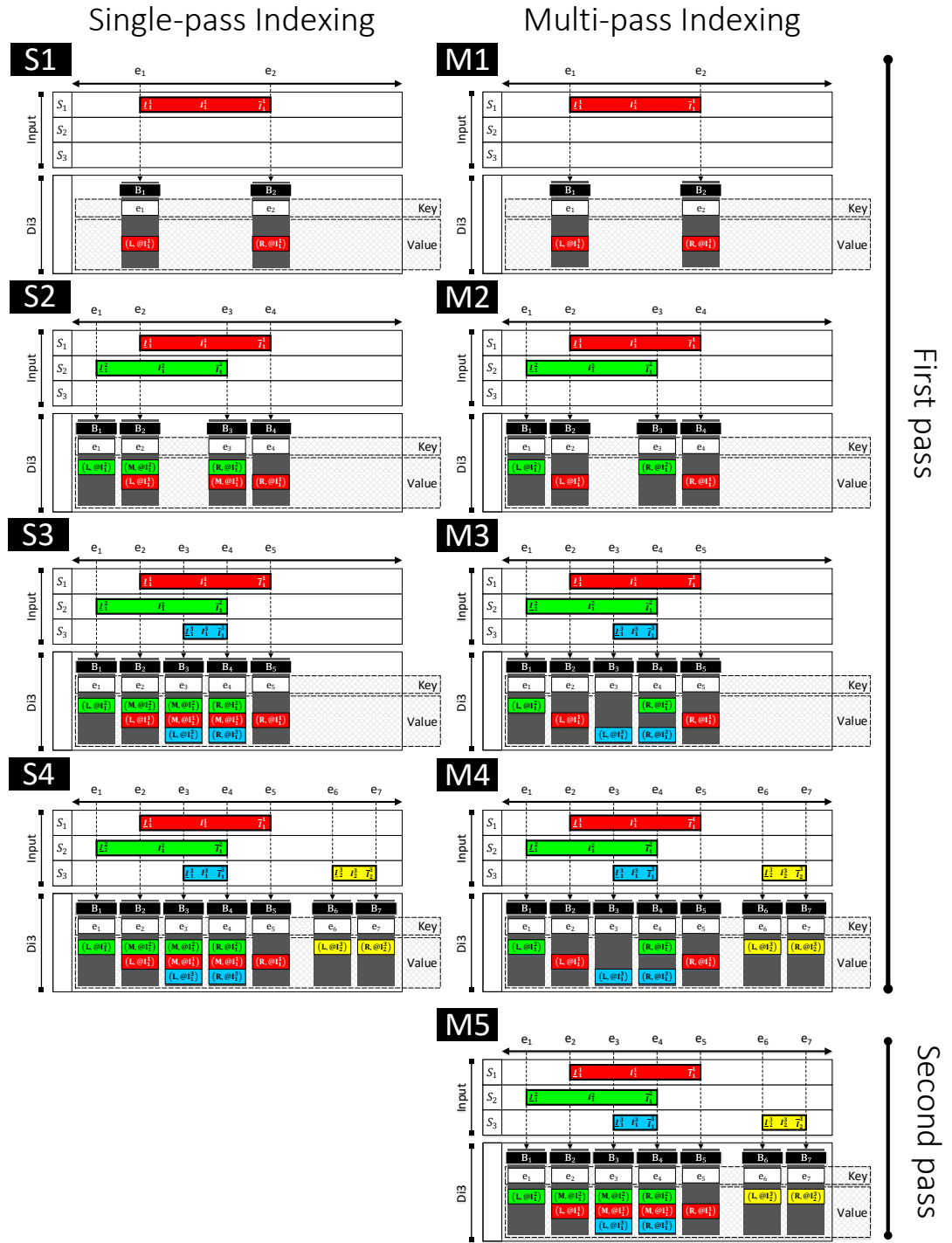


Figure 3.9: An illustration of single-pass compared to double-pass indexing.

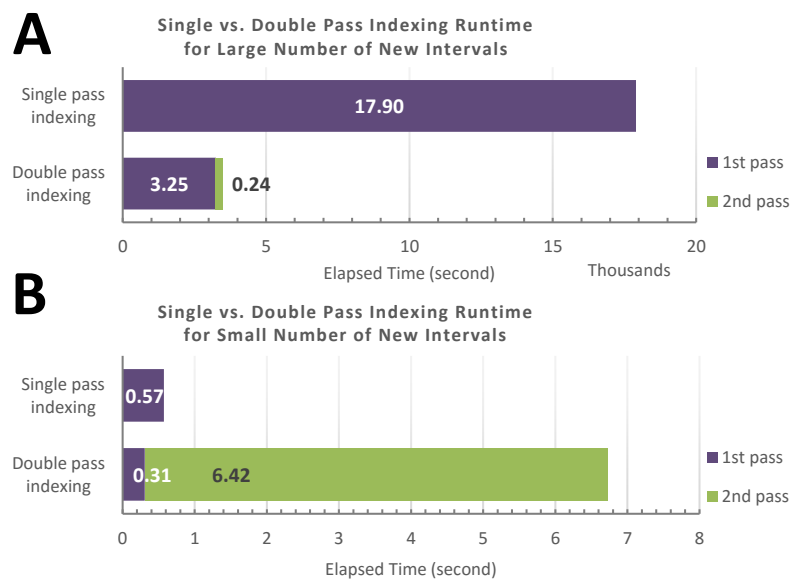


Figure 3.10: A comparison between single-pass and double-pass indexing in two scenarios: (A) large number vs. (B) small number of new intervals.

is defined as a key-value pair; the *key* is defined by a *grouping* function, while the *value* is defined by an *aggregation* function. More precisely:

In **Grouping**, a user-defined function groups a set of adjacent snapshots; depending on the semantic of the UDE, such groups reveal various aspects. For instance, in the genomic domain with intervals representing a biological activity, a group of snapshots is a region on a genome where at least one biological activity is observed. The grouping function defines the block *key*, as an interval on first resolution with start and stop being respectively the minimum and maximum coordinates of the snapshots in the block.

With **Aggregation**, the information of the snapshots within each block are then aggregated, thereby providing summary statistics on the specific attribute(s). Storing custom aggregations for each block reduces snapshot access demands when a particular aggregation is commonly used. The aggregation function defines the block *value*, which is generic in type and size. For instance, count of intervals represented by grouped snapshots, or standard deviation and mean of interval sizes.

Di3 already provides built-in functions for default grouping and aggregation; the default grouping function groups consecutive snapshots that point to consecutive overlapping intervals, and the built-in aggregation function stores the maximum accumulation of each group. These functions improve operations such as COVER, SUMMIT and Accumulation histogram, which depend on accumulation indexes. For instance, in a COVER query, when the max accumulation of a block is lower than the min accumulation `accMin` of the query, or the min accumulation of

a block is higher than the max accumulation accMax of the query, the entire group of intervals does not have an interval that satisfies the query and can be fully skipped. Moreover, disjoint groups can be processed in parallel with no need for synchronization mechanisms.

3.4 Retrievals

Present section discusses the algorithms of Di3 functions, and provides pseudo code for each.

Cover, the function COVER iterates over snapshots of first resolution and determines sets of intervals whose accumulation is between user-defined thresholds minAcc and maxAcc . The determined intervals are then passed to UDF for further application-specific operations. The Algorithm 1 provides pseudo code of this function.

Algorithm 1 Cover Algorithm

```
1: procedure COVER( $\text{minAcc}$ ,  $\text{maxAcc}$ , UDF)
2:    $\text{determined\_intervals} \leftarrow \emptyset$ 
3:    $\text{accumulation} \leftarrow 0$ 
4:    $B_{\text{marked}} \leftarrow \text{null}$ 
5:   for each snapshot  $B$  in first resolution of Di3 do
6:      $\text{accumulation} \leftarrow \text{accumulation at } B$ 
7:     if  $\text{minAcc} \leq \text{accumulation} \leq \text{maxAcc}$  then
8:        $B_{\text{marked}} \leftarrow B$ 
9:        $\text{determined\_intervals} \leftarrow \text{determined\_intervals} \cup \text{intervals of } B$ 
10:    else
11:      if  $B_{\text{marked}} \neq \text{null}$  AND ( $\text{accumulation} > \text{maxAcc}$  OR  $\text{accumulation} < \text{minAcc}$ ) then
12:         $\text{determined\_intervals} \leftarrow \text{determined\_intervals} \cup \text{intervals of } B$ 
13:        UDF( $B_{\text{marked}}$ ,  $B$ ,  $\text{determined\_intervals}$ )
14:         $B_{\text{marked}} \leftarrow \text{null}$ 
15:       $\text{determined\_intervals} \leftarrow \emptyset$ 
```

Summit, the function SUMMIT is a variant of COVER, similarly iterates over snapshots of first resolution and determines sets of intervals whose accumulation is between user-defined thresholds minAcc and maxAcc , and is local maximum. Similar to COVER, the retrieved intervals are then passed to UDF for further application-specific operations. The Algorithm 3 provides pseudo code of this function.

Algorithm 2 Map Algorithm

```
1: procedure MAP( $I_{\text{reference}}$ , UDF)
2:    $\text{determined\_intervals} \leftarrow \emptyset$ 
3:   for each snapshot  $B$  in first resolution of Di3 between  $I_{\text{reference}}$  and  $\bar{I}_{\text{reference}}$  do
4:      $\text{determined\_intervals} \leftarrow \text{determined\_intervals} \cup \text{intervals of } B$ 
5:     UDF( $\text{determined\_intervals}$ )
6:    $\text{determined\_intervals} \leftarrow \emptyset$ 
```

Algorithm 3 Summit Algorithm

```

1: procedure SUMMIT(minAcc, maxAcc, UDF)
2:   determined_intervals  $\leftarrow \emptyset$ 
3:   accumulation  $\leftarrow 0$ 
4:    $B_{\text{marked}} \leftarrow \text{null}$ 
5:   marked_accumulation  $\leftarrow 0$ 
6:   for each snapshot  $B$  in first resolution of Di3 do
7:     accumulation  $\leftarrow$  accumulation at  $B$ 
8:     if minAcc  $\leq$  accumulation  $\leq$  maxAcc then
9:        $B_{\text{marked}} \leftarrow B$ 
10:      marked_accumulation  $\leftarrow$  accumulation
11:      determined_intervals  $\leftarrow$  determined_intervals  $\cup$  intervals of  $B$ 
12:    else
13:      if  $B_{\text{marked}} \neq \text{null}$  AND (accumulation  $>$  maxAcc OR accumulation  $<$  minAcc) AND
        (marked_accumulation  $>$  accumulation OR marked_accumulation  $<$  accumulation) then
14:        determined_intervals  $\leftarrow$  determined_intervals  $\cup$  intervals of  $B$ 
15:        UDF( $B_{\text{marked}}, B$ , determined_intervals)
16:         $B_{\text{marked}} \leftarrow \text{null}$ 
17:        marked_accumulation  $\leftarrow 0$ 
18:        determined_intervals  $\leftarrow \emptyset$ 

```

Map, the function MAP finds all intervals overlapping a given reference interval $I_{\text{reference}}$, and passes them to UDF for further application-specific operations. The Algorithm 2 provides pseudo code of this function, the algorithm explains MAP for one reference intervals, hence the algorithm can be repeated for each reference interval in a reference sample.

4 Di4: 1D Intervals Incremental Inverted Index

Present chapter explains *1D intervals incremental inverted index* (Di4), a general-purpose indexing framework for interval-based data. The *incremental inverted* model of Di4 extends the *inverted* model of Di3 to minimize redundancies. Present chapter discuss Di4 data structure in details, and provide descriptions independent from Di3, such that an interested reader does not require to reference Chapter 3 to understand the concepts and procedures of Di4. The chapter is organized as follows: first, the challenges of Di3 model that motivates the *incremental* structure of Di4 are discussed in Section 4.1, Section 4.2 explains objectives of Di4 model, then the design of Di4 is explained in Section 4.3, and in sections 4.4, 4.5, and 4.6, respectively, the elements of Di4 are formally defined, then first and second resolutions of Di4 for bookkeeping intervals are described these elements, and Section 4.7 explains the algorithms for bookmarking intervals on first and second resolution, and finally, information retrieval based on first and second resolutions of Di4 is explained in Section 4.8.

4.1 Interval indexing beyond Di3

A snapshot of Di3 has an *inverted* structure which bookmarks all the intervals overlapping a snapshot position on the domain. This is an advantage of Di3 snapshot design, mainly because a single snapshot renders a complete picture of all the intervals overlapping a position on the domain independent from the other snapshots. For instance, referring to Figure 4.1, where two “Flight” events are modeled on a time domain; “*Flight-A*” *departures at 9:00AM and lands at 10:00AM*, and “*Flight-B*” *departures at 9:30AM and lands at 10:30AM*. The snapshot at 9:30AM renders: “Flight-B departures and Flight-A is flying”, which is a complete information of the events at 9:30AM. “Flight-A is flying” is not the causal event of 9:30AM snapshot, and can be inferred from either of 9:00AM or 10:00AM snapshots, however, the 9:30AM snapshot holds “Flight-A is flying” information so that it can render the condition of both events independent from the other snapshots.

Despite of the advantages of the inverted structure of Di3, scenarios with dense event occurrences and range queries introduce challenges to it. To be specific, consider a range query as

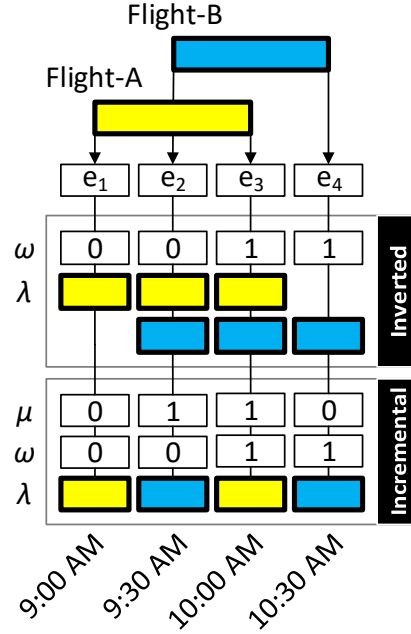


Figure 4.1: Design difference between Di3 (Inverted) and Di4 (Incremental).

“what happens between 9:30AM and 10:00AM?”. To answer the query, snapshots 9:30AM and 10:00AM are processed, which independently provide the following information:

- 9:30AM: “Flight-B departures” and is flying henceforth;
+ “Flight-A is flying”, it was flying before, and is flying henceforth.
- 10:00AM: “Flight-A lands” and it was flying before;
+ “Flight-B is flying”, it was flying before, and is flying henceforth.

The two snapshots in summary inform: “Flight-B departed at 9:30AM while Flight-A was flying, and when Flight-A landed at 10:00AM Flight-B was flying”. The underlined information is explicitly provided by Di3 snapshots, however, it is redundant since having both snapshots, the underlined information can be inferred as follows (italic information):

- 9:30AM: “Flight-B departures” and is flying henceforth;
and based on 10:00AM snapshot, Flight-A is flying now.
- 10:00AM: “Flight-A lands” and it was flying before;
and based on 9:30AM Flight-B is flying now.

Therefore, using only information of *causal* events, a complete picture of event occurrence is can not be recovered.

In scenarios with dense event occurrences, a considerable portion of events bookmarked by a single snapshot, are non-causal events, and a large number of those are repeated from the previous snapshots. In other words, the difference between two consecutive snapshots might be a single event, while a considerable number of information is shared between the snapshots. Di3 design decision for explicit information of non-causal events, although it facilitates reconstructing intervals overlapping a position on domain, which is specially useful for queries such as “find all intervals overlapping a point on domain”, introduces challenges. Indeed, inferable non-causal information and a considerable share between snapshot events introduce a significant (de-)serialization overhead when data do not fit in memory and a persistent data structure is used.

To eliminate redundancy, and make snapshots more practical, an *incremental* structure has been incorporated to the snapshot-based design Di3 design, named *1D intervals incremental inverted index* (Di4) and discussed in present chapter. Di4 keeps only the information about causal events in each snapshot, and infers non-causal events using neighbor snapshots. For instance, referring to Figure 4.1, for the snapshot 9:00AM, Di3 holds information of both “Flight-A” (non-causal) and “Flight-B” (causal), while Di4 holds only “Flight-B” (causal) and a counter of the non-causal events to be inferred. The incremental structure of Di4 the rewrites retrieval algorithms to “infer” non-causal events, and this makes such algorithms more complex than Di3 counterpart. Nevertheless, Di4 operations, spanning data structure manipulation to retrieval, perform significantly faster than Di3, mainly because of reduced events processing and smaller snapshots (de-)serialization.

4.2 Di4 Objectives

Di4 is a multi-resolution single-dimension generic indexing framework over interval-based data (see Figure 4.2) to support query execution and coordinate-oriented retrievals in favor of tertiary analysis, i.e., “sense-making”. The Di4 data model is a representation of interval-based data that maps positions on domain (e.g., Genome) to observed events (e.g., genomic activities), by indexing primary attribute of intervals – coordinates; *left* and *right*, in the NoSQL key-value pair paradigm. The aim of Di4 is to excel in retrievals and data mining operations related to occurrence and co-occurrence of intervals by rendering transparent the complexities of efficient, scalable, extensible, and comprehensive information retrieval system.

The Di4 model holds the following characteristics: (i) relative and absolute knowledge of the occurrence point on the domain, (ii) retrieve relative ordering of intervals for conjoint evaluation (e.g., range queries, or nearest neighbors), (iii) persisted index leveraging on a data (physical) layer technology, (iv) mainly static intervals such that commonly preserve their states and do not necessitate frequent update on persisted information, (v) the model does not map sense of *now*, and (vi) the model accepts user-defined functions (UDF) over primitive Di4 operations.

The Di4 is generic and designed for reusability and extensibility over any domain with (homo-

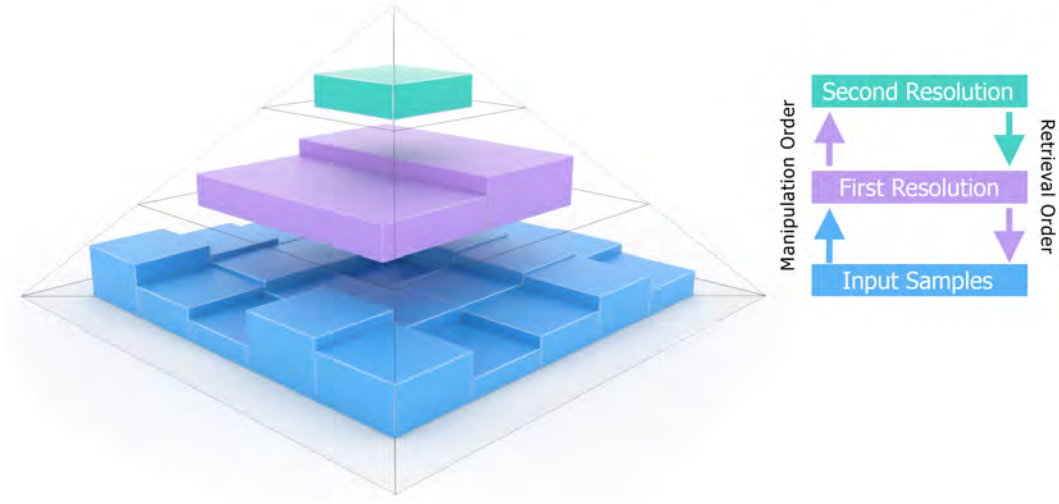


Figure 4.2: Di4 schematic design; the coordinates of intervals from input samples are indexed in a double-resolution paradigm. First resolution indexes the coordinates in inverted (i.e., Di3), and incremental inverted (i.e., Di4) structures. Second resolution aggregates the non-indexed attributes of groups of intervals. The size of persisted information decreases from input samples to the second resolution, with inverted index of first resolution being relatively larger in size compared to incremental inverted index. The manipulation of data structures logically starts from input samples and goes to second resolution through first resolution. However, the retrieval is conducted in reverse order.

geneous and heterogeneous) interval-based data, which is attained by two design decisions. First, the framework is defined at data access layer (DAL), independent from data (physical) layer, with maximum flexibility to business logic layer, while both layers are agnostic of the data model (see Figure 4.3). Second, the diversity of domains, data types, and semantics of intervals supplemented by application-driven operations, motivates implicit definition of *sense-making* functions, such that, in contrast to **what** functions to be implemented, Di4 defines **how** such functions are to be applied. In this regard the Di4 model accepts user-defined functions (UDF) through behavioral design patterns such as *strategy pattern* [28]. This design makes Di4 adaptable to any underlying key-value pair persistence technology (spanning classical data structures such as B+tree to LevelDB inspired by Google's BigTable technology, Apache Accumulo, Apache Cassandra, Apache HBase, or Symas Lightning Memory-Mapped Database (LMDB) (NoSQL databases are surveyed in [26] and [27])), and business logic and presentation layer scenarios (e.g., a graphical user interface). In general, the operations of Di4 are defined in three levels; *physical* level that maps Di4 model to persistence technology which makes the model agnostic to data layer, *logical* and *semantic* levels which respectively define basic region calculus and higher level functions while provide extensibility to different business logic and presentation layer scenarios using generic interface and UDF support. The operations of physical level are persistence technology-specific hence are generally discussed in present chapter, logical level functions are discussed in Section 4.8, and operations of semantic level are discussed by application in Chapter 6.

To avoid any infrastructure-related details and bias, and for the sake of unambiguous descrip-

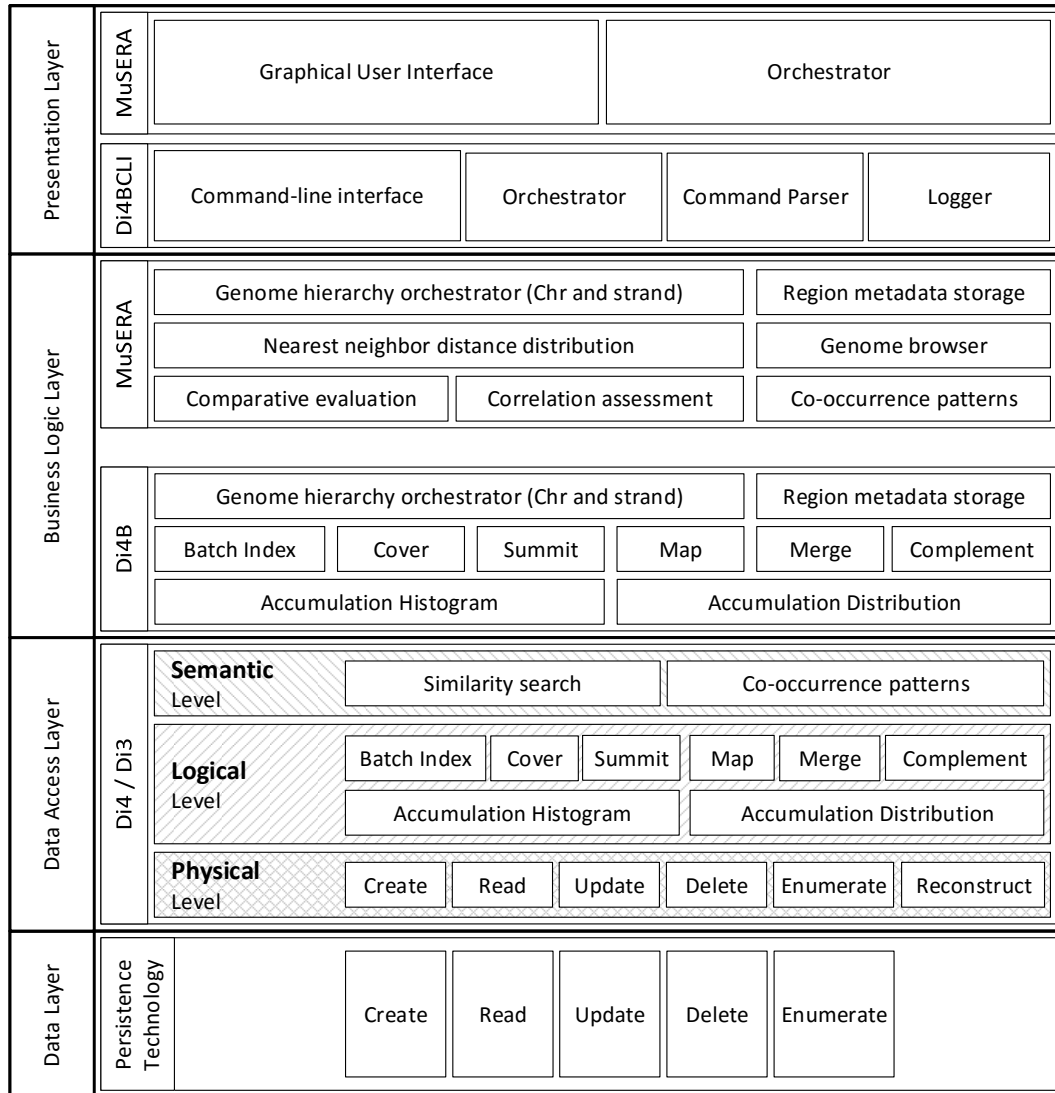


Figure 4.3: Di4 architecture. Di4 and Di3 are defined at DAL in three levels of functionality; The extensibility of Di4 and Di3 to different application scenarios is illustrated by two applications, Di4B-Di4BCLI which is a command line interface for basic functionality of Di4 and Di3 (used for performance evaluation in Chapter 5), and MuSERA which is a graphical tool for comparative evaluation of ChIP-seq peaks (discussed in Chapter 6).

tion, the description of Di4 in the following uses B+tree as underlying data structure to hold Di4 representation of data. Additionally, since the application focus of present dissertation is on genomics, for demonstration and evaluation purpose; Di4 for Bioinformatics (Di4B), which adapts Di4 to computational biology, and Di4B command line interface (Di4BCLI), which provides a console-level accessibility to Di4B, are defined respectively at business logic layer and presentation layer (see Figure 4.3). Additionally, the extensibility of Di4 to application scenarios with graphical user-interface and application-specific functionalities is illustrated

by MuSERA (a graphical tool for comparative analysis of ChIP-seq replicates), described in Chapter 6 (see Figure 4.3).

The cornerstone of Di4 innovations is the point-based index of interval coordinates that excels the retrievals and data mining operations on occurrence and co-occurrence of intervals. In general Di4 is designed for two class of operations as it follows:

- i. **Retrieval** to perform operations such as *similarity search* (e.g., “given a set of samples, find most similar set of samples in terms of co-occurrence of intervals”), *range queries* (e.g., “find regions on domain where a specific number of intervals co-occur”), and *region calculus* (e.g., “given an interval, find all intervals overlapping with it”).
- ii. **Data mining** that mainly aims to operations such as *co-occurrence patterns*, *dependency detection* (e.g., “given a dataset, determine the positions on which the regions usually co-occur” – with respect to the data in repository), and *deviation detection* (e.g., “given a dataset, find positions on which the regions do not commonly co-occur” – based on the data in repository.)

4.3 Di4 Design

A *durative* event has three attributes on domain: (i) *start*, where the action begins, (ii) *stop*, where the action is accomplished, and (iii) *middle*, where different degrees of the action are being executed; such that, a durative event is *happening* between an inclusive start and exclusive stop. For instance, “Flight-A” departures at 9:00AM (*start* attribute), and lands at 10:00AM (*stop* attribute), and all the time between 9:00AM and 10:00AM it was flying (*middle* attribute). *Punctual* events lack middle attribute and refer to transitional actions. For instance, a “genomic mutation” is a punctual event that describes a transition event from “mutated” to “not mutated”. A punctual event should not be mistaken with a *momentary* concept which is a single self-explanatory snapshot on the domain. For instance, “an aircraft is flying” is a momentary concept independent from “when aircraft departures” and “when aircraft lands”. Additionally, a durative event on contiguous domain can be represented by an infinite sequence of momentary snapshots, while a punctual event can not be subdivided.

Di4 models an abstract representation of durative and punctual events on contiguous domains using adaptive snapshots. The model leverages on the *instantaneous model* concept that any durative and punctual event on contiguous domain can be explicitly represented using the start and stop snapshots, and the infinite sequence of momentary snapshots of durative events can be represent implicitly. For instance, two “flight” events can be modeled using four explicit snapshots as follows (see Figure 4.1):

- i. 9:00AM: **Explicit**; Flight-A departures.
- ii. 9:30AM: **Explicit**; Flight-B departures. **Implicit**; Flight-A is flying, and was flying all the

time between 9:00AM and 9:30AM.

- iii. 10:00AM: **Explicit**; Flight-A lands. **Implicit**; Flight-B is flying, and all the time between 9:30AM and 10:00AM both flights were flying.
- iv. 10:30AM: **Explicit**; Flight-B lands. **Implicit**; all the time between 10:00AM and 10:30AM Flight-B was flying.

In general, a snapshot on a domain has two essential attributes; first, a snapshot has at least one *causal event*. For instance, the *causal event* of the snapshot at 9:30AM is the “departure of Flight-B”. Second, a snapshot captures both the causal events and momentary concepts. For instance, the “Flight-A” has infinite momentary snapshots spanning 9:00AM to 10:00AM, including 9:30AM when “Flight-B departs”; hence the snapshot of 9:30AM captures both the causal event (i.e., “Flight-B departs”) and the momentary concept of “Flight-A” (i.e., “Flight-A is flying”). According to these two attributes, the Di4 index model adapts *inverted index* paradigm.

A snapshot represents captured events *explicitly* by a pointer to the subject. For instance, the snapshot of 9:30AM has pointers to “Flight-A” and “Flight-B”, a pointer could be an ID of a flight in a database with all the related information such as aircraft model, passenger list, crew, and etc. A pointer to a causal event is required, however, pointers to momentary concepts might be suboptimal and evaluated as redundant in applications with dense events. For instance, while a pointer to “Flight-A” is given at 9:00AM and 10:00AM snapshots (because “Flight-A” is the causal event), it is not strictly necessary to repeat same information at 9:30AM snapshot where “Flight-A” overlaps with a momentary concept, mainly because it is implicit based on 9:00AM and 10:00AM snapshots. Therefore, to improve adaptability of Di4 to applications with dense events, supplementary to the inverted index, the Di4 index model adapts *incremental inverted index* paradigm where the pointers with momentary concepts are avoided. However, for the sake of accuracy, the incremental model keeps track of the cardinality of momentary concepts – *implicit* representation of events. For instance, the snapshot at 9:30AM under inverted index says “Flight-B departs and Flight-A is flying”, while under incremental inverted index it says “Flight-B departs and one another flight is flying” (see Figure 4.1).

4.4 Notation

Let J interval sets denoted as $\mathbb{S} = \{S_1, \dots, S_j, \dots, S_J\}$ where $S_j = \{I_1^j, I_2^j, \dots, I_i^j, \dots, I_{|S_j|}^j\}$ (see input section of Figure 4.4). $I = [\underline{I}, \bar{I})$ denotes an interval with \underline{I} and \bar{I} stating respectively the lower (left) and upper (right) bounds (ends) of interval I . Let Σ denote the universe of all elements constituting intervals, and $e \in \Sigma$ any element of such domain; an alternative definition of $I_i^j = [e' = \underline{I}_i^j, e'' = \bar{I}_i^j)$ where $e'' > e'$. An interval I is a left-closed and right-open¹ interval of

¹ This property is mainly in favor of interval definition in genomics; extending the algorithms to other interval types is straight-forward.

ascending ordered pair of e elements, that is always $\underline{I} < \bar{I}$; reverse intervals are not considered here. For any pair of intervals I and I' , exactly one property of following interval heptachotomy holds (similar generalization is also provided in [95], [120], and [121]):

- i. $\bar{I} \leq \underline{I}'$: Non-overlapping, I is to the left of I' .
- ii. $\bar{I} < \underline{I}' \wedge \underline{I} \leq \bar{I}' \leq \bar{I}'$: Overlapping, I is to the left of I' .
- iii. $\underline{I} < \underline{I}' \wedge \bar{I}' < \bar{I}$: I covers I' .
- iv. $\underline{I} = \underline{I}' \wedge \bar{I} = \bar{I}'$: perfect overlap.
- v. $\underline{I}' < \underline{I} \wedge \bar{I} < \bar{I}'$: I' covers I .
- vi. $\underline{I}' \leq \underline{I} \leq \bar{I}' \wedge \bar{I}' < \bar{I}$: Overlapping, I is to the right of I' .
- vii. $\bar{I}' \leq \underline{I}$: Non-overlapping, I is to the right of I' .

Let \mathcal{D}^+ denote the first resolution of Di4; $\mathcal{D}^+ = \{B_1, \dots, B_b, \dots, B_{|\mathcal{D}^+|}\}$ is the set of *snapshots* B on Σ , as in Figure 4.4. By definition, the mapping $\mathcal{D}^+ \rightarrow \Sigma$ is injective and non-surjective. A snapshot is a key-value pair element, where the key is $e \in \Sigma$ and the value is a $\langle \mu, \omega, \lambda \rangle$ tuple. The set of all keys and values are denoted respectively $\mathcal{D}_{\text{key}}^+$ and $\mathcal{D}_{\text{value}}^+$. The key is the snapshot coordinate (the notation e_b refers to the coordinate of snapshot B_b), and by definition, it is the unique identifier of B . With reference to the value $\langle \mu, \omega, \lambda \rangle$, the element $\mu_b \in \mathbb{N}_0$ represents the number of intervals that occurred before and terminating after e (i.e., $\mu_b = |\{I_i | e_b \in (I_i, \bar{I}_i)\}|$), such intervals are identified as *passing* intervals. Variable $\omega_b \in \mathbb{N}_0$ denotes the number of intervals whose right-end intersects with e_b (i.e., $\omega_b = |\{I_i | e_b = \bar{I}_i\}|$). The λ_b component is a collection of tuples $(\varphi, @I)$ where each tuple represents an interval intersecting e with either lower or upper bound (i.e., $\lambda_b = \{(\varphi, @I)_i | \underline{I}_i = e_b \vee \bar{I}_i = e_b\}$, such intervals are called *causal* intervals. The element φ is $\varphi_i = L \iff \underline{I}_i = e_b$, and $\varphi_i = R \iff \bar{I}_i = e_b$. Finally, the element $@I$ is a pointer to descriptive metadata of the interval the tuple corresponds to. By definition, a bijection property between I and $@I$ holds (i.e., a pointer points to only one interval, an interval is represented by only one pointer, and no null pointer or un-pointed interval exists), leveraging on this property, $@I$ is considered as the implicit identifier (ID) of I .

Second resolution of Di4 is denoted as \mathcal{D}^* and it is defined as a set of *blocks* \mathbb{B} on \mathcal{D}^+ , $\mathcal{D}^* = \{\mathbb{B}_1, \dots, \mathbb{B}_\tau, \dots, \mathbb{B}_{|\mathcal{D}^*|}\}$ (refer to Figure 4.4). A *block* bookmarks a set of successive snapshots where the lower and upper bounds are determined by a grouping function Δ . For instance, \mathbb{B}_1 on Figure 4.4 is bookkeeping snapshots $\{B_1, B_2, B_3, B_4, B_5\}$. Likewise a snapshot, a *block* is a key-value pair element where the set of all keys and values are denoted respectively $\mathcal{D}_{\text{key}}^*$ and $\mathcal{D}_{\text{value}}^*$. Let $\mathbb{I} = [\underline{\mathbb{I}}, \bar{\mathbb{I}}]$ denote the key of a block for $\mathbb{I}, \bar{\mathbb{I}} \in \{e_1, \dots, e_{|\mathcal{D}^+|}\}$, and $\langle \alpha, \gamma \rangle$ tuple be the value. $\alpha_\tau \in \mathbb{N}^+$ is the maximum accumulation of intervals at the snapshots bookmarked by \mathbb{I}_b , and the component $\gamma_\tau \in \mathbb{N}^+$ is the number of intervals located in the block bookmarked by \mathbb{I}_τ .

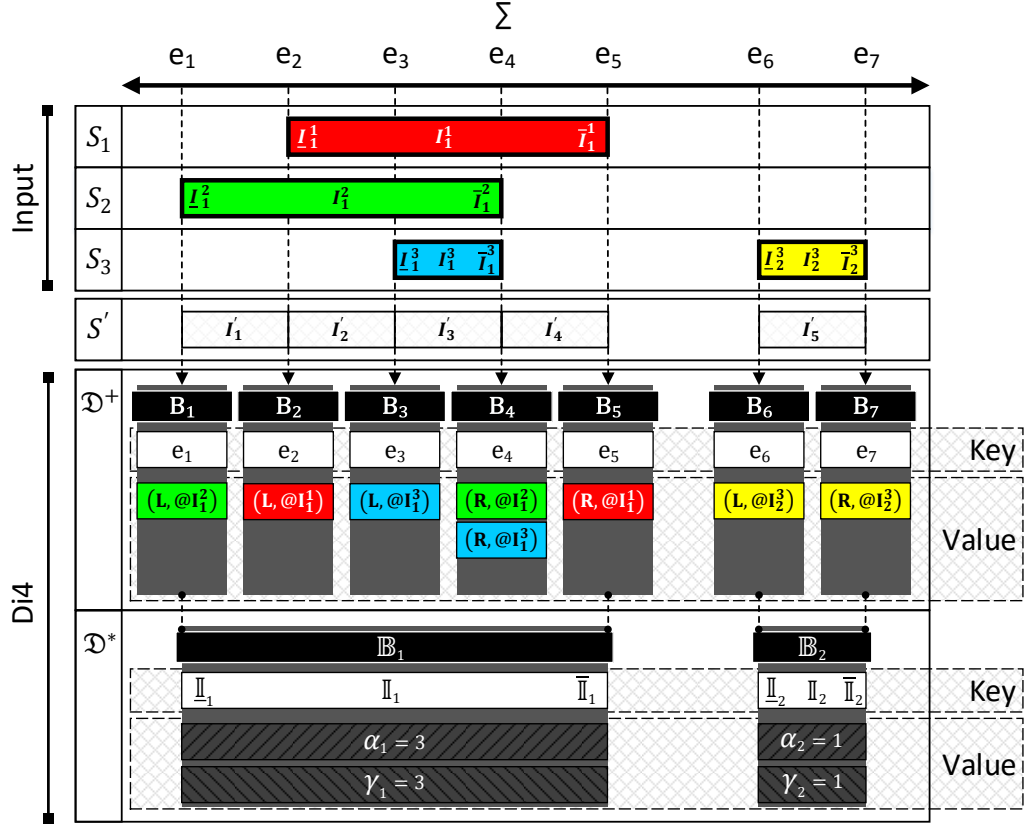


Figure 4.4: **Input:** Intervals are defined on Σ domain and are grouped in S_1 , S_2 , and S_3 samples, where \mathbb{S} is the universe of all sets. The i -th interval of j -th set is denoted as I_i^j and is distinguished by its lower I and upper \bar{I} bounds.

First resolution: The intervals of S' summarize \mathbb{S} in terms of bins: non-overlapping intervals of possibly different length. For instance, I_1^2 and I_1^1 form I'_1 . Di4 leverages on this concept and assigns a snapshot for each unique I' bound. For instance, Di4 bookmarks I'_1 by snapshot B_1 . Each snapshot is bookkeeping intervals overlapping a I' bound. For instance, B_4 is bookkeeping I_1^2 and I_1^3 as ending at e_4 (via λ_4 and ω_4), and one another interval passing e_4 (via μ_4).

Second resolution: Bookmarks are aggregated and form blocks which constitute second resolution. For instance, B_1, B_2, B_3, B_4 , and B_5 are grouped as B_1 , with $\mathbb{I}_1 = [B_1, B_5]$, and maximum accumulation ($\alpha_1 = 3$) and interval count ($\gamma_1 = 3$) are the aggregated information of intervals bookmarked by B_1, B_2, B_3, B_4 , and B_5 snapshots.

The current implementation of Di4 organizes snapshots both for \mathcal{D}^* and \mathcal{D}^+ using B+ tree, with β branching factor which enable efficient storage ($O(\log_\beta n)$) and retrieval ($O(\log_\beta n)$) from a persistent storage. The identifier of a snapshot, e_b , is stored as the key on the B+ tree, while the $\langle \mu, \omega, \lambda \rangle$ tuple forms the corresponding value node (see Figure 4.4).

Let δ_i^j denote the number of snapshots the interval I_i^j intersects (i.e., $\delta_i^j = |\{B_b | I_i^j \leq e_b \leq \bar{I}_i^j\}|$), and ϕ_i^j being the number of intervals I_i^j intersects (i.e., $\phi_i^j = |\{I_m^n | I_m^n \cap I_i^j \neq \emptyset\}|$). The Di4 holds following properties:

1. Leveraging of the definition of an interval, $I = [I, \bar{I}]$: if $\bar{I} = I'$ then $I \cap I' = \emptyset$
2. An interval I_i^j is represented by exactly two snapshots (i.e., $\delta_i^j = 2$) that is: $B_b : e_b = I_i^j$ and $B_{b'} : e_{b'} = \bar{I}_i^j$
3. For an interval $I : e_b = I, e_{b''} = \bar{I}$ if $\exists b' : b < b' < b''$ then I intersects with at least one another interval (i.e., if $\delta_i^j > 2$ then $\phi_i^j > 0$).

4.5 Di4 First Resolution

Now that we have defined intervals in the adaptive snapshotting structure, let describe how this snapshotting is obtained. The superimposition of intervals given by sets in \mathbb{S} induces a new set of non-overlapping intervals, denoted by S' , on Σ (see Figure 4.4). Let I' denote an interval of new set $S' = \{I'_1, I'_2, \dots, I'_{|S'|}\}$ with $I'_i, \bar{I}'_i \in \{\cup_{i,j} I_i^j \cup \cup_{i,j} \bar{I}_i^j\}$. Therefore I' is not necessarily a member of any S_j , and $\sum_j |S_j| \leq |S'| \leq 2 \sum_j |S_j| - 1$.

The first resolution of Di4, \mathfrak{D}^+ , is derived from the intervals of S' , such that, each unique $e = I'_i$ and $e = \bar{I}'_i$ of S' intervals is bookmarked by a \mathfrak{D}^+ snapshot (see Figure 4.4). Any two adjacent intervals of S' introduce a unique e as $e = \bar{I}'_i = I'_{i+1}$ resulting to a common snapshot bookkeeping both bounds (e.g., I'_1 and I'_2 on Figure 4.4 are bookmarked by B_2 , explicitly and implicitly respectively). Leveraging on the snapshot assignment semantic, snapshots hold the property of dichotomies (i.e., $\forall e, \exists ! b : e = e_b$). Each snapshot points to a coordinate on Σ , and conveys the condition of each set at that specific coordinate.² Snapshots summarize information about the coordinates where a *variation* occurs in intervals on Σ , that is $\forall e \mid \exists i, j : I_i^j = e \vee \bar{I}_i^j = e \iff \exists ! B$. A variation occurs when an interval starts (left-end is observed) or ends (right-end is observed). Hence a finite set of snapshots can be used to index finite set of intervals on both discrete and contiguous domains. However, depending on various interval intersections conditions, the number of required snapshots (i.e., $|\mathfrak{D}^+|$) varies spanning from $|S'| + 1$ to $2|S'|$.

The four components of each snapshot, i.e., e_b, μ_b, ω_b , and λ_b , carry the explanatory information about the coordinate on Σ which the snapshot bookmarks. The coordinate specified by $e_b \in \Sigma$ is called *snapshot coordinate* and it has the following property: $\exists i, j : [e_b = I_i^j] \wedge [e_b = \bar{I}_i^j]$. Note that, the set i, j satisfying the former property is not necessarily a singleton, because multiple intervals with the same lower-bound or upper-bound are possible.

Existence of a snapshot B_b identifies the presence of possibly multiple intervals intersect with e_b where at least one interval introduces the causal variation. A snapshot coordinate acknowledge solely the occurrence of a variation at location e_b on Σ , and does not render any information regarding the original intervals. The reconstruction of \mathbb{S} from Di4 is made

²The information are presented in inverted index; that is, information regarding interval I_i^j are present at B_b iff $I_i^j \cap e_b \neq \emptyset$, or in other words $I_i^j \leq e_b \leq \bar{I}_i^j$.

possible through μ_b , ω_b , and λ_b components where the role of each is explained in details in the following.

Causal intervals are stored in the λ_b component which is a set of $(\varphi, @I)$ tuples itself where a one-to-one correspondence between causal intervals and λ_b items holds. Therefore, by definition $|\lambda_b| = |\{I_i^j | \underline{I}_i^j = e_b \vee \bar{I}_i^j = e_b\}|$ and $|\lambda_b| \neq 0$. A pointer to the causal interval is given by $@I$ component of the relative tuple. The pointer is defined in its general form and could be specified as a memory address, or a key of a relational database, or any application specific pointer which points to descriptive metadata of the interval that helps answering queries that require consideration of descriptive metadata. The intersection condition of a causal interval is given by the φ component of the relative tuple which is $\varphi = L \iff \underline{I}_i^j = e_b$ (e.g., with reference to Figure 4.4; I_1^2 at B_1) and $\varphi = R \iff \bar{I}_i^j = e_b$ (e.g., I_1^1 at B_5 of Figure 4.4).

Different from Di3, the Di4 is an *incremental* inverted index where each snapshot explicitly points to causal intervals through the λ_b component, and only implicitly points to non-causal intervals via the μ_b component. Indeed, the component $\mu_b \in \mathbb{N}_0$ specifies the number of non-causal intervals intersecting e_b (i.e., $\mu_b = |\{I_i^j | \underline{I}_i^j < e_b < \bar{I}_i^j\}|$). By definition, exactly two λ 's, λ_b and $\lambda_{b''}$, exist that have reference to I_i^j where $e_b = \underline{I}_i^j$ and $e_{b''} = \bar{I}_i^j$ and any $B_{b'}$ where $\underline{I}_i^j < e_{b'} < \bar{I}_i^j$ has $\mu_{b'} > 0$.

Finally, each snapshot has a component denoted by $\omega_b \in \mathbb{N}_0$ that is defined as the number of intervals whose right-end intersect e_b (i.e., $\omega_b = |\{I_i^j | \bar{I}_i^j = e_b\}|$), and, by definition, $\omega_b \leq |\lambda_b|$. The component ω_b enables $O(1)$ recovery of the number of causal intervals whose left-end or right-end intersects the snapshot; that is: $|\{I_i^j | \bar{I}_i^j = e_b\}| = \omega_b$ and $|\{I_i^j | \underline{I}_i^j = e_b\}| = |\lambda_b| - \omega_b$.

Depending on the intervals of \mathbb{S} , intervals of \mathbb{S}' are not necessarily adjacent, hence neither are the snapshots. In some applications where events in terms of intervals happen sporadic such as in genetics while studying functions of genes, gaps such as the one between I_1^1 and I_2^3 in Figure 4.4 are common indeed. In contrast, applications with frequent event occurrences such as air traffic control over a major city, may rarely introduce gaps. Therefore, gap determination (sometimes referred to as complement) is advantageous for "consonantal" intervals detection.³

Being snapshots assigned to variations, gaps are imaginary intervals where no actual interval is present; hence no variation occurs, and accordingly, no snapshot is assigned to gaps. A gap is implicitly defined with two consecutive snapshots, where the former bookmarks only right-ends (e.g., B_5 in Figure 4.4), and the subsequent snapshot points to only left-ends (e.g., B_6 in Figure 4.4). In general, there is a gap between e_b and e_{b+1} if $\omega_b = |\lambda_b|$ or $\omega_{b+1} = 0 \wedge \mu_{b+1} = 0$. Yet each of these snapshots are equally sufficient to highlight a gap.

³Supposing intervals being events on an agenda, gaps could be interpreted as free slots for insertion of a new event. With intervals representing scheduled tasks for a server, gaps are server idle time apt to new task assignment. Intervals can denote DNA-protein interaction regions on genome, and gaps are DNA positions where no such interaction is observed yet.

Due to our objective toward minimal accesses to the original intervals, Di4 holds least essential information for \mathbb{S} reconstruction from the index and basic retrievals. Information presented by ω or φ are trivial to be obtained using λ and interval coordinate respectively; however, when intervals are stored on external sources, each access to retrieve φ might be sub-optimal, and likewise for ω . These two components minimize source accesses and increase the performance with a penalty of negligible extra storage space requirement. For queries demanding descriptive metadata of intervals, the data are accessible through the $@I$ component.

4.6 Di4 Second Resolution

The first resolution (\mathcal{D}^+) of Di4 is a detailed index that contains minimal essential information for reconstruction of original intervals. The second resolution (\mathcal{D}^*) aggregates \mathcal{D}^+ in two directions: item count and bookmarked information. In general, a group of consecutive snapshots is summarized as a *block* (\mathbb{B}) which is defined as a key-value pair element that forms the second resolution of Di4, $\mathcal{D}^* = \{\mathbb{B}_1, \dots, \mathbb{B}_\tau, \dots, \mathbb{B}_{|\mathcal{D}^*|}\}$ and $\mathbb{B}_\tau \cap \mathbb{B}_{\tau+1} = \emptyset$ (see Figure 4.4). A block *key* is an interval $\mathbb{I} = [\underline{\mathbb{I}}, \bar{\mathbb{I}}]$ on \mathcal{D}^+ for $\underline{\mathbb{I}}$ and $\bar{\mathbb{I}}$ denoting respectively the lower and upper bounds of the interval \mathbb{I} . The *value* of a block is an application and demand specific aggregation of the snapshots grouped in the block. For instance, the value could be the *count* of the intervals bookmarked by the snapshots grouped by the block, or an aggregation of p-values of the intervals. Similar to first resolution, the blocks of second resolution can be persisted using any key-value pair storage technology; current implementation of Di4 uses classical B+tree.

The grouping and aggregation functions are described in details in following sections.

4.6.1 Grouping

A snapshot exists for each unique bound which bookmarks the occurrence of an event on Σ . Accordingly, a function can define a set of adjacent snapshots that represent a collection of event occurrences. Depending on the semantic of the function, such collections convey different messages, or reveal various aspects of the events. For instance, let Σ be date on a calendar, and intervals representing presence of an airplane in an airport for lower and upper bound of an interval being respectively the arrival and departure time of the airplane. Groups of snapshots of a maximum distance of 5min are representing the traffic time on the runway. As another example, let Σ be the DNA of a species and intervals representing regions of genetic function observations. Collections of all overlapping intervals could be interpreted as regions of biological interaction of the study.

A grouping function Δ , groups consecutive snapshots in a block (\mathbb{B}) by determining the two bounds of \mathbb{I} such that $\underline{\mathbb{I}} \in \{B_1, \dots, B_b\}$ and $\bar{\mathbb{I}} \in \{B_{b+1}, \dots, B_{|\mathcal{D}^+|}\}$. The function Δ is application-specific and user-defined. In general, Δ takes the snapshots of Di4 as input, and returns a set of block keys ($\Delta(\mathcal{D}_{\text{key}}^+) = \{\mathbb{I}\}$). If a custom Δ function is not provided, Di4 uses a default function that groups snapshots based on “gaps”, such that all the snapshots between two

consecutive “gaps” are grouped together. In general, supposing $\mathbb{I}_\tau = B_\alpha$, $\bar{\mathbb{I}}_\tau = B_\beta$, and let $A(B_b)$ denote accumulation of intervals at snapshot B_b , the Δ function is defined as follows:

$$\Delta(\mathcal{D}_{\text{key}}^+) = \left\{ \mathbb{I} \mid A(B_{\alpha-1}) = 0 \wedge A(B_{\beta+1}) = 0 \wedge (\forall B_b, \alpha < b < \beta : A(B_b) \neq 0) \right\}$$

4.6.2 Aggregation

The first resolution provides detailed information for each single $e \in \Sigma$ spanning from interval coordinate and accumulation count to descriptive metadata. The information of the snapshots constituting blocks are then aggregated in favor of providing summary statistics on the specific attributes. The aggregate function(s) and target attribute(s) are application-specific and user-defined, determined in accordance with application-oriented data analysis and retrieval demand. For instance, in the aforementioned airport example, the number of the intervals constituting a block is a pragmatic aggregation which can reveal extensive aspects such as high traffic on the runway, where the block with highest value, is the time period of heaviest runway traffic. Although such information can be collected by directly processing snapshots; however, storing custom aggregations for each block reduces snapshot access demands specially if the a particular aggregation is commonly appealed.

The default aggregation of Di4 calculates *maximum accumulation* ($\alpha \in \mathbb{N}$) and *counts* ($\gamma \in \mathbb{N}$) intervals bookmarked by the grouped snapshots. Hence, the default aggregation results a tuple as (α, γ) as *value* of each block (see Figure 4.4). The two aggregated attributes are generally defined as follows:

$$\alpha_\tau = \max_{b=\mathbb{I}}^{\bar{\mathbb{I}}} (\mu_b + |\lambda_b| - \omega_b)$$

$$\gamma_\tau = \left| \left\{ I_i^j \mid \mathbb{I}_\tau \leq I_i^j \wedge \bar{I}_i^j \leq \bar{\mathbb{I}}_\tau \right\} \right|$$

4.7 Di4 Indexing Algorithms

The indexing process consists of essential operation for creating the two resolutions of Di4 data structure from input intervals. The process starts from $S_j \in \mathbb{S}$ sets, proceeds iterating through the sets and indexing intervals, and stops when \mathcal{D}^+ and \mathcal{D}^* are properly structured by all intervals. Di4 organizes snapshots and blocks on independent B+trees⁴ forming first and second resolution respectively.

Indexing algorithms for both first and second resolutions ensure data logical integrity by maintaining the accuracy and consistency of intervals on \mathcal{D}^+ and \mathcal{D}^* at a full procedure of data structures manipulation. First resolution indexing algorithm ensure that exactly two snapshots point to an interval, and if the interval intersects another interval at least one

⁴Roger Knapp's excellent B+tree implementation is used, the library is open-source distributed under the Apache License, Version 2.0 and is available at <http://www.nuget.org/packages/CSharpTest.Net.BPlusTree>.

snapshot exists between the priors with $\mu \geq 1$. The second resolution indexing algorithm maintains data logical integrity by creating non-overlapping blocks, and exactly one block referring to the set of snapshots the segmentation function defines. The algorithms will not modify the pointers to intervals, and will change only the affected blocks or snapshots through an update procedure.

The process of populating both resolutions is described in details in following sections.

4.7.1 Bookmarking intervals on First Resolution

The first step of initializing Di4 data structure is populating first resolution (\mathcal{D}^+) by source intervals. Likewise, the first step of indexing a new set is to update \mathcal{D}^+ by new intervals. The source intervals (i.e., S_j) are the input of first resolution indexing procedure. First resolution index provides coordinates of intervals to be used both for creating second resolution, and various retrieval functions.

In general, the procedure of indexing an interval is twofold. First, create, or update if already exist, two snapshots referring to lower and upper bounds of the interval I_i^j (i.e., $B_{b'}|e_{b'} = I_i^j$ and $B_{b''}|e_{b''} = \bar{I}_i^j$) while the $\mu_{b'}$ and $\mu_{b''}$ components of the new snapshots are initialized with respect to closest right and left-hand-side neighbors respectively. Second, increment the μ_b component of all snapshots between lower and upper bounds (i.e., $\mu_b|e_{b'} < e_b < e_{b''}$). By definition, a snapshot at e_b already exist *iff* I_i^j intersects with at least one another interval. Therefore, a correct value of μ_b is initialized through a snapshot *read* (first step), and is maintained by n *updates* for n newly introduced intervals overlapping at e_b (second step). Di4 is stored on a persistent storage and each *read* or *update* request is an I/O operation which shall be minimized to avoid performance penalty.

In this regard, similar to Di3, two types of indexing algorithms are proposed. *Single-pass indexing*: correctly initializes μ_b and maintains it's value at a new interval insertion at a cost of $n + 1$ I/O operations. *Double-pass indexing*: neither fully initializes nor maintains μ_b at *first-pass* rather updates *all* μ 's to correct values by *second-pass* at a cost of 1 I/O operation per each μ . The two algorithms are compared in Section 4.7.1.

Single-pass vs Double-pass

The superiority of one algorithm over the other for indexing a new set S_j depends on $|S_j|$ and $|\mathcal{D}^+|$. Present section discusses the differences between the two algorithms, and benchmarks them using ENCODE narrow peaks with 34, 142, 740 intervals distributed in 970 samples; and running on an Amazon EC2 machine with Intel®Xeon®E5-2670 v2 processor, 320 GFLOPS, and 122GB RAM. The ENCODE narrow peaks are chosen for benchmarking for two reasons. First, it is a real and commonly used dataset. Second, the accumulation distribution of intervals, which is important, because if two intervals intersect, then there will be at least one snapshot that single pass indexing algorithm will update it. The accumulation distribution

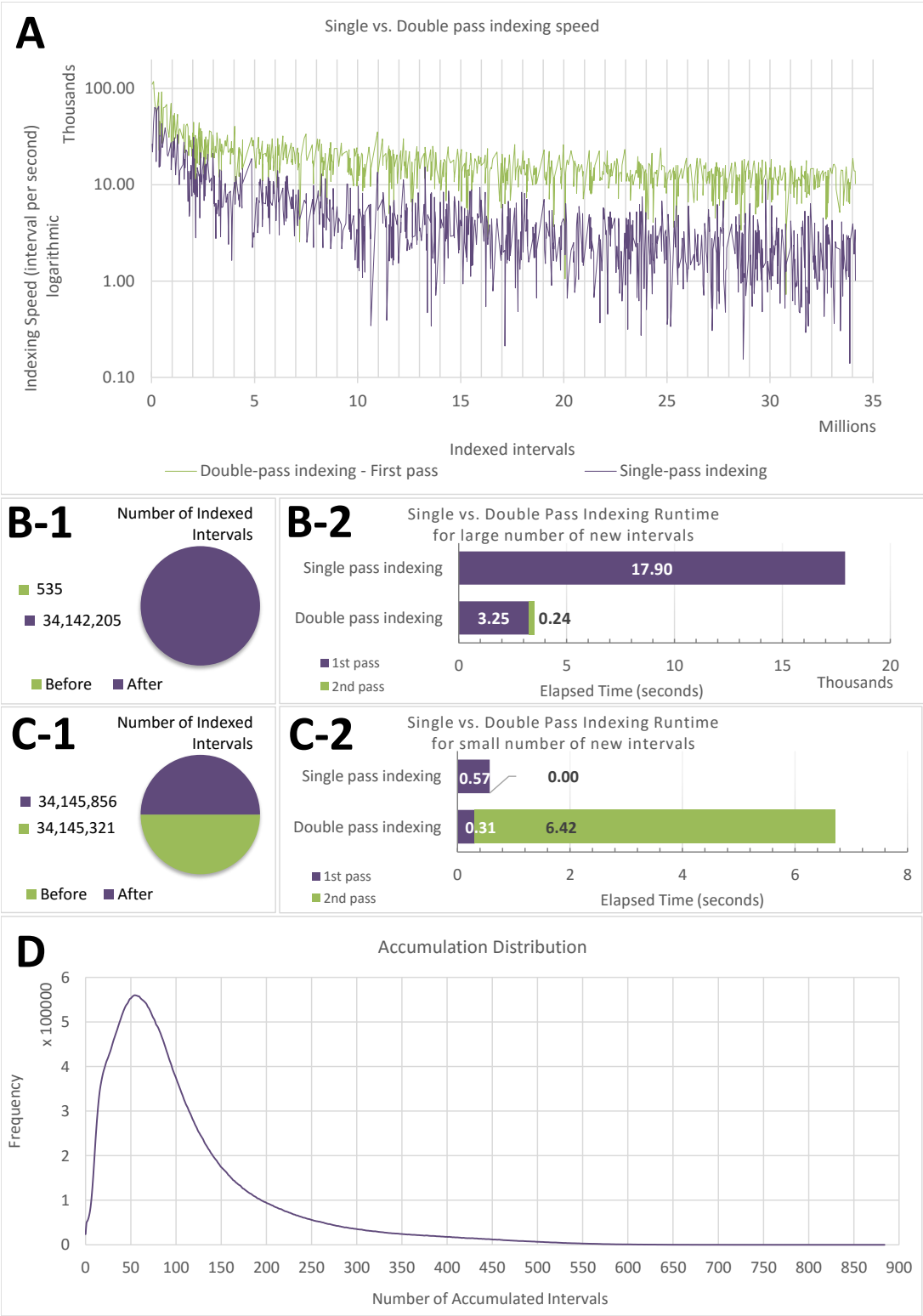


Figure 4.5: Single-pass indexing compared to double-pass indexing. **A:** double-pass vs. single-pass for creating snapshots. **B:** superiority of double-pass for initializing data structure. **C:** superiority of single-pass for updating Di4. **D:** accumulation distribution.

of ENCODE narrow peaks dataset is given on panel D of Figure 4.7.1, which shows that 20 to 200 intervals accumulation is quite common, whereas, conditions with 800 intervals intersecting is rarely observed. Such accumulation distribution allows assessing different aspects of algorithms.

Initialization superiority: double-pass indexing is superior for initializing \mathcal{D}^+ by S_j (i.e., when $\sum_j |S_j| \gg |\mathcal{D}^+|$). In general, double-pass indexing is relatively faster than single-pass in organizing essential snapshots on underlying B+tree structure of \mathcal{D}^+ . This is due to the overhead of data structure consistency conservation at each new interval insertion by single-pass, which is neglected by first-pass of double-pass indexing method. This penalty is in direct relation with interval accumulation distribution; such that, the denser the intervals are, the more updates are to be performed by single-pass method that reduces indexing speed. The methods are benchmarked by adding $3.4E + 6$ intervals to a \mathcal{D}^+ which is initialized by 535 intervals (see panel B-1 on Figure 4.5). The first-pass of double-pass indexing algorithm outperforms single-pass method for organizing necessary snapshots (see panel A on Figure 4.5). The second-pass of double-pass algorithm is significantly fast; this is because, it neither inserts a new snapshot, nor changes the size of inserted snapshots (i.e., changing $|\lambda_b|$ which affects (de)serialization), rather modifies μ component only. Therefore, double-pass indexing, considering both first-pass and second-pass, outperforms single-pass indexing algorithm for this scenario; 58min compared to 5h (see panel B-2 on Figure 4.5).

Update superiority: single-pass indexing is comparatively optimal when $\sum_j |S_j| \ll |\mathcal{D}^+|$, which is a common scenario of updating \mathcal{D}^+ by few intervals. The single-pass and double-pass indexing algorithms are benchmarked for updating \mathcal{D}^+ with 535 intervals which is initialized by $3.4E + 6$ intervals (see panel C-1 on Figure 4.5). The single-pass algorithm takes 57msec to update \mathcal{D}^+ , while the double-pass algorithm takes ~7sec in total (see panel C-2 on Figure 4.7.1). This is because, single-pass maintains the consistency of data structure by updating few snapshots, while double-pass algorithm iterates over all snapshots and updates same number as single-pass. Therefore, the overhead of second-pass in this scenario is iteration over all snapshots, which makes double-pass indexing algorithm relatively slower than single-pass method.

Single-pass indexing

The single-pass indexing algorithm updates the first resolution of Di4 (\mathcal{D}^+) by a given interval I_i^j . The algorithm is consistent, such that it modifies only the effected snapshots of \mathcal{D}^+ , and ensures the accuracy of \mathcal{D}^+ data structure at each successful indexing of every single interval. The algorithm is divided into three major steps for interval I_i^j : (i) index lower bound of the interval using B_α snapshot such that $e_\alpha = I_i^j$, (ii) index upper bound of the interval in B_β snapshot where $e_\beta = \bar{I}_i^j$ ($\alpha < \beta$), and (iii) increment μ component of all snapshots between B_α and B_β . A pseudo code of single-pass indexing is given in Algorithm 4 and described as follows.

Algorithm 4 Single-pass Indexing Algorithm

```

1: procedure INDEX( $I_i^j$ )
2:   Read  $B_\alpha \mid e_{\alpha-1} < \underline{I}_i^j \leq e_\alpha$ 
3:   if  $e_\alpha = \underline{I}_i^j$  then
4:     Update  $\lambda_\alpha \leftarrow \lambda_\alpha \cup (L, @I)_i^j$ 
5:   else
6:     Create  $B_{\alpha'}$  as  $e_{\alpha'} \leftarrow \underline{I}_i^j$ ,  $\lambda_{\alpha'} \leftarrow (L, @I)_i^j$ ,  $\mu_{\alpha'} \leftarrow \mu_\alpha + \omega_\alpha$ , and  $\omega_{\alpha'} \leftarrow 0$ 
7:     Update  $\mu_\alpha \leftarrow \mu_\alpha + 1$ 
8:     for each  $B_b \mid B_\alpha < B_b < B_\beta$  where  $B_\beta \leq \bar{I}_i^j < B_{\beta+1}$  do
9:       Update  $\mu_b \leftarrow \mu_b + 1$ 
10:    if  $e_\beta = \bar{I}_i^j$  then
11:      Update  $B_\beta$  as  $\lambda_\beta \leftarrow \lambda_\beta \cup (R, @I)_i^j$  and  $\omega_\beta \leftarrow \omega_\beta + 1$ 
12:    else
13:      Update  $\mu_\beta \leftarrow \mu_\beta + 1$ 
14:      Create  $B_{\beta'}$  as  $e_{\beta'} \leftarrow \bar{I}_i^j$ ,  $\lambda_{\beta'} \leftarrow (R, @I)_i^j$ ,  $\mu_{\beta'} \leftarrow \mu_\beta + \omega_\beta$ , and  $\omega_{\beta'} \leftarrow 1$ 

```

The algorithm starts by indexing \underline{I}_i^j . The first step is to find the first snapshot B_α intersecting I_i^j such that e_α is the minimum indexed coordinate greater than or equal to \underline{I}_i^j . If $\underline{I}_i^j = e_\alpha$ then the referred variant coordinates are identical and hence B_α is the snapshot that can also bookmark I_i^j . Accordingly, the lower bound of the interval is indexed through updating λ_α by adding a new tuple $(L, @I)_i^j$ which makes B_α the snapshot that indexes \underline{I}_i^j . However, if $\underline{I}_i^j < e_\alpha$ then it implies that the lower bound of the interval refers to a coordinate that is not indexed yet, hence a new snapshot $B_{\alpha'}$ with $e_{\alpha'} = \underline{I}_i^j$ is created. The new causal coordinate (i.e., $e_{\alpha'}$) is environed by two consecutive indexed variations as: $e_{\alpha-1} < e_{\alpha'} < e_\alpha$; and any interval I^* intersects $e_{\alpha'}$ iff it overlaps both $e_{\alpha-1}$ and e_α . However, to minimize persistent storage accesses and since B_α is already fetched, the condition $I^* \cap e_{\alpha'}$ is assessed independent from $e_{\alpha-1}$ that is: $I^* \cap e_{\alpha'} \iff \underline{I}^* < e_\alpha \leq \bar{I}^*$. Note that, since $e_{\alpha-1}$ and e_α are consecutive snapshots: $\underline{I}^* < e_\alpha \leq \bar{I}^* \Rightarrow \underline{I}^* \leq e_{\alpha-1} < \bar{I}^* \Rightarrow \underline{I}^* < e_{\alpha'} < \bar{I}^* \Rightarrow I^* \cap e_{\alpha'} \neq \emptyset$. Based on these reasonings the new snapshot $B_{\alpha'}$ is initialized as following and inserted into \mathcal{D}^+ :

- $\mu_{\alpha'} \leftarrow \mu_\alpha + \omega_\alpha$: By definition, the lower bound of an interval that *stops* at or *passes* e_α is indexed at $B_{\alpha-1}$ or a prior snapshot. Accordingly, since $\alpha - 1 < \alpha' < \alpha$, such intervals *pass* $B_{\alpha'}$ as well and their count is obtained by adding the number of intervals *stopping* at B_α to those who *pass* the snapshot.
- $\omega_{\alpha'} \leftarrow 0$: the coordinate was not indexed, hence none of the indexed intervals overlap $e_{\alpha'}$ with its upper bound.
- $\lambda_{\alpha'} \leftarrow \{(L, @I)_i^j\}$: I_i^j is the only causal interval.

The algorithm continues by iterating from B_α to B_β exclusively for e_β being the minimum indexed coordinate greater than or equal to \bar{I}_i^j . Let $\{B_\alpha, \dots, B_b, \dots, B_\beta\}$ be the set of snapshots

iterated over; if I_i^j intersects e_α and e_β then it also intersects e_b . By definition, \mathcal{D}^+ preserves an incremental structure that is by explicitly referring to an interval at two boundary snapshots through λ components, and implicitly at interior snapshots through μ components. Therefore, μ_b components are incremented while iterating over to maintain the incremental property.

The last step is to index the upper bound coordinate of the interval I_i^j . The algorithm starts by comparing \bar{I}_i^j and e_β . If $e_\beta = \bar{I}_i^j$ then the upper bound is referring to a coordinate that is already indexed, hence its only required to update B_β by adding $(R, @I)_i^j$ tuple to λ_β and incrementing ω_β component. However, if $e_\beta < \bar{I}_i^j$, then a new snapshot $B_{\beta'}$ with $e_{\beta'} = \bar{I}_i^j$ is required. In this case, the upper bound of the interval is pointing to a coordinate between two consecutive indexed coordinates e_β and $e_{\beta+1}$. An interval I^* intersects the upper bound of I_i^j iff it overlaps both e_β and $e_{\beta+1}$. However, the condition can be verified by leveraging on the information provided by B_β only. In this regard, the new snapshot $B_{\beta'}$ is initialized as follows:

- $\mu_{\beta'} \leftarrow \mu_\beta + \omega_\beta$: Based on aforementioned reasoning, any interval *passing* or *stopping* at e_β , *passes* $\mu_{\beta'}$.
- $\omega_{\beta'} \leftarrow 1$: The coordinate was not indexed, therefore no other interval except I_i^j overlaps $e_{\beta'}$ with its upper bound.
- $\lambda_{\beta'} \leftarrow \{(R, @I)_i^j\}$: I_i^j is the only causal interval at $e_{\beta'}$.

The algorithm concludes by inserting $B_{\beta'}$ into \mathcal{D}^+ .

Double-pass indexing

The fundamental logic of indexing an interval on first resolution can be viewed as: organize upper and lower bounds in distinct snapshots, and update all snapshots in between. Indexing the two bounds may require new snapshots to be initialized and persisted on the underlying B+tree structure. In contrast, updating procedure does not introduce new snapshots rather it updates only μ component of existing ones; an update with no impact on the value size of a key.⁵ The *single-pass* indexing algorithm intertwines the two concepts which despite of the advantages, it has drawbacks of redundant B+tree traverses and snapshot updates when multiple intervals overlap (which is quite common indeed); in addition to the overhead of correct initialization requirement of a new snapshot. The value of a μ component is the number of intervals whose left-end is observed in λ component of prior snapshots and right-end is not observed yet. Therefore, μ can be set to its correct value with a single update. *Double-pass* indexing leverages on these reasonings and separates lower and upper bound

⁵With reference to Notation section (Section 4.4), the *key* is snapshot coordinate, and *value* is (μ, ω, λ) tuple. Update procedure modifies μ component only which is an integer and incrementing its number would not effect the size of μ (if not compressed); therefore, the procedure does not affect *value* size and would not require B+tree modification. To clarify the point, suppose λ component is to be updated which is a list of $(\varphi, @I)$ tuple itself. Any tuple added to λ will increase the *value* size and may incur an update to organization of *values* on the physical storage.

Algorithm 5 First pass of Double-pass Indexing Algorithm

```

1: procedure INDEXFIRSTPASS( $I_i^j$ )
2:   Read  $B_\alpha \mid e_{\alpha-1} < \underline{I}_i^j \leq e_\alpha$ 
3:   if  $e_\alpha = \underline{I}_i^j$  then
4:     Update  $\lambda_\alpha \leftarrow \lambda_\alpha \cup (L, @I)_i^j$ 
5:   else
6:     Create  $B_{\alpha'}$  as  $e_{\alpha'} \leftarrow \underline{I}_i^j$ ,  $\lambda_{\alpha'} \leftarrow (L, @I)_i^j$ ,  $\mu_{\alpha'} \leftarrow \mu_\alpha + \omega_\alpha$ , and  $\omega_{\alpha'} \leftarrow 0$ 
7:     Read  $B_\beta \mid B_\beta \leq \bar{I}_i^j < B_{\beta+1}$ 
8:     if  $e_\beta = \bar{I}_i^j$  then
9:       Update  $B_\beta$  as  $\lambda_\beta \leftarrow \lambda_\beta \cup (R, @I)_i^j$  and  $\omega_\beta \leftarrow \omega_\beta + 1$ 
10:    else
11:      Create  $B_{\beta'}$  as  $e_{\beta'} \leftarrow \bar{I}_i^j$ ,  $\lambda_{\beta'} \leftarrow (R, @I)_i^j$ ,  $\mu_{\beta'} \leftarrow \mu_\beta + \omega_\beta$ , and  $\omega_{\beta'} \leftarrow 1$ 

```

indexing from μ component manipulation. The algorithm is advantageous when the number of intervals to be indexed is considerably high, that includes scenarios such as initializing or populating Di4 by a considerable number of intervals.

Similar to single-pass, double-pass algorithm is consistent, such that it modifies only the effected snapshots. The first pass manipulates ω and λ components, and μ is updated by second pass. Therefore, double-pass indexing algorithm maintains the accuracy of the D^+ by successful execution of both passes. First pass iterates over intervals and populates first resolution; and second pass iterates over snapshots and maintains the accuracy. The first and second pass procedures are explained in details in the following.

First pass The first pass manipulates \mathfrak{D}^+ by necessary snapshots to index the boundaries of intervals. A pseudo code of first pass of double-pass indexing is given in Algorithm 5 and described as follows. The algorithm iterates over a set of intervals (S_j), and indexes intervals in batch. For an interval I_i^j it inserts two snapshots B_α and B_β respectively for lower and upper bounds (i.e., $e_\alpha = \underline{I}_i^j$ and $e_\beta = \bar{I}_i^j$) through an atomic operation. The algorithm starts by populating \mathfrak{D}^+ by B_α . If the coordinate e_α is already indexed then the same snapshot can bookmark upper bound of I_i^j too; in this regard, it is required to update the λ component of the snapshot by $(L, @I)_i^j$ tuple. If the coordinate is not indexed, then the snapshot B_α is initialized as following and inserted in \mathfrak{D}^+ :

$$\mu_\alpha \leftarrow 0, \omega_\alpha \leftarrow 0, \text{ and } \lambda_\alpha \leftarrow (L, @I)_i^j$$

. The algorithm continues by indexing B_β . If a snapshot at e_β coordinate exists, then the same snapshot will be updated as following to bookmark lower bound of I_i^j : $\omega \leftarrow \omega + 1$ and $\lambda \leftarrow \lambda + (R, @I)_i^j$. However, if the coordinate is not indexed, the snapshot B_β is initialized as following and inserted into \mathfrak{D}^+ : $\mu_\beta \leftarrow 0, \omega_\beta \leftarrow 1, \text{ and } \lambda_\beta \leftarrow (R, @I)_i^j$.

Algorithm 6 Second pass of Double-pass Indexing Algorithm

```

1: procedure INDEXSECONDPASS
2:    $\mu_{\text{temp}} \leftarrow 0$ 
3:    $T \leftarrow \emptyset$ 
4:   for  $b = 0$  to  $b = |\mathcal{D}^+|$  do
5:      $\mu_{\text{temp}} \leftarrow |T| + \omega_b$ 
6:     for each  $(\varphi, @I) \in \lambda_b$  do
7:       if  $@I \in T$  then
8:          $T \leftarrow T \setminus @I$ 
9:       else
10:         $T \leftarrow T \cup @I$ 
11:     if  $\mu_b \neq \mu_{\text{temp}}$  then
12:       Update  $\mu_b \leftarrow \mu_{\text{temp}}$ 

```

Second pass The second pass procedure ensures the accuracy of data structure by iterating over existing snapshots and updating the μ components. The pseudo code is given in Algorithm 6 and described as follows. Let T be a temporary list of pointers to intervals which is updated while iterating through snapshots using λ_b ; and T_b denotes T set when it reaches snapshot b . If upper bound of an interval is observed, its pointer will be added to T , and will be removed when the lower bound is determined. Therefore, at each snapshot B_b , $T_b - \{I | \bar{I} = e_b \vee \bar{I} = e_b\}$ is the set of intervals *passing* B_b . The second pass algorithm leverages on $|T_b|$ to determine the correct value of each μ_b component ($\mu_b = |\{I | \bar{I} < e_b < \bar{I}\}|$), and updates it if persisted value differs the determined one. The precedence of actions μ_b determination and T_b update by λ_b is optional, however following conditions shall be considered and an option be chosen for correct implementation:

- Determine μ_b first then update T_b : $\mu_b \leftarrow |T_b| - \omega_b$ (our choice)
- Update T_b first then determine μ_b : $\mu_b \leftarrow |T_b| - |\lambda_b| + \omega_b$

The algorithm starts iteration from the first snapshot (B_1). By definition, B_1 being the first indexed coordinate implies that there is no interval with left-end at a prior coordinate; therefore $\nexists I | \bar{I} < e_1 \leq \bar{I} \Rightarrow \mu_1 = 0, \omega_1 = 0$. Additionally, at the beginning of iteration the set $T_1 = \emptyset$ and $\omega_1 = 0$ which results $\mu_1 = 0$. However, since μ_1 was 0 and it equals the determined value, the snapshot will not update; and the set T_1 will be populated by λ_1 . The algorithm moves to next snapshot (B_2) and updates μ_2 component *iff* $\mu_2 \neq |T_2| - \omega_2$. Then the algorithm updates T_2 with respect to λ_2 that is by adding $@I$ to T_2 if the corresponding tuple is $(L, @I)$, or remove it from T_2 if $< R, @I >$. The process continues by applying same procedure on all subsequent snapshots.

4.8 Information Retrieval based on Intervals

Generally, genomic intervals are organized in files (known as *samples*) in logical accordance with common NGS pipelines. Such organization is not necessarily optimal for cross-sample inferences. Therefore, Di4 aims at organizing the intervals in data structures that provide optimal cross-sample region-level inferences. The data structure is supplemented by *information retrieval* (IR) functions that provide means of finding intervals from \mathcal{D}^* and \mathcal{D}^+ that satisfy an IR function criteria. Additionally, the functions leverage on behavioral design patterns such as *strategy pattern* [122] that facilitates adaptation to application by enabling user defined functions (UDF). Retrieval functions are generally defined as follows:

$$f_{\text{Name}}([\mathfrak{R}], [\mathcal{D}^*], [\mathcal{D}^+], [f(T)]) = \text{output}$$

where *Name* is the function name. The \mathfrak{R} denotes a set of arguments defining the coordinate criteria of the function. A retrieval function may operate upon either of \mathcal{D}^* or \mathcal{D}^+ , or may leverage on both using \mathfrak{R} arguments. This defines coordinate and feature oriented functions described in following. Additionally, the UDF is denoted by $f(T)$, where T is the set of determined intervals. Note that, $f(T)$ is provided via *design pattern* and may apply on any attribute(s) of T intervals (e.g., coordinates, or meta-data), and may produce any output (e.g., a single value, an arbitrary size tuple of possibly different items). This defines first and higher order functions described in following.

Coordinate and feature oriented functions

The relative location (or occurrence) of intervals is the key attribute of majority of interval-based operations, which, if supplemented by application-driven meta-data aggregation, reveals extensive aspects of the events abstracted by intervals. For instance, co-localization of genomic intervals of homogeneous samples (e.g., biological/technical replicates) with a known region on DNA (e.g., transcription factor binding site) is an evidence of biological activity on the position with combined statistical significance obtained by aggregation of individual significances (e.g., combining p-values using Fisher's combined probability test). In general, such inferences are either based on a big collection of data, or a small sub-section of a big collection; nevertheless, an optimal method for cross-sample interval manipulations is needed. The Di4 organizes intervals by indexing the primary attribute of any interval – coordinates – on first resolution. Additionally, alternative application-driven commonly-used attributes are aggregated by second resolution to optimize linear scan on first resolution for a non-indexed attribute. Accordingly, the retrieval functions are categorized as *coordinate-oriented functions* (COF) on \mathcal{D}^+ , and *feature-oriented functions* (FOF) on \mathcal{D}^* . A COF determines intervals located at particular coordinates defined by function criteria (e.g., co-localization of intervals, or n -th closest neighbor), and a FOF targets intervals of specific characteristic(s) complying function criteria (e.g., regions of specific interval accumulation).

First and higher order functions

The coordinates are the least-common characteristic between all the events generalized as intervals, and are used as indexing key and define relative location and ordering. Intervals of different types vary in the meta-data. For instance, in genomics, an interval could represent ChIP-seq peak with meta-data providing p-values, while another interval may present a gene that have geneID in meta-data. The disparity of meta-data type introduce two challenges: first, data persistence, second, meta-data-specific aggregations. The reference to meta-data by $@I$ component of λ addresses the first challenge by enabling varied meta-data types to be stored on external storage with a corresponding persisted pointer in Di4. For instance, meta-data could possibly be stored on a SQL databased where different types are stored on different tables (e.g., ChIP-seq peaks are on table different from genes), and a pointer to a record on a table is stored on Di4. The second challenge is addressed by leveraging on *functional programming* paradigm. The diversity of meta-data types and particular application-driven operations prevents explicit definition of aggregate functions. Therefore, in contrast to “**WHAT** aggregation functions should be defined”, Di4 defines “**HOW** such functions should be applied”. In this regard, Di4 provides *higher-order* IR functions that accept custom aggregate functions as input in addition to retrieval criteria. Leveraging on pattern strategy, the polymorphic higher-order IR functions are defined, which allow a generic aggregate function as input. Such a design pattern encourages a developer to focus on the essence of “sense-making” from retrieved intervals rather than technical details of the retrieval. Additionally, aggregate functions that are composition of different functions are allowed, which expands the re-usability of custom aggregate functions.

Similar to Di3, also Di4 provides functions to facilitate highlighting various aspects of indexed data; as follows:

- i. **COVER** the function determines a set of non-overlapping intervals of a contiguous intersection with accumulation constrains (see **COVER** example on Figure 4.6).
- ii. **SUMMIT** it is a variant of **COVER** and reports a sub-region of maximum accumulation within **COVER** accumulation constrains (see **SUMMIT** example on Figure 4.6).
- iii. **MAP** given a set of intervals, it determines a set of overlapping intervals from the data structure (see **MAP** example on Figure 4.6).
- iv. **MERGE** determines regions on the domain (Σ) with at least one event occurrence.
- v. **COMPLEMENT** is the inverse of **MERGE** that finds regions on domain (Σ) with no event occurrence.
- vi. **ACCHIS**: *accumulation histogram* provides a contiguous set of intervals defined by snapshots valued as the accumulation of composing intervals (see **ACCHIS** example on Figure 4.6).
- vii. **ACCDIS**: *accumulation distribution* calculates the distribution of intervals accumulations.

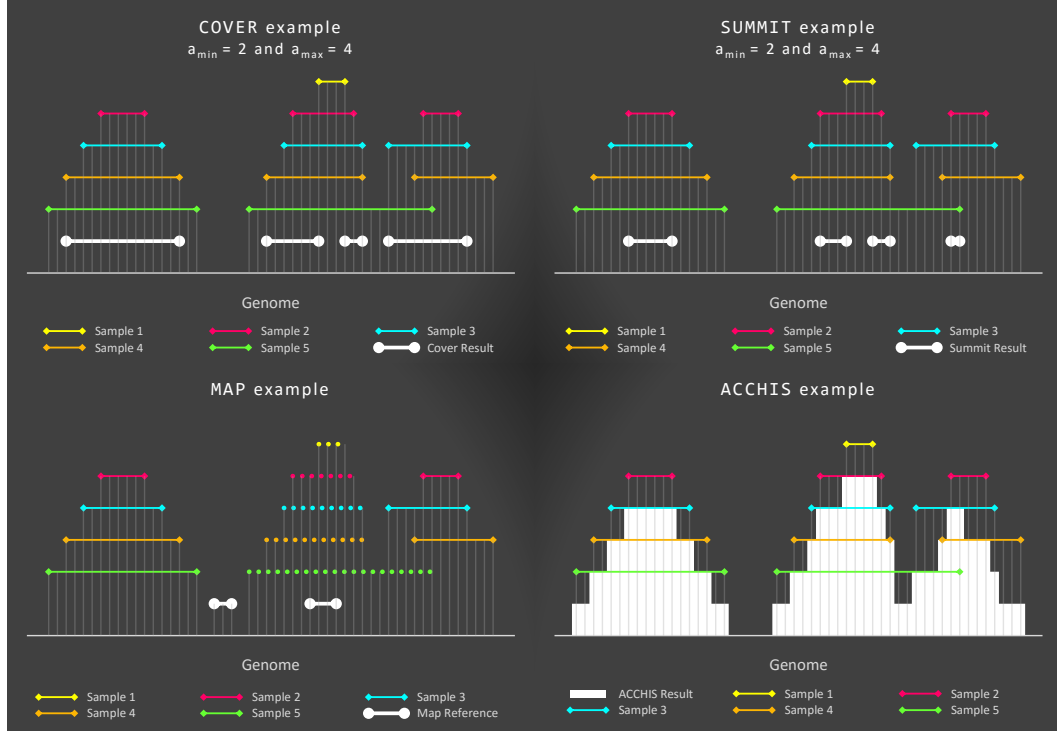

 Figure 4.6: Example of functions *COVER*, *SUMMIT*, *MAP*, and *ACCHIS*.

Table 4.1: Taxonomy of Di4 functions

Function	\mathcal{D}^+	\mathcal{D}^*	COF	FOF	UDF
MAP	✓		✓		✓
COVER	✓	✓		✓	✓
SUMMIT	✓	✓		✓	✓
MERGE		✓			
COMPLEMENT		✓	✓		
ACCHIS	✓			✓	
ACCDIS	✓			✓	

These functions cover COF/FOF and first/higher-order characteristics, and leverage on either \mathcal{D}^* , \mathcal{D}^+ , or both (see Table 4.1).

In general, a retrieval from Di4 is a two-step process. *First step* determines a set of snapshots that satisfy criteria from coordinate perspective. This step is the base of any retrieval from Di4 and it differs from one function to another. Di4 has incremental structure, such that a snapshot contains points to *causal* intervals only. Therefore, the first step intertwines a procedure called *reconstruction*, which reconstructs all intervals overlapping a snapshot. The reconstructed intervals are then passed to *second step* that applies a UDF on the intervals.

In following, present section first discusses the *reconstruction* procedure (Section 4.8.1), then functions *COVER*, *SUMMIT*, and *MAP* are discussed respectively in section 4.8.2, 4.8.3, and 4.8.4.

4.8.1 Data Structure Reconstruction

Di4 has incremental structure, such that each snapshot has pointers to causal intervals only, and pointers to non-causal intervals are given by neighbor snapshots. Pointers to the intervals are required to access metadata and execute UDFs. Therefore, the incremental structure necessitates *reconstruction* of intervals bookmarked by snapshots, which is finding pointers (i.e., $@I$) to all intervals overlapping a snapshot (including both causal and non-causal intervals). For instance, consider a snapshot B_b as $\mu_b = 9$, $\omega_b = 0$, and $\lambda_b = \{(L, @I)_i^j\}$. This snapshot bookmarks 10 intervals: one causal interval (i.e., I_i^j) and 9 non-causal intervals. The *reconstruction* algorithm determines the pointers of all the 10 intervals.

Design of data structure reconstruction algorithm In general, given a snapshot B_b , the *reconstruction* algorithm traverses neighbor snapshots “in-order” (i.e., B_{b+1}, B_{b+2}, \dots) for non-causal intervals pointers. For instance, referring to Figure 4.7, the snapshot B_5 bookmarks two intervals; one is the *causal event*, and one is a *non-causal event*. The right-end of R_3 is the *causal event*, and the *reconstruction* algorithm finds a pointer to R_3 in λ_5 . However, the algorithm has to traverse to B_6 to find a pointer to R_1 which is the *non-causal event* of B_5 (while it is the *causal event* of B_6). To determine pointers of all the intervals overlapping a snapshot (e.g., B_5), the *reconstruction* algorithm processes additional snapshots (e.g., B_6 for all the intervals of B_5). The number of additional snapshots to be processed for a given snapshot, depends on the bookmarked intervals and the given snapshot. For instance, while the *reconstruction* algorithm traverse one additional snapshot for B_5 , it traverses 4 additional snapshots for B_2 (i.e., B_3, B_4, B_5 , and B_6). The additional snapshots to be processed, motivates an optimal design. To clarify, consider the example of Figure 4.7, and let the query be: “find regions on domain where two intervals overlap, and sum the p-values”.⁶ The IR function finds snapshot B_2 that satisfies the criteria, because $|\lambda_2| + \mu_2 - \omega_2 = 2$; therefore, the intervals bookmarked by B_2 has to be reconstructed. The *reconstruction* algorithm traverses B_2, B_3, B_4, B_5 , and B_6 , and determines R_2 and R_1 as intervals bookmarked by B_2 . Then the IR function proceeds to B_3 , which does not satisfy the criteria; hence, the IR function returns R_1 and R_2 complying criteria between B_2 (i.e., $e_2 = \underline{R}_2$) and B_3 (i.e., $e_3 = \bar{R}_2$), and sums the p-values. The IR function proceeds to B_4 , which complies criteria; therefore, the intervals bookmarked by B_4 has to be reconstructed. The *reconstruction* algorithm determines R_3 as the *causal event*, and traverses B_5 and B_6 for the *non-causal event*, i.e., R_1 . Then the IR function proceeds to B_5 , which does not satisfy the criteria; therefore, the IR function returns R_1 and R_3 as two intervals complying criteria between B_4 (i.e., $e_4 = \underline{R}_3$) and B_5 (i.e., $e_5 = \bar{R}_3$), and sums the p-values. The IR function proceeds to B_6 , it does not comply criteria; and since B_6 is the last snapshot, the IR function concludes. Lets evaluate the two functions. The snapshots B_3 to B_6 are traversed two times, once by the IR function, and once by the *reconstruction* algorithm; respectively for intervals complying the query, and *reconstruction* of bookmarked intervals. Depending on the query

⁶This query can be answered using COVER function with minimum and maximum accumulation arguments set to 2, and a UDF for aggregation of p-values. The COVER function is described based on Di3 in Section 3.4 and is explained for Di4 in Section 4.8.2

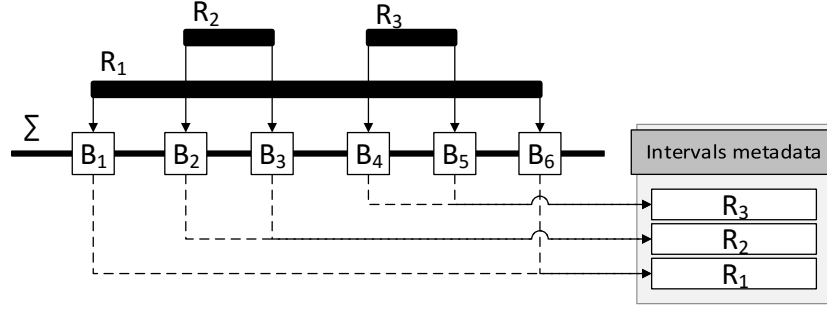


Figure 4.7: A sample region for data structure reconstruction.

and intervals, this procedure may traverse some snapshots many times, which it is a significant performance penalty. Therefore, to reduce the number of snapshots traversal to **one**, Di4 intertwined IR function traverse with *reconstruction* algorithm. This design, minimizes runtime complexity of algorithm at the cost of implementation complexity. However, Di4 aims for better performance, and complicated implementation is acceptable.

Now that we have briefly explained the *reconstruction* challenge, let describe the algorithm in details.

The complementary resolutions of Di4 have distinct structures and render particular aspects of indexed intervals, which make each resolution appropriate for different retrieval functions. The first resolution, \mathfrak{D}^+ , has incremental structure and provides minimal comprehensive bookkeeping of intervals, and immediate information of the coordinates of the intervals that makes it appropriate for COF retrievals. The second resolution, \mathfrak{D}^* , provides aggregated characteristics on sets of snapshots of first resolution that facilitate FOF retrievals from \mathfrak{D}^+ limiting search domain to a fraction of snapshots.

A block, $\mathbb{B}_\tau \in \mathfrak{D}^*$, bookmarks a *region* on \mathfrak{D}^+ containing contiguous snapshots spanning from B_α to B_δ and corresponding to $\underline{\mathbb{I}}_\tau$ and $\bar{\mathbb{I}}_\tau$ respectively (see Figure 4.8). The components γ_τ and α_τ of the block \mathbb{B}_τ denote respectively the cardinality and maximum accumulation of intervals between the snapshots B_α and B_δ (see Section 4.6). Blocks approximate snapshots, and can inform if they contain at least one subset that satisfy a criteria. However, a block does not inform the number of subsets, and exact position of snapshots.

A FOF retrieval iterates over *blocks* of second resolution, and if aggregated attribute satisfy criteria, then a subset of snapshots represented by the block also satisfy criteria. For instance, “find regions on domain where 10 intervals overlap”.⁷ To answer this query, a FOF function compares maximum accumulation (α_τ) of each block with 10, and if $\alpha_\tau > 10$, then a subset of snapshots represented by the block is bookkeeping 10 overlapping intervals.

⁷This query can be answered using *Cover* function with minimum and maximum accumulation arguments set to 10. The *Cover* function is described based on Di3 in Section 3.4 and is explained for Di4 in Section 4.8.2

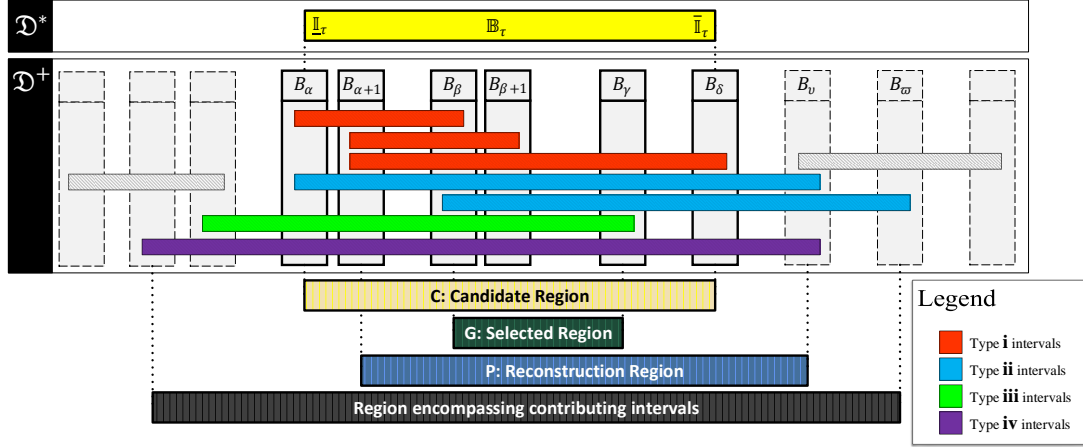


Figure 4.8: Candidate, Selected, and Reconstruction regions.

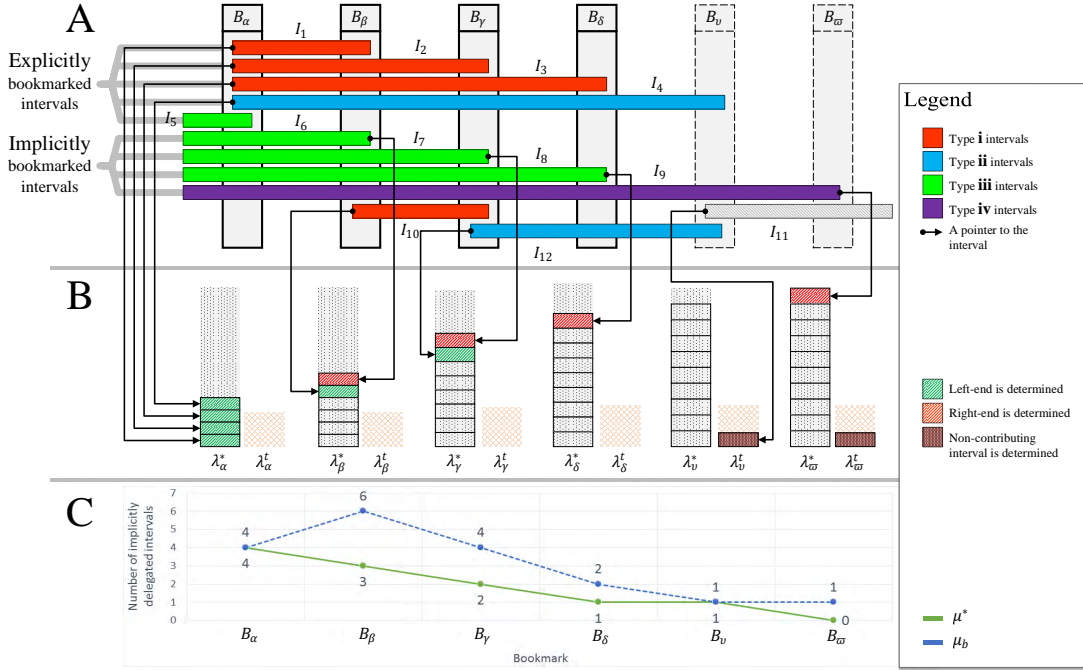


Figure 4.9: Reconstruction of intervals of different types.

In general, a block that satisfies FOF criteria defines a *candidate* region $C = \{B_\alpha \dots B_\delta\}$ on first resolution (see Figure 4.8). Let *selected* region be a subset of *candidate* region that satisfy criteria, in general: $G = \{B_\beta \dots B_\gamma\} \subset C$ for $\alpha < \beta < \gamma < \delta$. Note that, a *candidate* region may contain any number of *selected* regions.

The incremental structure of \mathcal{D}^+ necessitates *reconstruction* of intervals bookmarked by the snapshots of *selected* region(s). An interval is *reconstructed* when a pointer to the interval is determined; and Di4 holds pointer to an interval at the snapshots where the interval is the

causal event. The *reconstruction* of intervals of a candidate region, depends on the intervals intersection condition with B_α —the first snapshot of the candidate region. A snapshot B_α bookmarks an interval I either *explicitly* by a pointer to the interval in λ_α component if $\underline{I} = e_\alpha$ or $\bar{I} = e_\alpha$ (I is a *causal* interval); or *implicitly* by one unit incremental of μ_α component if $\underline{I} < e_\alpha < \bar{I}$ (I is a *non-causal* interval, refer to Section 4.5 for details). In general, an interval I_i contributing to the *candidate* region has four possible intersection conditions with B_α as follows (see Figure 4.9):

- i. $e_\alpha \leq \underline{I}_i \quad \wedge \quad \bar{I}_i \leq e_\delta$; Explicit
- ii. $e_\alpha \leq \underline{I}_i \leq e_\delta \quad \wedge \quad e_\delta < \bar{I}_i$; Explicit/Implicit
- iii. $\underline{I}_i < e_\alpha \quad \wedge \quad e_\alpha < \bar{I}_i \leq e_\delta$; Implicit
- iv. $\underline{I}_i < e_\alpha \quad \wedge \quad e_\delta < \bar{I}_i$; Implicit

The intervals of type (i) and (ii) occur at e_α or a following coordinate, and end subsequently (see Figure 4.9). Therefore, while traversing the *candidate* region, such intervals are explicitly bookmarked by the snapshot that corresponds to their left-end. Therefore, such intervals are reconstructed at first encounter. However, the first snapshot that bookmarks an interval of type (iii) and (iv) is B_α , which is implicitly bookkeeping such intervals through their cardinality given by μ_α component (see Figure 4.9). Such intervals are bookmarked by a snapshot that corresponds to their right-end, hence reconstruction of such intervals requires further traversing the snapshots.

Let *reconstruction* region, P , be the set of contiguous snapshots to be traversed to reconstruct the intervals *implicitly* bookmarked at B_α or B_β (see Figure 4.8). The left-end of *reconstruction* region is $\underline{P} = B_{\alpha+1}$ when the function traverses *candidate* region (i.e., running a FOF), or $\underline{P} = B_{\beta+1}$ if it does not (i.e., running a COF). The right-end of *reconstruction* region is $\bar{P} = B_v$ where all the *implicitly* bookmarked intervals by B_α or B_β are reconstructed (i.e., $(\forall I_i | \underline{I}_i < \underline{P} < \bar{I}_i) \rightarrow \bar{I}_i \leq e_v$).

Reconstruction Algorithm

The algorithm reconstructs the *explicitly* and *implicitly* bookmarked intervals traversing the *reconstruction* region using temporary variables μ^* and λ^* . Let μ^* be the number of implicitly bookmarked intervals of type (iii) and (iv), and λ^* be the set of reconstructed intervals while traversing the *reconstruction* region. The traversal of *reconstruction* region is intertwined with retrieval functions traversing the *candidate* or *selected* region(s).

The algorithm starts iteration by initializing μ^* and λ^* based on the corresponding components of the first snapshot of iteration (see Figure 4.9). The first snapshot is $B_{\alpha'} \leftarrow B_\alpha$ if traversal starts on *candidate* region ($C = \{B_\alpha \dots B_\delta\}$, see Figure 4.8), and $B_{\alpha'} \leftarrow B_\beta$ if a *selected* region is the start point ($G = \{B_\beta \dots B_\gamma\} \subset C$, see Figure 4.8). Accordingly, $\mu^* \leftarrow \mu_{\alpha'}$, and λ^* is initialized

with the set of the pointers of intervals starting at $B_{\alpha'}$ that is: $\lambda^* \leftarrow \{@I_i \mid @I_i \in \lambda_{\alpha'} \wedge \varphi_i = L\}$ (see Figure 4.9). Note that, the intervals *ending* at $B_{\alpha'}$ (i.e., $\bar{I}_i = e_{\alpha'}$; for instance I_5 on Figure 4.9) do not contribute to *reconstruction* region, and hence are avoided.

The algorithm proceeds traversing the *reconstruction* region and updating μ^* and λ^* . The tuples $(\varphi, @I)$ of each snapshot are either bookkeeping an interval that is *implicitly* bookmarked at $B_{\alpha'}$ (e.g., I_7 at B_β on Figure 4.9), or correspond to the right-end of a reconstructed interval (e.g., I_2 at B_β on Figure 4.9). In general, possible conditions of bookmarked intervals, and updates on λ^* and μ^* variables accordingly, are defined as follows:

a. Reconstruct *explicitly* bookmarked intervals

$\varphi_i = L$: the tuple is bookkeeping the left-end of an interval of type (i) or (ii). Therefore, $@I_i$ is added to λ^* ; for instance, referring to Figure 4.9 I_6 and I_8 at snapshots B_β and B_γ respectively.

b. Reconstruct *implicitly* bookmarked intervals

$\varphi_i = R \wedge @I_i \notin \lambda^*$: the tuple is bookkeeping the right-end of an interval of type (iii) or (iv), which reconstructs an interval *implicitly* bookmarked at $B_{\alpha'}$. Accordingly, $@I_i$ is added to λ^* and μ^* is decremented one unit ($\mu^* \leftarrow \mu^* - 1$). For instance, I_7 and I_9 respectively at snapshots B_β and B_γ on Figure 4.9.

c. Does not reconstruct a new interval

$\varphi_i = R \wedge @I_i \in \lambda^*$: the tuple is bookkeeping the right-end of an interval of type (i) or (ii), such that, the left-end is observed traversing former snapshots; therefore no update on λ^* and μ^* is required. For instance, I_1 at B_β or I_2 at B_γ on Figure 4.9.

The *reconstruction* algorithm terminates when two conditions are met; first, all snapshots of *candidate* or *selected* region are traversed. Second, all the *implicitly* bookmarked intervals are reconstructed (i.e., $\mu^* = 0$). Therefore, having processed the last snapshot of *candidate* or *selected* region, the condition $\mu^* = 0$ is evaluated and the algorithm is terminated upon confirmation. However, $\mu^* \neq 0$ necessitate the extension of traversal to reconstruct missing intervals (i.e., intervals of type (iv), see I_9 on Figure 4.9). For instance, referring to Figure 4.9, B_δ is the last snapshot to be traversed, however, at this snapshot $\mu^* = 1$ (i.e., I_9 is not reconstructed yet). In such conditions, the algorithm continues traversing the snapshots and updating temporary variables using $(\varphi, @I)_i$, and a secondary temporary list of λ^* 's as follows:

a. Reconstruct *explicitly* bookmarked intervals

$\varphi_i = R \wedge @I_i \in \lambda^*$: the tuple is bookkeeping the right-end of an interval of type (iii), which does not necessitate updating neither λ^* nor μ^* (e.g., I_4 at B_ρ of Figure 4.9).

b. Reconstruct *implicitly* bookmarked intervals

$\varphi_i = R \wedge @I_i \notin \lambda^* \wedge @I_i \notin \lambda^*$: such tuples correspond to the right-end of intervals of type (iv) that reconstruct intervals *implicitly* bookmarked by snapshots of *candidate* region. Therefore, $@I_i$ is added to λ^* , and μ^* is decremented one unit (e.g., I_9 at B_ω of Figure 4.9).

c. **Left-end of intervals not contributing to *candidate* region**

$\varphi_i = L$: the pointer of such intervals are added to λ_{temp} . For instance, I_{11} at B_v on Figure 4.9.

d. **Right-end of intervals not contributing to *candidate* region**

$\varphi_i = R \wedge @I_i \in \lambda_{\text{temp}}$: the pointer of such intervals are removed from λ_{temp}

The algorithm iterates until μ^* reaches zero.

Algorithm 7 Reconstruction stack: Open

```

1:  $G^* \leftarrow \{\text{left}, \text{right}, \mu, \lambda\}$ 
2:  $\lambda_{\text{temp}} \leftarrow \{\lambda\}$ 
    $\triangleright G^*$  and  $\lambda_{\text{temp}}$  are temporary variables that are accessible by all procedures of
   reconstruction.
3: procedure OPEN( $B_b$ )
4:    $\mu_{\text{new}} \leftarrow \mu_b - |\lambda_{\text{temp}}|$ 
5:    $\lambda_{\text{new}} \leftarrow \{(T, @I) \mid (\varphi, @I) \in \lambda_{\text{temp}}\}$ 
6:    $\lambda_{\text{temp}} \leftarrow \emptyset$ 
7:   for each  $\lambda \in \lambda_b$  do
8:     if  $\varphi = T$  then
9:        $\lambda_{\text{new}} \leftarrow \lambda_{\text{new}} \cup \lambda$ 
10:    else
11:      UPDATE $G^*$ ( $@I$ )
12:    if  $|G^*| > 0$  then
13:      for each  $\lambda \in G^*_{|G^*|}$  do
14:        if  $\varphi = T$  then
15:           $\lambda_{\text{new}} \leftarrow \lambda_{\text{new}} \cup (T, @I)$ 
16:           $\mu_{\text{new}} \leftarrow \mu_{\text{new}} - 1$ 
17:      CONCLUDE()
18:    $G^* \leftarrow G^* \cup (b, e_b, \mu_{\text{new}}, \lambda_{\text{new}})$   $\triangleright$  right-end value is temporarily set to  $e_b$ 

```

Algorithm 8 Reconstruction stack: Close

```

1: procedure CLOSE( $B_b$ )
2:   set right element of  $G^*_{|G^*|}$  to  $e_b$ 
3:   for each  $\lambda \in \lambda_b$  do
4:     if  $\varphi = T$  then
5:        $\lambda_{\text{temp}} \leftarrow \lambda_{\text{temp}} \cup \lambda$ 
6:     else
7:       UPDATE $G^*$ ( $@I$ )

```

Termination Analysis

The algorithm terminates upon validation of two conditions as follows:

Algorithm 9 Reconstruction stack: Update

```

1: procedure UPDATE( $B_b$ , regionType)
2:   if regionType = selected then
3:     for each  $\lambda \in \lambda_b$  do
4:       if  $\varphi = T$  then
5:          $G_{|G^*|}^* \leftarrow G_{|G^*|}^* \cup \lambda$ 
6:       else
7:         UPDATE $G^*$ (@ $I$ )
8:   else
9:     for each  $\lambda \in \lambda_b$  do
10:      if  $\varphi = T$  then
11:         $\lambda_{\text{temp}} \leftarrow \lambda_{\text{temp}} \cup \lambda$ 
12:      else
13:        if @ $I \in \lambda_{\text{temp}}$  then
14:           $\lambda_{\text{temp}} \leftarrow \lambda_{\text{temp}} \setminus \lambda$ 
15:        else
16:          UPDATE $G^*$ (@ $I$ )

```

Algorithm 10 Reconstruction stack: Update G^*

```

1: procedure UPDATE $G^*$ (@ $I$ )
2:   for each  $G \in G^*$  do
3:     if @ $I \in G_\lambda$  then                                      $\triangleright G_\lambda$  denotes  $\lambda$  component of  $G$  tuple.
4:       set corresponding  $\varphi$  to  $F$ 
5:     else
6:        $G_\lambda \leftarrow G_\lambda \cup (F, @I)$ 
7:        $G_\mu \leftarrow G_\mu - 1$ 

```

Algorithm 11 Reconstruction stack: Conclude

```

1: procedure CONCLUDE
2:   for each  $G \in G^*$  do
3:     if  $G_\mu = 0$  then
4:       call UDF and pass: left, right, { $@I \mid @I \in G_\lambda$ }
5:       remove  $G$ 

```

1. **Traversal reaches the last snapshot**

The last snapshot is B_δ when traversing the *candidate* region, or B_γ if traversing a *selected* region.

2. $\mu^* = 0$

Each snapshot bookmarks intervals *implicitly* and *explicitly*. In a region traversal (e.g., candidate region), the implicitly bookmarked intervals are either common with the first snapshot, or are disclosed in preceding snapshots. Therefore, the value of μ^* which is initialized as $\mu^* \leftarrow \mu_{\alpha'}$, decreases at reconstruction of an *implicitly* bookmarked interval. The traversal continues until $\mu^* = 0$. Given that Di4 data structures ensures existence of

two snapshots bookkeeping the two ends of a interval; therefore, the right-end of any implicitly bookmarked interval is determined. In general, traversing $\{B_\alpha \dots B_\beta \dots B_\gamma \dots B_\delta\}$ the following conditions are defined:

- a. Let I_i be an implicitly bookmarked interval at B_α ; by definition, $\exists! B_x, x > \alpha \mid e_x = \bar{I}_i$.
- b. Let I_j be an implicitly bookmarked interval at B_γ ; by definition, $\exists! B_y, y < \gamma \mid e_y = \underline{I}_j$. Accordingly, following conditions are possible:
 - if $y > \alpha$ then $\exists! B_\beta \mid e_y = e_\beta$; the interval is reconstructed at prior snapshots.
 - if $y = \alpha$ then $e_y = e_\alpha$; the interval is reconstructed at prior snapshots (e.g., I_3 on Figure 4.9).
 - if $y < \alpha$ then $\exists! B_\delta, \delta > \gamma \mid e_\delta = \bar{I}_j$; requires further traversal (e.g., I_9 on Figure 4.9).

4.8.2 Cover

The function COVER identifies co-occurrence of Genomic intervals with a quantitative measure, which yields a collection of consecutive snapshots whose referenced intervals are of specific accumulation (see COVER example on Figure 4.6). Let a_{\min} and a_{\max} denote respectively the minimum and maximum accumulation criteria, and a_b be the accumulation of intervals at B_b (i.e., $a_b \leftarrow \mu_b + |\lambda_b| - \omega_b$). The snapshot B_b is evaluated as complying accumulation criteria if: $a_{\min} \leq a_b \leq a_{\max}$. Accordingly, the definition of COVER is given by Definition 4.8.2.

Definition 4.8.1: COVER

The syntax of COVER function, f_{cover} , that takes arguments \mathfrak{D}^* , \mathfrak{D}^+ , a_{\min} , a_{\max} , and $f(T)$ is defined as follows:

$$f_{\text{cover}}(a_{\min}, a_{\max}, \mathfrak{D}^*, \mathfrak{D}^+, f(T)) = \{R_1, \dots, R_j, \dots, R_J\}$$

such that:

$$R_j = [e_\alpha, e_\delta]$$

where:

$$a_{\min} \geq a(B_{\alpha-1}) \text{ or } a_{\max} \leq a(B_{\alpha-1})$$

$$a_{\min} \leq a(B_\alpha) \leq a_{\max}$$

$$a_{\min} \leq a(B_\beta) \leq a_{\max} \text{ for } \beta \in \mathbb{N}_0$$

$$a_{\min} \leq a(B_\delta) \leq a_{\max} \text{ and } \beta < \delta$$

$$a_{\min} \geq a(B_{\delta+1}) \text{ or } a_{\max} \leq a(B_{\delta+1})$$

Motivation

Genome-wide binding profiles are increasingly becoming available for wide range of DNA-protein interactions; however, dissecting vast majority of complex regulatory networks requires consideration of co-occurrences of different protein bindings (e.g., co-occurrence of *cis*-regulatory modules with Transcription Factors [123] [124]). The quantity of co-occurring intervals is the first measure to disclose such biological characteristics. This requirement motivated the definition of COVER function that identifies regions with specific co-occurrence of intervals.

Intervals and their co-occurrences are widely studied in the spatial/temporal data field. Since the first definition of frequent neighboring co-occurrences in spatial data by Morimoto et al. [125], various methods are published targeting static and dynamic events (e.g., [126], [127]). In general, the methods are defined in four layers as follows:

- i. **Data representation:** the intervals are commonly organized in (rectangular/triangular/polygonal) grids-based (e.g., [128]) and clusters-based (e.g., [129]) representations.
- ii. **Indexing** modules are used by some methods to improve the data access performance (e.g., spatio-temporal trajectory indexing technique [130], [131], and [132]; see also [133] and two extensive surveys [134], [135]).
- iii. **Candidate generation:** candidates are collections of co-localization patterns and are identified using variety of different techniques such Apriori-based [136] (e.g., [128], [137]), clustering-based (e.g., [138], [129], and [139]), or hashing-based (e.g., disjoint cubes hashing [140]).
- iv. **Co-occurrence patterns identification**, which are defined upon candidates using approaches such as Apriori-gen-based [136].

In general, patterns are defined leveraging on candidates (i.e., iii.) and the contributing (e.g., [141], [137]), or no-contributing (e.g., *false candidates* [128]) instances. The co-occurring intervals are commonly identified by joining co-occurrence patterns and their instances (e.g., [142]). However, the join operation is the hot-spot of join-based approaches ([143]) that motivated *partial-join* [141] and *join-less* [144] approaches. Ultimately, Han et al. proposed a method to mine the co-occurrence patterns without generating candidate regions ([145], [146]).

Intervals possibly represent events with mixed feature types. For instance, the location of an airplane at a given time can be bookmarked using a spatio-temporal interval. Mining the co-occurrence patterns of intervals who are neighbors in a multi feature space, motivated mixed-drove co-occurrence pattern mining [147][148]. Such patterns are commonly identified in multiple steps, which is by identifying mixed-drove candidate patterns on one feature, and pruning-out on other features (e.g., [149], [150], [151]). Celik et al. [152] inter-twined the candidate pattern identification with filter in a 10-step algorithm to improve performance.

Algorithm 12 Cover

```

1: procedure COVER( $a_{\min}, a_{\max}, \text{UDF}$ )
2:   get a reconstruction stack initialized with UDF
3:    $a_{\text{tag}} \leftarrow -1$ 
4:   for each  $\mathbb{B}_\tau \in \mathcal{D}^*$  do
5:     if  $a_{\min} \leq \alpha_\tau$  then
6:       reset reconstruction
7:       for each  $B_b \mid \underline{l}_\tau \leq e_b \leq \bar{l}_\tau$  do
8:          $a \leftarrow \mu_b + |\lambda_b| - \omega_b$ 
9:         if  $a_{\text{tag}} = -1 \wedge a_{\min} \leq a \leq a_{\max}$  then
10:           $a_{\text{tag}} \leftarrow a$ 
11:          start reconstruction by  $B_b$ 
12:        else
13:          if  $a_{\text{tag}} \neq -1 \wedge (a < a_{\min} \vee a > a_{\max})$  then
14:            close reconstruction by  $B_b$ 
15:             $a_{\text{tag}} \leftarrow -1$ 
16:          update reconstruction by  $B_b$ 
17:        conclude reconstruction

```

Genomics challenge the current approaches by introducing multi-feature heterogeneous intervals. For instance, a ChIP-seq peak is an **interval** representing significance of a DNA-protein-antibody interaction using a p-value (a numerical attribute). An gene is also represented by an **interval** associated with a gene ID (a string attribute). The least common characteristic between such intervals is the coordinates, which is indexed by Di4. Genomics is interested in both co-occurrence (coordinates) and mixed-feature patterns (coordinates and additional features). The co-occurrence patterns are commonly defined in terms of accumulation of intervals with determined criteria (e.g., min/max accumulation). However, mixed-feature patterns are application-driven where the targeted attributes and corresponding criteria varies between applications.

To address this requirement, Di4 defines the COVER function which identifies density-based co-occurrence patterns on coordinates, and incorporates user-defined application-oriented pattern finding method on additional attributes through $f(T)$. The function $f(T)$ is a UDE, and could also aggregate any of the attributes of T intervals.

Algorithm

The second resolution, \mathcal{D}^* , is an abstract representation of first resolution (\mathcal{D}^+); and \mathcal{D}^+ is an abstraction of input samples. The coordinates of intervals as primary attributes, are indexed in \mathcal{D}^+ , and additional attributes of intervals for a group of snapshots are aggregated in \mathcal{D}^* . The aggregation of non-indexed attributes facilitates retrievals on such attributes by limiting function's domain to a fraction of snapshots whose aggregated attribute complies function criteria. The COVER function identifies co-occurrence patterns defined as accumulation of

a specific number intervals, which is a non-indexed property. However, second resolution aggregates this attribute for a group of snapshots, and necessitates a traversal on first resolution for identification of particular accumulation and contributing intervals. Therefore, COVER leverages on \mathfrak{D}^* and aggregated accumulation to minimize the number of snapshots to be processed with respect to the function criteria.

The COVER algorithm leverages on blocks ($\mathbb{B}_\tau \in \mathfrak{D}^*$) and a comparison between α_τ and a_{\min} to avoid unnecessary traversal on \mathfrak{D}^+ . Such that, if $a_{\min} \leq \alpha_\tau$ then snapshots spanning from B_α to B_δ (see Figure 4.8) are subject to COVER by set(s) of contiguous snapshots satisfying a_{\min} and a_{\max} criteria. These snapshots form *candidate* region, and are traversed in determination of particular co-occurrence patterns and contributing intervals, defined as *selected* region(s) ($D = \{[e_\beta, e_\gamma] \mid \forall B_b : \beta \leq b \leq \gamma \Rightarrow a_{\min} \leq a_b \leq a_{\max}\}$). However, $a_{\min} > \alpha_\tau$ implies insufficient accumulation from B_α to B_δ , and therefore a traverse on such snapshots is redundant and avoided accordingly.

The algorithm is parallelized on \mathfrak{D}^* , and each thread processes a subset of blocks ($[\mathbb{B}_{\tau'}, \mathbb{B}_{\tau''}]$). The process executed by each of the threads on a subset of blocks is described as follows.

The algorithm iterates over $[\mathbb{B}_{\tau'}, \mathbb{B}_{\tau''}]$, and for every block \mathbb{B}_τ it evaluates $a_{\min} \leq \alpha_\tau$; and upon confirmation, it initiates an iteration over snapshots $[B_\alpha, B_\delta]$ such that $e_\alpha = \mathbb{I}_\tau$ and $e_\delta = \mathbb{I}_\tau$ as COVER *candidate* region (C). The iteration over C for determination of COVER *selected* region(s) (G) is inter-twined with reconstruction algorithm on reconstructing of contributing intervals (see Section 4.8.1). Traversing C , a snapshot $B_b \in C$ is processed for two objectives as follows:

A. Identification of *selected* regions:

The condition $a_{\min} \leq a_b \leq a_{\max}$ is evaluated, and confirmation yields $B_b \in G$. By definition, if $B_{b-1} \notin G$ then B_b is the left-end of G . However, condition rejection has two consequences; either B_b is the right-end of G if the left-end was determined, or B_b is avoided as a snapshot bookkeeping intervals of improper accumulation. Ultimately, having determined both ends of *selected* region, the reconstructed intervals are passed to $f(T)$ to aggregate additional attribute(s), or apply an application-driven pattern identification.

B. Reconstruction of contributing intervals: (see Section 4.8.1 for details)

The reconstruction procedure depends on whether a snapshot is a member of candidate region or not; accordingly:

- $B_b \in C$

The reconstruction of every implicitly bookmarked interval I_i by B_b is performed using λ^* . If $\underline{I}_i = e_b$ then a tuple as $(L, @I)_i$ is added to λ^* . However, if $\bar{I}_i = e_b$, then I_i could be the right-end of an interval whose left-end was implicitly bookmarked, or an explicitly bookmarked interval. The condition is evaluated using λ^* as:

if $(L, @I)_i \in \lambda^*$ then update λ^* with $(R, @I)_i$.

if $(L, @I)_i \notin \lambda^*$ then add $(R, @I)_i$ to λ^* and decrement μ^* (i.e., $\mu^* \leftarrow \mu^* - 1$).

- $B_b \notin C \wedge B_b \in P$ The *reconstruction* region is traversed to determine the right-end

of intervals that was implicitly bookmarked by B_α ; accordingly, every implicitly bookmarked interval I_i by B_b is reconstructed as follows:

- > $\bar{I}_i = e_b$, then the interval does not contribute to *selected* region, hence $(L, @I)_i$ is added to λ^t .
- > $\bar{I}_i = e_b$, and if $(L, @I)_i \in \lambda^t$ then it is removed; if $(L, @I)_i \in \lambda^*$ then it is update to $(R, @I)_i$; otherwise, $(R, @I)_i$ is added to λ^* and $\mu^* \leftarrow \mu^* - 1$.

Having processed the *candidate* region for all *selected* regions, and identified all contributing intervals by traversing the *reconstruction* region; the algorithm concludes iteration over $[B_\alpha, B_\delta]$, and proceeds to next block.

4.8.3 Summit

The function SUMMIT is a variation of COVER function, and determines region(s) of local maximum within COVER regions (see SUMMIT example on Figure 4.6). Let a_{\min} and a_{\max} denote respectively the minimum and maximum accumulation criteria, and a_b be the accumulation of intervals at B_b (i.e., $a_b \leftarrow \mu_b + |\lambda_b| - \omega_b$). The snapshot B_b is evaluated as complying accumulation criteria if: $a_{\min} \leq a_b \leq a_{\max}$. Accordingly, the definition of SUMMIT is given as in Definition 4.8.3.

Definition 4.8.2: SUMMIT

The syntax of SUMMIT function, f_{summit} , that takes arguments a_{\min} , a_{\max} , \mathfrak{D}^* , \mathfrak{D}^+ , and $f(T)$ is defined as follows:

$$f_{\text{summit}}(a_{\min}, a_{\max}, \mathfrak{D}^*, \mathfrak{D}^+, f(T)) = \{R_1, \dots, R_j, \dots, R_J\}$$

such that:

$$R_j = [e_\alpha, e_\delta]$$

where:

$$a_{\min} \geq a(B_{\alpha-1}) \text{ or } a_{\max} \leq a(B_{\alpha-1})$$

$$a(B_{\alpha-1}) < a(B_\alpha)$$

$$a_{\min} \leq a(B_\alpha) \leq a_{\max}$$

$$a_{\min} \leq a(B_\beta) \leq a_{\max} \text{ for } \beta \in \mathbb{N}_0$$

$$a_{\min} \leq a(B_\delta) \leq a_{\max} \text{ and } \beta < \delta$$

$$a_{\min} \geq a(B_{\delta+1}) \text{ or } a_{\max} \leq a(B_{\delta+1}) \text{ or } a(B_{\delta+1}) < a(B_\delta)$$

The SUMMIT function algorithm is similar to COVER function, with minor differences. The function algorithm is given by Algorithm 13.

Algorithm 13 Summit

```

1: procedure SUMMIT( $a_{\min}, a_{\max}, \text{UDF}$ )
2:   get a reconstruction stack initialized with UDF
3:    $a_{\text{tag}} \leftarrow -1$ 
4:    $a_{\text{previous}} \leftarrow 0$ 
5:   for each  $\mathbb{B}_\tau \in \mathcal{D}^*$  do
6:     if  $a_{\min} \leq \alpha_\tau$  then
7:       reset reconstruction
8:       for each  $B_b \mid \underline{l}_\tau \leq e_b \leq \bar{l}_\tau$  do
9:          $a \leftarrow \mu_b + |\lambda_b| - \omega_b$ 
10:        if  $a_{\text{tag}} < a \wedge a_{\text{previous}} < a \wedge a_{\min} \leq a \leq a_{\max}$  then
11:           $a_{\text{tag}} \leftarrow a$ 
12:          start reconstruction by  $B_b$ 
13:        else
14:          if  $a_{\text{tag}} > a \vee (a_{\text{tag}} < a \wedge (a < a_{\min} \vee a > a_{\max}) \wedge a_{\text{tag}} \neq -1)$  then
15:            close reconstruction by  $B_b$ 
16:             $a_{\text{tag}} \leftarrow -1$ 
17:             $a_{\text{previous}} \leftarrow a$ 
18:          update reconstruction by  $B_b$ 
19:        conclude reconstruction

```

4.8.4 Map

The MAP function executes a range query on coordinate attribute. In general, given a reference interval I^r , the MAP function finds all the intervals indexed in Di4 that overlap with I^r (see MAP example on Figure 4.6). This functionality is similar to classical interval-tree [57] operation. In addition to a reference, the function takes a UDF and passes the determined intervals to the UDF. The output of the UDF is then reported back as an attribute of I^r . For instance, a UDF may take all the intervals overlapping I^r and return their cardinality. The MAP function is generally defined in Definition 4.8.4 for reference interval I^r .

Algorithm

The MAP function takes a set of non-overlapping reference intervals and a UDF as input, and operates on first resolution of Di4 (\mathcal{D}^+). The algorithm of MAP function iterates over reference intervals $S^r = \{I_1^r, \dots, I_k^r, \dots, I_K^r\}$ and snapshots of Di4 to find indexed intervals overlapping each reference interval. For a reference interval I_k^r , the algorithm determines a set of snapshots $[B_\beta, B_\gamma]$ called *selected region* (G , see Section 4.8.1) such that $e_{\beta-1} < \underline{l}_k^r \leq e_\beta$ and $e_\gamma \leq \bar{l}_k^r < e_{\gamma+1}$. The incremental structure of Di4 necessitates *reconstruction* of intervals bookmarked by the snapshots of the *selected region*. The *reconstruction region* could possibly be longer than *selected region*. Therefore, the selected region of multiple reference intervals may overlap. Note that, the selected region of two consecutive reference intervals overlap *iff* there exist at least one indexed interval that overlaps both of the reference intervals. Therefore, to avoid traversing

snapshots multiple times (in case of overlapping *selected* regions), the reconstruction process is intertwined with *MAP* algorithm. In general the procedure of *MAP* function is as follows.

Definition 4.8.3: MAP

The syntax of MAP function, f_{map} , that takes arguments \mathfrak{D}^+ , $f(T)$, and a reference interval I^r , and it is defined as follows:

$$f_{\text{map}}(I^r, \mathfrak{D}^+, f(T)) = \{I_i\}$$

such that I^r and I_i satisfy conditions (ii), (iii), (iv), (v), or (vi) of interval heptachotomy defined in Section 4.4, which are:

$$\begin{aligned} \bar{I}^r &< \underline{I}_i \wedge \underline{I}_i \leq \bar{I}^r \leq \bar{I}_i \\ \underline{I}^r &< \underline{I}_i \wedge \bar{I}_i < \bar{I}^r \\ \underline{I}^r &= \underline{I}_i \wedge \bar{I}^r = \bar{I}_i \\ \underline{I}_i &< \underline{I}^r \wedge \bar{I}^r < \bar{I}_i \\ \underline{I}_i &\leq \underline{I}^r \wedge \bar{I}_i < \bar{I}^r \end{aligned}$$

Step 1. Run dichotomic search to find B_β .

Having determined B_β three conditions as follows are possible:

- i. if $e_\beta = \underline{I}_k^r$, then $\lambda^* \leftarrow \left\{ @I_i^j \mid (\varphi, @I)_i^j \in \lambda_\beta \wedge \varphi_i = L \right\}$ The temporary variable λ^* (see Section 4.8.1) is initialized with the pointers of intervals in λ_β that start at e_β .
- ii. if $\underline{I}_k^r < e_\beta < \bar{I}_k^r$, then $\lambda^* \leftarrow \left\{ @I_i^j \mid (\varphi, @I)_i^j \in \lambda_\beta \right\}$ The temporary variable λ^* is initialized with all the pointers to the intervals in λ_β .
- iii. if $\bar{I}_k^r \leq e_\beta$ then $\lambda^* \leftarrow \left\{ @I_i^j \mid (\varphi, @I)_i^j \in \lambda_\beta \wedge \varphi_i = R \right\}$ The temporary variable λ^* is initialized with the pointers of intervals in λ_β that stop at e_β

Step 2. Traverse *selected* region The algorithm traverses the snapshots between B_β exclusive and B_γ inclusive, and for each $(\varphi, @I)_i^j$ it does:

- . if $\varphi_i^j = L$ then $\lambda^* \leftarrow \lambda^* \cup @I_i^j$.
- . if $\varphi_i^j = R$ and $@I_i^j \notin \lambda^*$ then $\lambda^* \leftarrow \lambda^* \cup @I_i^j$ and $\mu^* \leftarrow \mu^* - 1$ (see Section 4.8.1).
- . if $\varphi_i^j = R$ and $@I_i^j \in \lambda^*$ then no further action is required.

Step 3. Finalize *selected* region

- . if $\mu^* = 0$ then all the contributing intervals to the reference region I_k^r are reconstructed, hence the search is completed. Then the intervals contributing to the reference region are passed to the UDF, and algorithm proceeds with I_{k+1}^r starting at *Step 1*.
- . if $\mu^* \neq 0$ then few of the intervals overlapping reference region I_k^r are not reconstructed yet, hence the algorithm proceeds traversing snapshots to reconstruct such intervals. In this traversal, the coordinate of each snapshot is compared with

the left-end of the reference interval I_{k+1}^r . If a snapshot overlaps the reference interval I_{k+1}^r , then the algorithm proceeds with *Step 2* for the reference interval I_{k+1}^r while reconstructing intervals for I_k^r . Note that, in this condition, the MAP algorithm traverses the *selected* region of both I_k^r and I_{k+1}^r , and *reconstruction* region of I_k^r only. This design of the algorithm minimizes the number of snapshots to be traversed.

5 Di4 and Di3 Performance Evaluation and Comparison

The present chapter evaluates Di3 and Di4 performance in comparison with common tools in bioinformatics. The chapter is organized as follows; the customization of Di3 and Di4 for genomics, and environment setup (including data and machines) for performance assessment are discussed in Section 5.1, Section 5.2 evaluates Di3 and Di4 performance, and finally Section 5.3 compares Di3 and Di4 performance to common tools of bioinformatics field under common scenarios.

5.1 Performance Evaluation Setup

For the performance evaluation, we customized Di3 and Di4 to the genomic domain by building *Di4 for Bioinformatics* (Di4B) at the business logic layer that initialize several independent Di3 and Di4 instances, one for each DNA chromosome and strand; the *Di4B command line interface* (Di4BCLI) client at presentation layer (see Figure 5.1) provides user interaction through a set of commands. These include: primitives for initializing the indexes, primitives for the operations, and primitives for setting indexing modes and degree of parallelization. In general, the Di4BCLI commands (listed in Table 5.1) have standard command argument structure, where the number of arguments varies between different commands. Having executed a command, its runtime is reported on console, and also saved in a user-defined log file.

The *Index* and *BatchIndex* primitives take a sample or a collection of samples as argument and, based on the indexing mode, they index the intervals respectively in single-pass or double-pass mode. Under double-pass indexing mode, the command *2Pass* (which takes no arguments) executes the second-pass of the indexing. The second resolution of Di3 is created/updated by the *2RI* command (that takes no arguments). The *Cover* and *Summit* commands take *minAcc*, *maxAcc*, *aggregate*, and *output* arguments, execute the functions with the parameters and export results to the output file. The *Map* command takes *reference*, *aggregate*, and *output* arguments, executes the function and exports results in the output file. The *Merge*, *Complement*, *AccHis*, and *AccDis* commands take *output* argument, execute the function and report results to the output file. The *GetIM* reports current setting for indexing

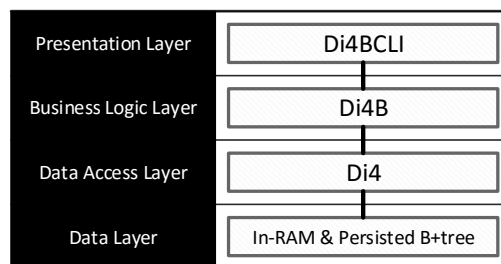


Figure 5.1: Application design of Di4, Di4 for Bioinformatics (Di4B), and Di4B Command Line Interface (Di4BCLI), that customize Di4 and Di3 for genomic domain.

mode, and *SetIM* takes a *mode* argument which is either *single* or *multi*, and sets indexing mode accordingly. Finally, *GetDP* reports current setting for degree of parallelization, and *SetDP* takes two numbers as *chr-degree* and *Di4/Di3-degree* of parallelization and updates the execution environment accordingly.

The performance of Di3 and Di4 are evaluated using samples downloaded from ENCODE which is a public repository.¹ The downloaded data are grouped in 9 datasets, as described in Table 5.2; the datasets vary in size, but are similar in interval accumulation distribution which are plotted in Figure 5.2. Data distribution may have a strong effect on the performance of an index. For instance, non-uniformly distributed data may accumulate a big load of information on some keys, while other keys may have lighter load; this is suboptimal because some keys are very expensive to process, while others are cheap. This affects also parallel execution, as some threads are busy for a very long time, while others are set free very early. The first step to avoid such draw backs is at design level, by making correct decisions, based on the nature of the data, for the key and value of the index.

The performance is assessed on two machines; a standard laptop to evaluate performance for small-scale processes, and a virtual private server (VPS) for large-scale execution. The specification of these two machines is given in Figure 5.3. Theoretical peak performance of each machine's processor is given in Giga Floating Point Operations Per Second (GFLOPS) which is calculated as follows:

$$\text{GFLOPS} = (\text{CPU speed in GHz}) * (\# \text{ of cores}) * (\text{CPU instructions per cycle})$$

The CPU speed and number of cores are obtained from the vendors website. The CPU instructions per cycle (IPC) parameter depends on Advanced Vector Extensions (AVX) feature of a CPU. The CPUs of machines we used for benchmarking belong to Intel® *Sandy Bridge* and *Ivy Bridge* architecture that benefit from AVX256 that hold 8 single precision values. Therefore, the IPC of these CPUs is as follows:²

¹<https://genome.ucsc.edu/ENCODE/>

²Source: http://media.wix.com/ugd/e53cc7_d9d115bb3548496481e893af130cf943.pdf

5.1. Performance Evaluation Setup

Table 5.1: Di3BCLI commands.

Command	Description
<i>Index</i>	Takes a filename and indexes all its regions.
<i>BatchIndex</i>	Takes a set of files specified using wild-card characters.
<i>2Pass</i>	Runs second-pass of indexing in double-pass indexing mode.
<i>2RI</i>	Indexes second resolution.
<i>Cover</i>	Executes COVER function and exports results.
<i>Summit</i>	Executes SUMMIT function and exports results.
<i>Map</i>	Executes MAP function and exports results.
<i>Merge</i>	Executes MERGE function and exports results.
<i>Complement</i>	Executes COMPLEMENT function and exports results.
<i>AccHis</i>	Determines <i>Accumulation histogram</i> and exports results.
<i>AccDis</i>	Determines <i>Accumulation distribution</i> and exports results.
<i>GetIM</i>	Reports current setting for indexing mode.
<i>SetIM</i>	Sets indexing mode to the specified one.
<i>GetDP</i>	Reports current setting for degree of parallelization.
<i>SetDP</i>	Sets degree of parallelization to the specified one.

Table 5.2: Datasets used for Di3 and Di4 benchmarking.

Dataset label	Sample count	Region count	Dataset size (MB)
<i>C1</i>	12	89,623	6.14
<i>C2</i>	22	258,406	17.40
<i>C3</i>	45	456,385	30.72
<i>B1</i>	90	1,407,493	92.16
<i>B2</i>	180	4,649,767	321.53
<i>A1</i>	500	28,392,674	1,382.40
<i>A2</i>	1,000	59,980,303	3,246.08
<i>A3</i>	1,500	94,997,460	4,986.88
<i>A4</i>	2,000	143,563,549	7,147.52

Machine label		M1	M2
Machine type		Amazon EC2	Laptop
Instance type		c3.8xlarge	
Processor	Physical Processor	Intel® Xeon® E5-2680 v2	Intel® Core™ i3-2310M
	# of Cores	10	2
	# of Threads	20	4
	Clock speed (GHz)	2.8	2.1
	IPC	8	8
	GFLOPS	224	33.6
RAM (GB)		60	8
SSD (MB/s)	Seq (R/W)	428.67 / 373.99	486.82 / 282.05
	4K (R/W)	21.37 / 38.32	21.78 / 41.75
	4K 64-Thread (R/W)	281.68 / 186.45	933.37 / 256.31

Table 5.3: Specification of machines used for performance evaluation.

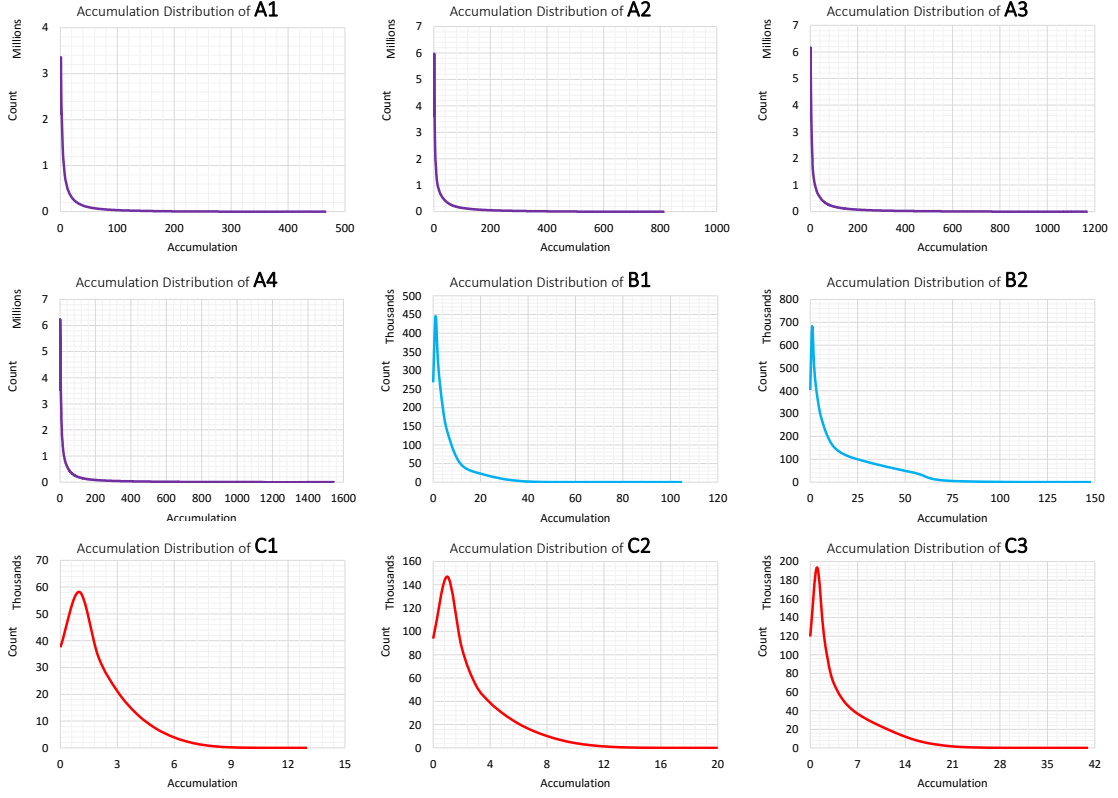


Figure 5.2: Interval/region accumulation distribution in datasets.

- IPC = 8 : double-precision FLOPs/cycle (4 AVX addition and multiplication).
- IPC = 16 : single-precision FLOPs/cycle (8 AVX addition and multiplication).

The machines have Solid-State Drive (SSD) storage device. Both Di3 and Di4 extensively use I/O operations when are run in persisted mode. Therefore, the storage performance of a machine is an important aspect that impacts the Di3 and Di4 performance. The storage devices of the machines are assessed for sequential read/write (i.e., the time it takes to read and write a 1GB file), random read/write of 4K blocks. Both Di3 and Di4 run in user-defined degree-of-parallelism; therefore, storage performance in a multi-threaded execution is an important parameter. Storage devices are tested for random read/write of 4K blocks in 64 threads. Results for sequential, random 4K, and parallel random 4K tests are given in Table 5.3.

5.2 Comparison between inverted index and incremental inverted index

The current section discusses the performance of Di3 and Di4, both for indexing intervals, and retrieval functions. In all the experiments of this section, we used **M1** machine (see Table 5.3), and **A1**, **A2**, **A3**, and **A4** datasets (see Table 5.2).

5.2. Comparison between inverted index and incremental inverted index

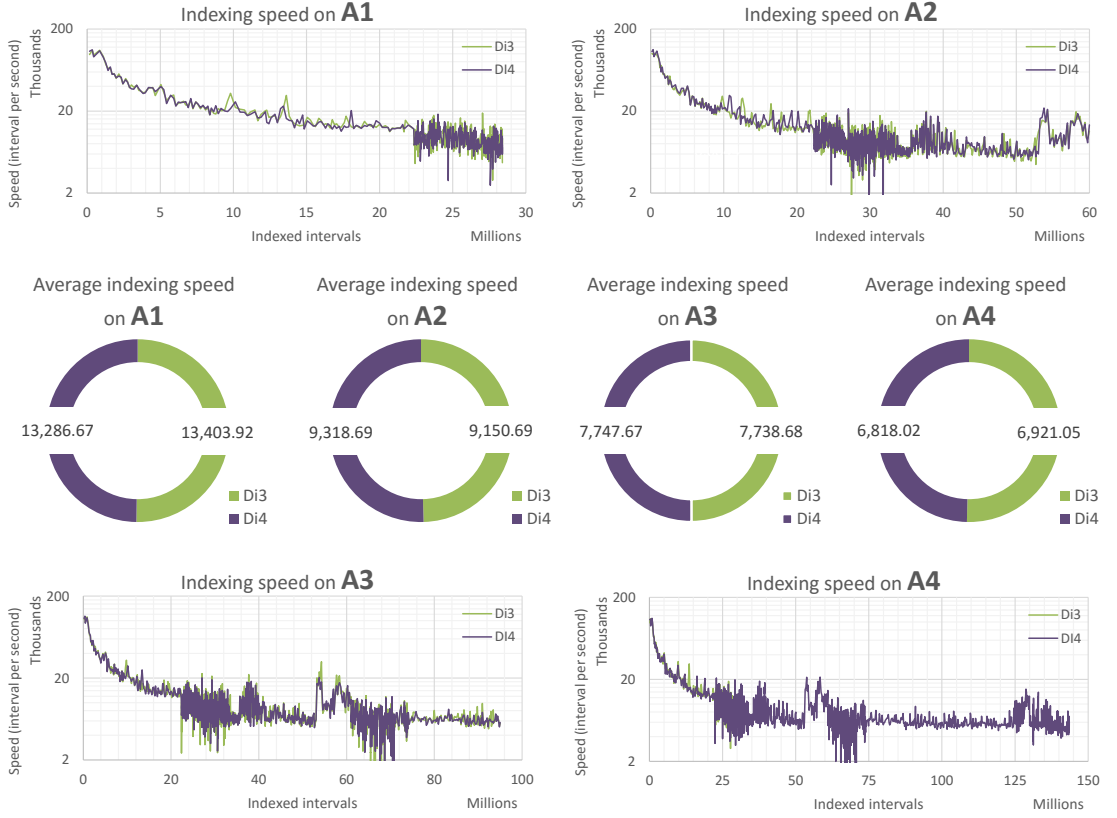


Figure 5.3: First pass indexing speed of Di3 and Di4 on four datasets.

5.2.1 Comparison of indexing speed

The indexing speed of Di3 and Di4 is evaluated for first-resolution (in double-pass indexing mode) and second-resolution, by executing Di3 and Di4 separately on the M1 machine (see Table 5.3) for indexing all the intervals of A1, A2, A3, and A4 datasets (see Table 5.2). The first step is to index intervals on the first resolution. Accordingly, we assess the first-pass of bookmarking intervals on first-resolution. The indexing speed (intervals per second) and average indexing speed of Di3 and Di4 for the first-pass is plotted separately for each dataset on Figure 5.3. In general, both methods execute similar actions at first step, which is creating snapshots, bookkeeping *causal* intervals by λ component of the snapshots, and inserting newly created snapshots to the data structure of first-resolution (i.e., inserting to key-value pair storage technology at physical layer, such as B+tree). Therefore, Di3 and Di4 first pass indexing complexity is similar, which is confirmed by the speed and average indexing speed plots on the four dataset (see Figure 5.3), and indexing elapsed time (see first-pass elapsed time on Figure 5.4).

The Di3 and Di4 indexing algorithms mainly differ in the second-pass of double-pass indexing, and indexing of second resolution. In general, at second-pass of indexing intervals, for the first-resolution, Di3 and Di4 perform similar procedures, which is reading snapshots created

during the first-pass, and updating each snapshot such that it complies the design of Di3 and Di4. The algorithms of Di3 and Di4 take similar time to read a snapshot from corresponding first-resolution, because the snapshots of both Di3 and Di4 after first-pass are bookkeeping only *causal* intervals, therefore, the snapshot are similar in size and hence take equal time to be de-serialized from persistence technology. However, the complexity of update is different between Di3 and Di4. The second-pass indexing algorithm of Di3 updates λ component of snapshots by adding new $(\varphi, @I)$ tuples ($\varphi = M$ for all newly inserted tuples). The number of newly added tuples depend on the accumulation of intervals (i.e., the more the intervals are accumulated, the more tuples to be added to each snapshot), and has impact on the complexity of update procedure (i.e., the larger the number of new tuples to be added, the longer it takes to update a snapshot). On the other hand, the second-pass indexing algorithm of Di4 updates first resolution by modifying the μ component of snapshots. In general, the second-pass indexing algorithm of Di3 increases the size of snapshots by new tuples added to λ , while the second-pass indexing algorithm of Di4 keeps the size intact and modifies the value of μ . Therefore, the second-pass of Di3 takes longer time than Di4, which is confirmed by the elapsed time of second pass plotted for four datasets on Figure 5.4.

The second-resolution data structure is common between Di3 and Di4, however, the complexity of creating it differs depending on whether it is created based on the first resolution of Di3 or Di4. The grouping and aggregation functions of second-resolution are independent from Di3 and Di4 bookmarking design. In general, the complexity of creating/updating blocks of second-resolution is independent from Di3 and Di4 design. However, the blocks of second-resolution are created based on the intervals bookmarked by snapshots. Therefore, the complexity of reconstructing bookmarked intervals, which differs between Di3 and Di4, impacts the second resolution indexing runtime (see Figure 5.4). Since reconstructing intervals from Di4 is relatively faster than Di3, creating second resolution based on Di4 is relatively faster than creating it based on Di3 (see Figure 5.4).

5.2.2 Comparison of index file size

The persisted data size of Di3 and Di4 are compared with each other, which includes snapshots and blocks organized respectively in first-resolution and second-resolution. The file size is a factor of the number and size of snapshots. The Di3 and Di4 both create a snapshot for each end of an interval, therefore, Di3 and Di4 create same number of snapshots. However, the snapshots of Di3 explicitly bookmark all overlapping intervals (i.e., inverted structure), while the snapshots of Di4 are bookkeeping only *causal* intervals explicitly (i.e., incremental inverted structure); therefore, the snapshots of Di3 are relatively larger than the snapshots of Di4. This makes the index file size of Di3 relatively larger than the file size of Di4 (see Figure 5.4).

Additionally, the growth rate of Di3 index file size is significantly faster than Di4 due to the inverted structure of Di3. Let explain this by an example; consider two consecutive snapshots B_i and B_j , where each is bookkeeping 1 causal interval, and n non-causal interval. In Di3, B_i

5.2. Comparison between inverted index and incremental inverted index

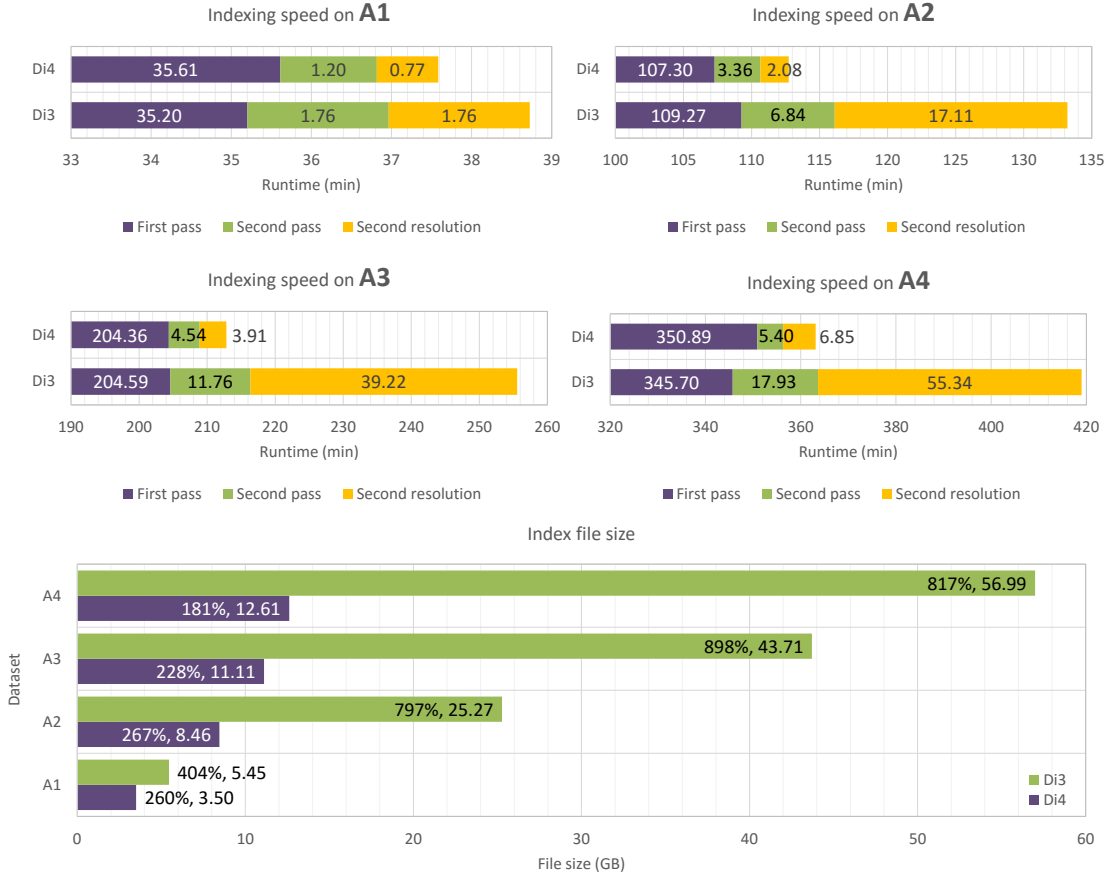


Figure 5.4: Comparison of Di3 and Di4 indexing elapsed time and index file size. The “Index file size” in addition to index file size, in data label, it includes the index file size proportional to input data size calculated as $(\text{index file size} / \text{input size}) * 100$.

and B_j have $n + 1$ tuples in λ components (i.e., $|\lambda_i| = n + 1$ and $|\lambda_j| = n + 1$), while in Di4, B_i and B_j have 1 tuple in λ components. Let B_k be a new snapshot created for indexing a new interval, such that $e_i < e_k < e_j$ (which creates B_i, B_k, B_j). The new snapshot in Di3 *explicitly* bookmarks the intervals bookmarked by B_j , which is by duplicating all the tuples in λ_j in λ_k such that $|\lambda_k| = |\lambda_j| + 1$. However, the new snapshot in Di4 *implicitly* bookmarks the intervals bookmarked by B_j by μ_k component. Therefore, with one new snapshot, the number of tuples in λ components in Di3 is: $|\lambda_i| + (2 * |\lambda_j|)$ while in Di4 it is: $|\lambda_i| + |\lambda_j| + 1$; this makes Di3 grow faster in file size compared to Di4 (see Figure 5.4).

5.2.3 Comparison of retrievals

The section benchmarks Di3 and Di4 for the IR functions COVER, SUMMIT, and MAP. The performance of Di3 and Di4 for each of the functions is discussed separately in the following.

Benchmark for Accumulation Operations

The Di3 and Di4 performance for COVER, SUMMIT, ACCHIS (accumulation histogram) and ACCDIS (accumulation distribution) operations are assessed. The function SUMMIT is a variation of COVER, similarly, the function ACCDIS is a variation of ACCHIS; therefore, their performance is at the same scale (see Figure 5.5). Both the accumulation histogram and distribution functions scan all snapshots; hence, their performance can be evaluated as the maximum time required for a full scan of Di3 and Di4, which is linear to dataset size and it differs between Di3 and Di4. In general, a linear scan is faster on Di4 compared to Di3 due to the incremental structure of Di4 snapshots, which are faster to de-serialize and exclude redundant information (see Figure 5.5).

The functions COVER and SUMMIT use both resolutions of Di3 and Di4 (i.e., first-resolution and second-resolution). As for accumulation histogram and distribution operations, the functions COVER and SUMMIT require linear scan of snapshots for regions of specific accumulation of intervals. However, the second resolution index prunes a percentage of linear scan based on `minAcc` and `maxAcc` parameters, and on their overlap with the accumulation distribution of data. The less effective pruning is expected with parameters set around the peak of accumulation distribution, therefore the COVER and SUMMIT functions are expected to be faster than the ACCHIS and ACCDIS with such choice of parameters.

The second resolution is common between Di3 and Di4; therefore, COVER and SUMMIT of Di3 have similar complexity to COVER and SUMMIT of Di4 for traversing second resolution. Di3 and Di4 differ in first-resolution; therefore, the COVER and SUMMIT functions of Di3 perform differently from the COVER and SUMMIT functions of Di4 in traversing the first-resolution. The snapshots of Di3 are larger in size compared to the snapshot of Di4; therefore, the de-serialization of Di3 snapshots is more expensive compared to Di4 snapshots. In contrast, a single snapshot of Di3 renders the information of all the intervals overlapping the position on domain to which the snapshot corresponds to, while reconstructing intervals overlapping a position on domain using Di4 snapshots may require processing additional snapshots.

In general, the penalties of Di3 and Di4 are: Di3 has larger snapshots that are expensive for de-serialization and duplicated information in a sequence of snapshots (which reduce traversal speed), while Di4 requires more snapshots to be processed. The superiority of one over the other depends on the input data and the retrieval function. The COVER and SUMMIT functions traverse a set of snapshots (determined by a block, see Section 3.4 and Section 4.8.2), and the traverse is faster with snapshots of Di4, because they are smaller in size compared to the snapshots of Di3, and exclude duplication. Additionally, the traverse required for reconstructing intervals is intertwined with the traverse of COVER and SUMMIT functions (see Section 4.8.1), therefore, the number of snapshots to be traversed (in addition to the ones required for COVER or SUMMIT) for reconstructing contributing intervals is minimized.

The performance is benchmarked with `minAcc` and `maxAcc` at the peak of accumulation distribution, e.g., `minAcc` = 80 and `maxAcc` = 100 for A2 (see Figure 5.2 for accumulation

5.2. Comparison between inverted index and incremental inverted index

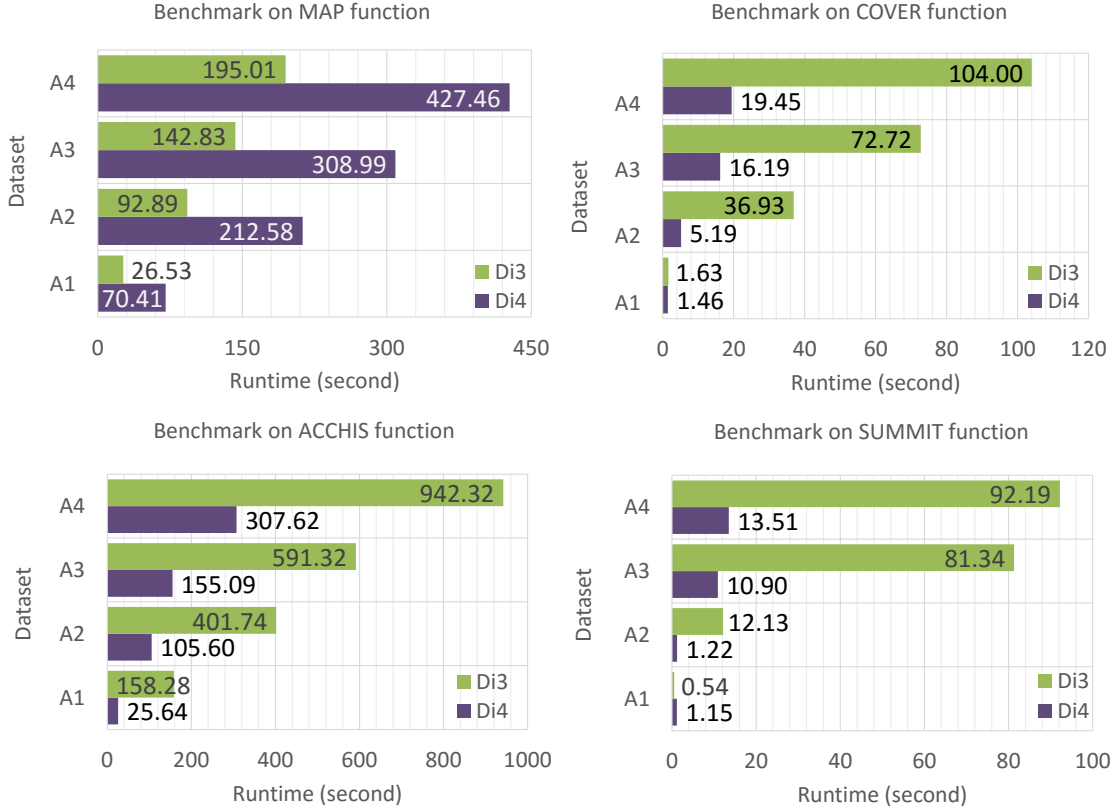


Figure 5.5: Benchmark of retrieval functions *COVER*, *SUMMIT*, and *MAP* on *A1*, *A2*, *A3*, and *A4* datasets, running on *M1* machine.

distribution). Figure 5.5 confirms two points; first, even with the peak of the parameter values the functions perform faster than full scan. Second, the *COVER* and *SUMMIT* functions of *Di4* are relatively faster than the *COVER* and *SUMMIT* functions of *Di3*.

Benchmark of MAP

The *MAP* operation directly operates upon coordinates and snapshots, and performs a dichotomic search to find regions that satisfy a given criteria. The *Di3* snapshots determined by the dichotomic search of *MAP* function renders all the contributing intervals, while the snapshots of *Di4* depend on neighbor snapshots to reconstruct the intervals overlapping reference regions. The number of neighbor snapshots to be processed additionally, depends on the distribution of indexed intervals and reference regions. The number may grow as large as it minimizes the advantage of a dichotomic search, and it takes longer time to process additional *Di4* snapshots than it takes to process limited number of *Di3* snapshots. In general, for a single reference region, $O(\log|\mathfrak{B}^+|)$ is the number of *Di3* snapshots to be processed, and $O(\log|\mathfrak{B}^+| + n)$ is the number of *Di4* snapshots for n neighbor snapshots. For a relatively small n , the *MAP* operation processes same number of snapshots on *Di3* and *Di4*; since *Di4* snapshots are faster to de-serialize and avoid duplicates, then *MAP* on *Di4* performs relatively faster than

on Di3. For a relatively large n , the MAP operation process more snapshots on Di4 than on Di3, whereas the cost of traversing additional snapshots is more than the de-serialization and duplicated information penalty of few Di3 snapshots. Accordingly, the MAP function may perform better on Di3 (with inverted snapshots and duplicate information) than Di4 (with incremental snapshots).

A benchmark of the MAP operation using a reference sample from the ENCODE repository, which includes 196,180 regions (8.9 MB in size) is given in Figure 5.5. The figure shows excellent scalability with respect to growth in data size for both Di3 and Di4. Note that the reference sample is also an ENCODE narrow peak sample; hence, its intervals are mostly co-localized with the indexed intervals. Therefore, a very big percentage of indexed data overlaps with the reference intervals. In this condition, n is relatively large. Therefore, Di3 performs significantly faster than Di4 on the four datasets.

5.2.4 Performance evaluation based on different degrees of parallelization

We benchmarked Di4 by exploiting two levels of parallelism: chromosome level parallelism (i.e., executing operations on multiple chromosomes concurrently) and Di4 level of parallelism (i.e., each chromosome is further divided into multiple sections or *bins*, and multiple threads process the resulting sections). The evaluation is using 100 ENCODE narrow peaks containing 18,410,405 intervals, and it runs on M2 machine. The results are plotted in Figure 5.6 in *performance coefficient* calculated as follows:

$$x'_n = \left(\frac{\min(x_1, x_2, \dots, x_n, \dots, x_N)}{x_n} \right) * 100$$

The lowest value (i.e., fastest operation) is transformed to 100, and all other values are scaled between 0 and 1 based on their difference with lowest value.

Note that, M2 machine has 2 cores and 4 *logical processor*, hence a degree-of-parallelization 16x16 (256 threads) is beyond the M2s' logical processor count. Di4 indexes data relatively faster when the degree-of-parallelism is even between two levels of parallelism and in total are equal to M2s' logical processor count. Di4 is penalized with *locking* overhead in favor of enabling multi-threading. The degree-of-parallelism should balance between logical processor counts and locking overhead, such that locking overhead is not compensated if Di4 is executed in single-thread. This aspect is highlighted in Figure 5.6 by MAP, COVER, and SUMMIT functions plot; where Di4 runs relatively faster when degree-of-parallelism is close to logical processor count. The functions ACCHIS and ACCDIS linearly scan Di4 and do not run in parallel at Di4 level; and only one thread runs even if higher parallelization degree. Therefore these functions run relatively faster when Di4 runs single threaded (see Figure 5.6).

5.2. Comparison between inverted index and incremental inverted index

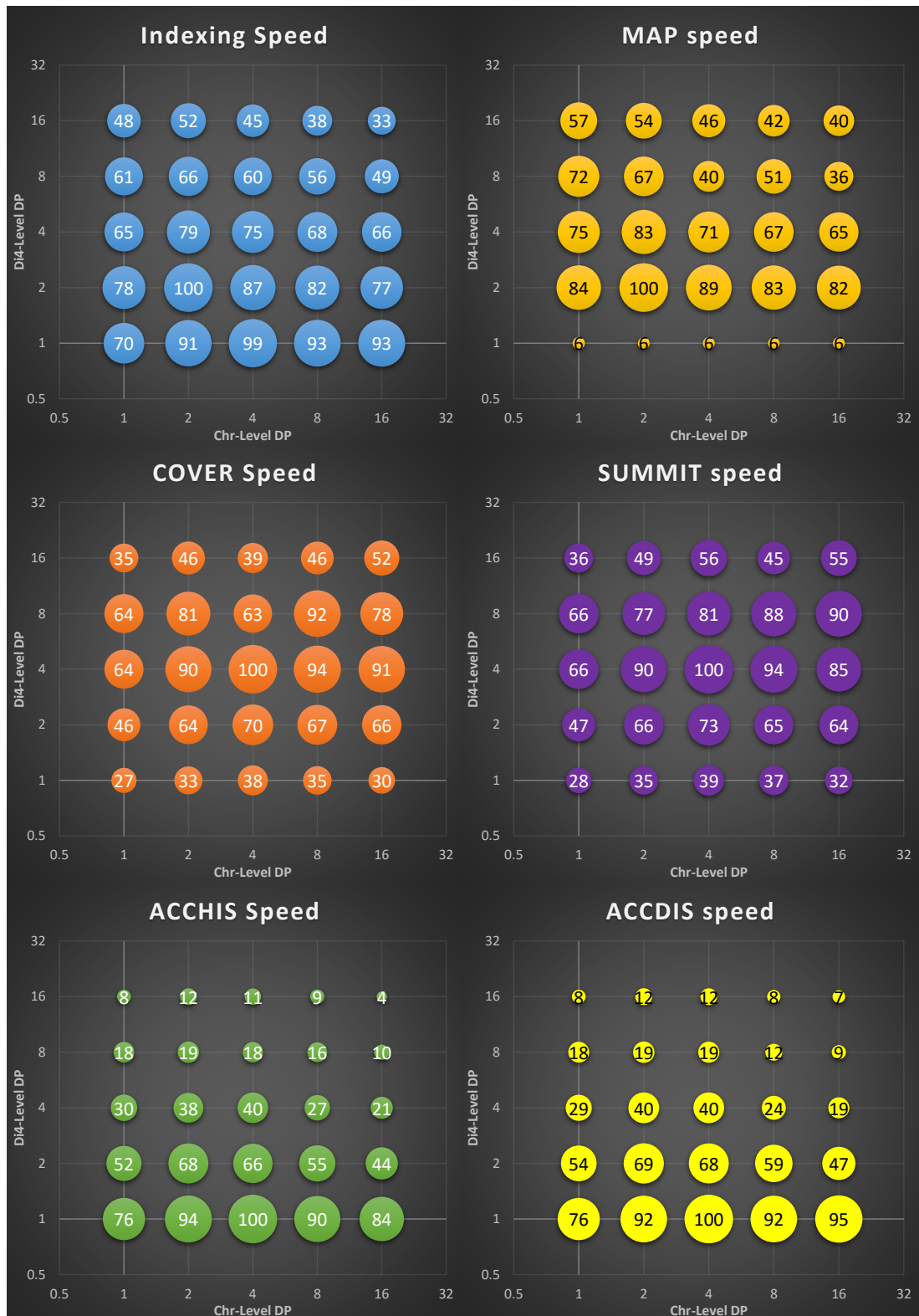


Figure 5.6: Performance evaluation based on different degrees of parallelization. The values are in performance coefficient such that 100 is the fastest.

5.2.5 Performance evaluation based on different persistence setup

Present section discusses Di4 performance correlation with persistence setup. In general, Di4 performance is evaluated under varying *partition allocation unit size* for NTFS partition spanning 512b to 8192b, and *block size* spanning 512b to 8192b. The evaluation is done using 1,000 ENCODE narrow peaks containing 34,099,281 intervals in total, and runs on M1 machine. The results are given in Figure 5.7 for *indexing speed* and runtime of functions MAP, COVER, SUMMIT, ACCHIS, and ACCDIS; and in Figure 5.8 for persisted data size. In general, partition allocation size (cluster size) do not affect the index file size (see Figure 5.8) which is due to the SSD storage type of M1 machine (compared to HDD); while block size is in direct relation with file size, such that with smaller block sizes (e.g., 512bytes) data is more compact than with higher block sizes. The functions performance also vary based on different partition allocation and block size; however, the data and the function also affect this evaluation. For instance, if on average, snapshots are bigger than block sizes, then multiple blocks should be read to de-serialize a snapshot, while, have the same snapshot on a bigger block size would require a single block to de-serialize the snapshot. In general, the functions perform relatively better with 4Kbytes block and cluster sizes.

5.3 Comparison of Di3 and Di4 performance with common bioinformatics tools

We start with an evaluation of Di3 enacted from a user interface, and then we present a comparison with BEDTools and BEDOPS, the most popular tools for genomic region calculus.

This section compares the performance of Di3 and Di4 against the two tools commonly used in genomic region processing, namely, BEDTools [35] and BEDOPS [36]. The benchmark runs on the M2 machine, with Di3 and Di4 executed under Microsoft Windows® 10 operating system, and BEDTools and BEDOPS under Linux. The BEDTools and BEDOPS run in-memory; therefore, Di3 and Di4 are also executed in-memory (i.e., without persisting data structures to hard disk). Additionally, the Python and shell scripts for the batch execution of BEDTools and BEDOPS are prepared as it follows.

```
1 sort-bed reference.narrow > reference_sorted.bed
2 for i in Files/*
3 do
4     sort-bed $i > $i.sorted
5     bedmap --ec --count --echo reference_sorted.bed $i.sorted > $i.res
6 done
```

5.3. Comparison of Di3 and Di4 performance with common bioinformatics tools

```
1  from __future__ import print_function
2  from subprocess import call
3  import os
4  import pickle
5
6  def ListAllFiles(scd):
7      f = []
8      for root, sub, files in os.walk(scd):
9          for x in files:
10             f += [os.path.join(root, x)]
11      return f
12
13  path = "testFiles/"
14  singleFile = path + "/allFiles"
15  singleFileSorted = singleFile + "Sorted"
16  filenames = ListAllFiles(path)
17  with open(singleFile, 'w') as outfile:
18      for fname in filenames:
19          with open(fname) as infile:
20              for line in infile:
21                  outfile.write(line)
22
23  f = open(singleFile, "r")
24  lines = [line for line in f if line.strip()]
25  f.close()
26  lines.sort()
27  pickle.dump(lines, singleFileSorted)
28  with open(singleFileSorted, 'w') as outFile:
29      for item in lines:
30          outFile.write(item)
31
32  call("bedtools map -a ~/Desktop/refP.bed -b "+ singleFile + " -o count > ~/
      Desktop/results/res2.bed", shell = True)
```

Among the possible operations that are available from the Di4B Command Line Interface, BEDTools and BEDOPS implement the MAP operator, i.e., given a reference sample, find input intervals overlapping with the reference regions. Therefore, we compared the MAP operator of Di3 and Di4 against the **bedtool map** from BEDTools and the **bedmap** from BEDOPS. We considered two typical *usage scenarios* in genomic region processing: (i) personal repository, and (ii) on-the-fly processing.

Personal Repository

This is a common scenario for bioinformaticians', where a personal repository of in-house data, as output of the execution of a NGS data processing pipeline, and/or data obtained from publicly available repositories is persistent on their machine. Such data are stored for further processing or to be used for comparative evaluation and cross-referencing. The repository is a collection of properly organized files, indexed and persistent in Di3 and Di4. This benchmark is run on M2 machine (see Table 5.3) and uses B1 and B2 datasets (see Table 5.2), and a reference sample including 196,180 regions (8.9MB in size).

The data were pre-processed, i.e., filtered, and regions in samples were sorted preliminary for

Chapter 5. Di4 and Di3 Performance Evaluation and Comparison

BEDTools and BEDOPS, and indexed for Di3 and Di4. Hence, the benchmark started from pre-processed data, and for comparison considered only execution time. As Figure 5.10 shows, Di3 and Di4 performs significantly faster than BEDTools and BEDOPS.

In general, the indexing time Di3 and Di4 is equal (as shown in Panel D and B on Figure 5.10 for first-pass indexing speed and total indexing elapsed time respectively), and so is the persisted information size (see Panel C on Figure 5.10). Di3 runs relatively faster than Di4 on the MAP operation (see panel A1 on Figure 5.10). As discussed in Section 5.2, the performance of Di4 for COVER and SUMMIT depends on n additional snapshots to traverse to reconstruct contributing intervals. The panels A2 and A3 on Figure 5.10 confirms that, in this scenario, the penalty of traversing larger snapshots with duplicated information of Di3 is less than the penalty of n additional traverse of Di4. The Panel A4 on Figure 5.10 plots the runtime of accumulation histogram operation, which shows that in general, traversing all snapshots of Di4 is faster than traversing Di3.

On-the-Fly Processing

Processing data on-the-fly is a bioinformaticians' daily-base scenario, where a relatively small dataset is obtained from the execution of a NGS data processing pipeline and it may not be archived for further evaluation. The in-memory version of Di3 and Di4 are benchmarked versus BEDTools and BEDOPS on MAP operation using one reference, including 196,180 regions (8.9MB in size), and three datasets of ENCODE narrow peak samples, C1, C2, and C3 (see Table 5.2) as target, and run tools on M2 machine (see Table 5.3).

The data were not pre-processed; i.e., data were not sorted for BEDTools and BEDOPS, and not indexed for Di3 and Di4. Hence, the execution time incorporates pre-processing time in all cases. Figure 5.9 shows that Di3 and Di4 performs faster than BEDTools and BEDOPS on the three datasets. This highlights that Di3 and Di4 are also an agile back-end data structure for on-the-fly processing, even by incorporating the indexing time within the processing time.

5.3. Comparison of Di3 and Di4 performance with common bioinformatics tools

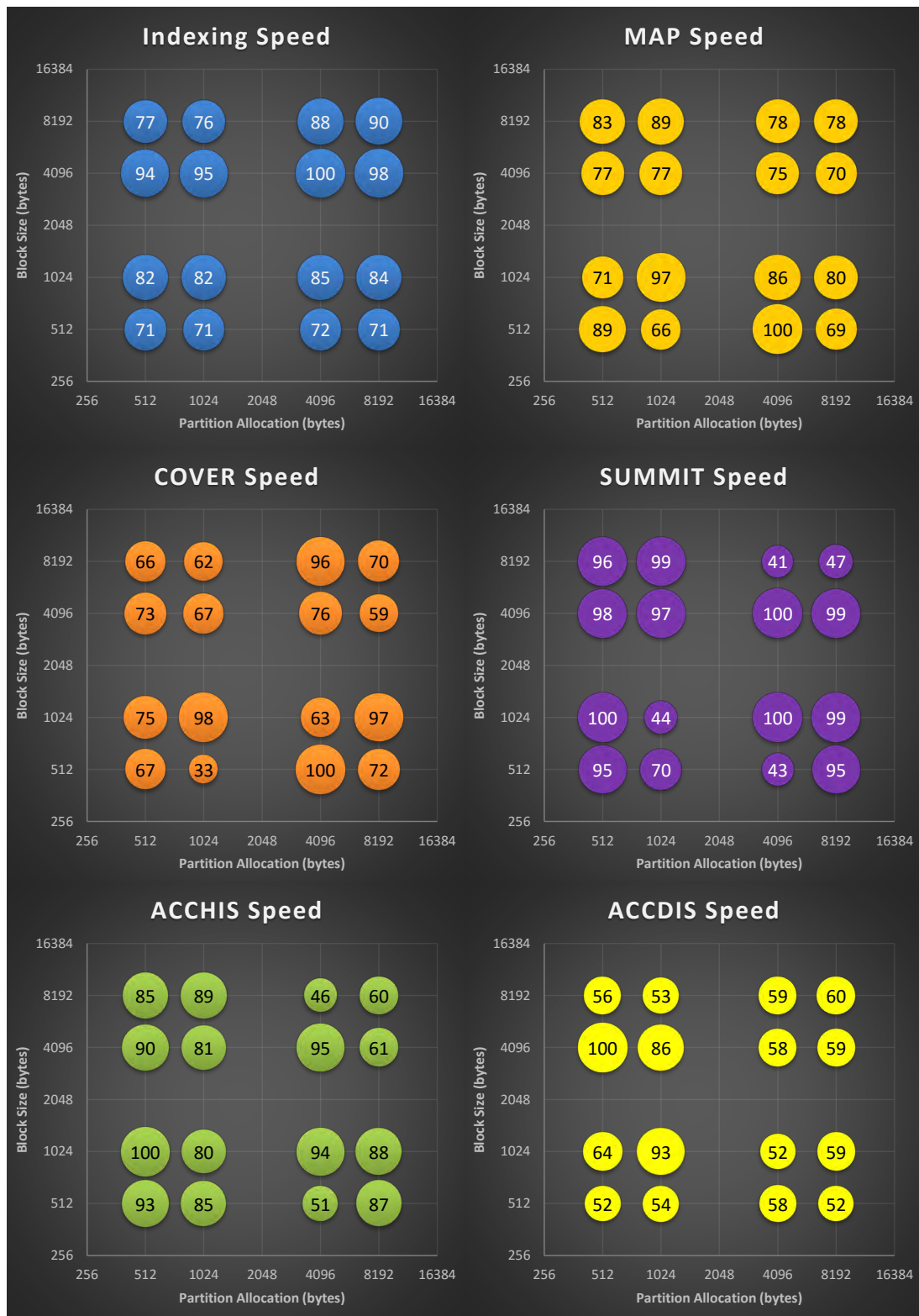


Figure 5.7: Di4 operations performance correlation with persistence setup. Block size is the number of bytes per file block size. The values are in performance coefficient such that 100 is the fastest.



Figure 5.8: Di4 index size on correlation with persistence setup. Block size is the number of bytes per file block size. The values are in size coefficient such that 100 is the smallest size.

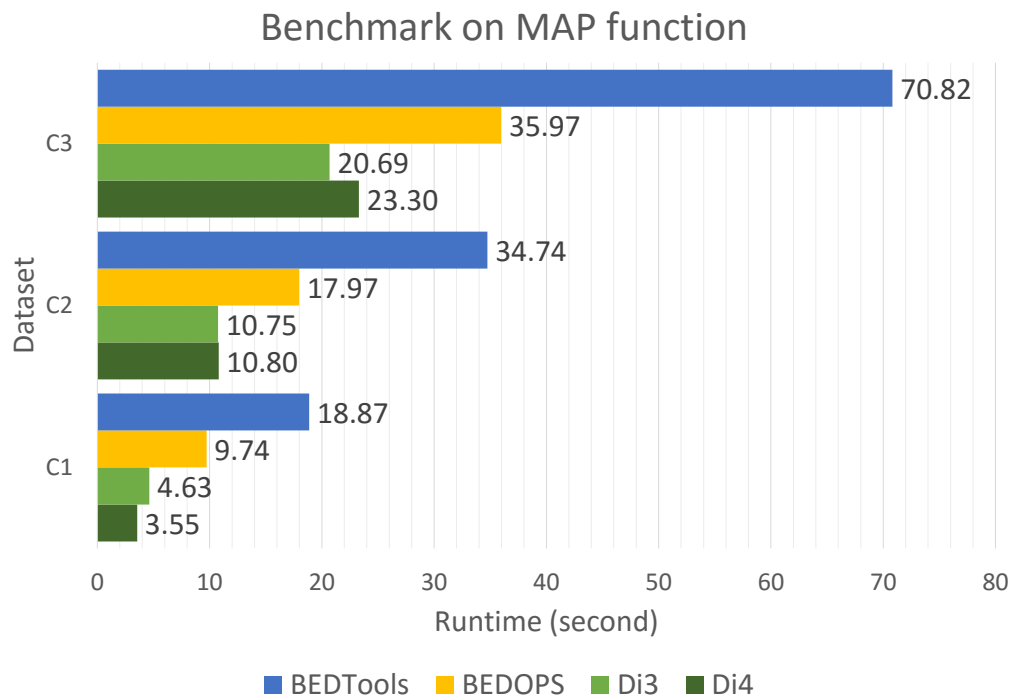


Figure 5.9: Benchmark: On-the-fly processing scenario

5.3. Comparison of Di3 and Di4 performance with common bioinformatics tools

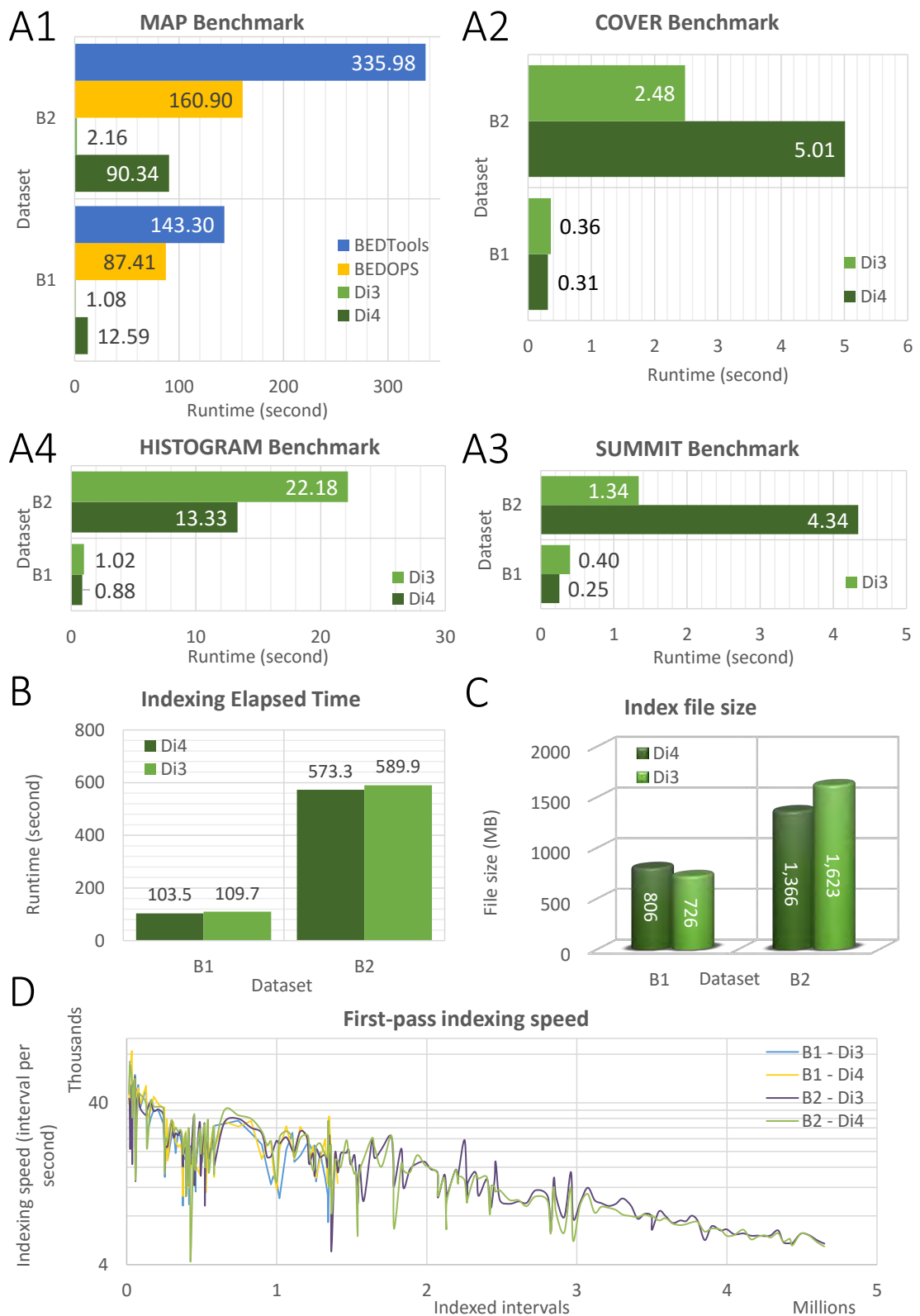


Figure 5.10: Benchmark: Personal repository scenario.

6 Di3 Application in Comparative Analysis of ChIP-seq Replicates

Present chapter discusses Multiple Sample Peak Calling (MSPC), a novel method to rigorously combine the results of peak calls in ChIP-seq replicates and to obtain new, sample-specific, peak lists taking into account their combined evidence. Additionally, present chapter discusses Multiple Sample Enriched Region Assessment (MuSERA), a novel, advanced graphical tool which extends Di3 to efficiently implement, extend and generalize MSPC. In addition, MuSERA further assess the identified enriched regions using Di3 through common tertiary analysis procedures such as functional analyses, correlation assessment, nearest feature distance distribution, and visualization on a genome browser. Furthermore, MuSERA implements an intuitive graphical interface that provides several graphic displays which help the user in gaining a deeper insight and biological evaluation of the analysis results.

6.1 Introduction

Chromatin immunoprecipitation followed by next-generation sequencing (ChIP-seq) is a multi-purpose technology that allows the precise determination of DNA or RNA sequences within a sample of interest [153], and is widely used for studying chromatin modifications and Protein-DNA in vivo interactions. The analysis of ChIP-Seq samples outputs a number of enriched regions (ER), each indicating a protein-DNA interaction or a specific chromatin modification. ERs (or “peaks”) are called when the read distribution is significantly different from the background, and its corresponding significance measure (extreme value probabilities, p-value) is below a user-defined threshold. Inferior enrichment regions with p-values close to the background signal are either veritably a portion of background slightly enriched due to some biological or technical bias in experiment, or indeed are biologically important regions with enrichments less significant than expected. The protocol is subject to noise [154]; hence to avoid a large number of false positives, commonly used thresholds are often very stringent; this generates many false negatives (undiscovered, genuine interaction sites, with read distribution considerably above background level).

Given the intrinsic noise of the ChIP-seq protocol, it is good practice to repeat every experiment at least twice, as the guidelines of the ENCODE project indicate [30]. The information contained in replicates can then be used to assess the validity of the peaks obtained from a single sample, especially of those with low-intensity. Biological replicates (i.e., multiple different but biologically equivalent samples, grown/treated under the same conditions) are used to assess the variations of the biological effect studied, and the differences among samples are attributed to true biological variability. Technical replicates (i.e., multiple samples taken from the very same biological system) are used to measure the reproducibility of a specific experiment and its analysis, and differences are attributed to technical issues in measurement. In both cases we expect largely overlapping ER patterns across replicates, with technical replicates much more homogeneous than biological ones. Leveraging on this feature, we propose to use replicates for a comprehensive evaluation of ERs across multiple samples.

In addition, a rapidly growing domain of important biological questions are being addressed by the comparative analysis of multiple ChIP-seq data as a significant and authentic technique. Multiple developmental stages, time courses (e.g., [155]), identification of differential binding patterns of Transcription Factors (TF) or histone mark modifications [156], [157], [158] are of distinct biologically important aspects appropriately address by comparing multiple ChIP-seq data. Current analytical approaches for ChIP-seq analysis are largely tend towards single-sample studies (reviewed in [159]), and have limited applicability in comparative settings that aim to identify combinatorial patterns of enrichment across multiple datasets, and discriminate sub-threshold binding from truly non-bound regions.

Multiple Sample Peak Calling In this chapter, we propose Multiple Sample Peak Calling (MSPC), a novel method to rigorously combine the results of peak calls in ChIP-seq replicates and to obtain new, sample-specific, peak lists taking into account their combined evidence. The method takes as input, for each replicate, a list of enriched genomic regions and a measure of their individual significance in terms of a p-value. Starting from a permissive call, the initial ERs are divided in “stringent” (highly significant) and “weak” (moderately significant), and the presence of overlapping enriched regions across multiple replicates is assessed. Non-overlapping regions can be penalized or discarded according to specific needs. The significance of the overlapping regions is rigorously combined with the Fisher’s method to obtain a global score. Finally, this score is assessed against an adjustable threshold on the combined evidence, and peaks in each replicate are either confirmed or discarded (see Figure 6.1). In other words, we are able to “rescue” weak peaks, which would probably be discarded in a single-sample analysis, when their combined evidence across multiple samples is sufficiently strong.

Multiple Sample Enriched Region Assessment The authenticity of combined evidence depends on variety of factors including the quality of replicates and accordingly called ERs, and also the choice of parameters to combine evidences. Inaccurate setup may lead to un-

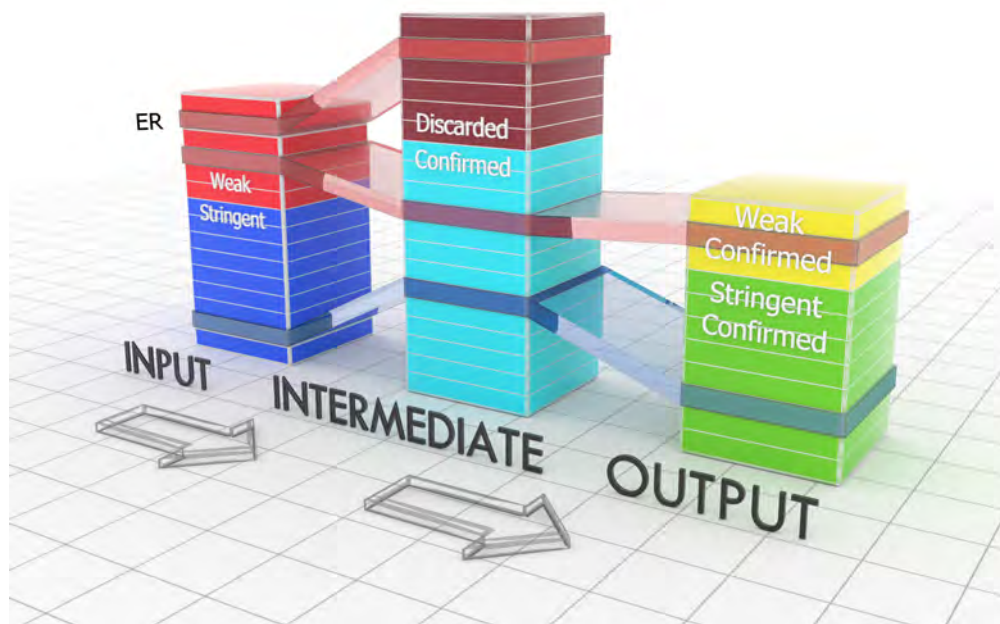


Figure 6.1: Schematic view of the proposed method. First, with a permissive call, peaks of a single individual sample are divided as stringent and weak. Then, combining the evidence of multiple replicates, the peaks in each replicate are confirmed or discarded.

desirable results. However, commonly an imprecise setup is not evident until the results are further assessed using common procedures such as visualization on a genome browser, functional analysis, or nearest-neighbor search, which motivates a trial-and-error paradigm for correct setup. For instance, combining two replicates with stringent ERs may confirm a small fraction of the ERs and discard a majority of them. In this case, a further assessment such as calculation of distribution of distances between the ERs of two replicates may show that ERs are relatively close to each other but non-overlapping. This observation can hint on peak calling parameters (e.g., peak calling threshold is very high so that a number of narrow and adjacent peaks are determined instead of a wide and strong one), or may encourage being more flexible on overlapping ERs requirement. Alternatively, suppose two replicates which are experiments on TFBS and hence it's expected from identified ERs to be at specific distance from known genes. It is expected from confirmed ERs to be close and discarded ERs to be farther. However, this can be verified using known genes and different sets of ERs as inputs to functional analysis and nearest neighbor search methods; where observations may encourage modifying the parameters of combining replicates or may hint an improper setup of the pipeline (e.g., imprecise peak calling setup, improper data cleaning pipeline, a wrong down-sampling of reads, or a bad choice of control sample). All these are based on trial-and-error and require the usage of different programs; by importing the output of one program into another (assuming the possibilities on inconsistency between the supported formats and possible requirement for conversions) which is cumbersome. To address this

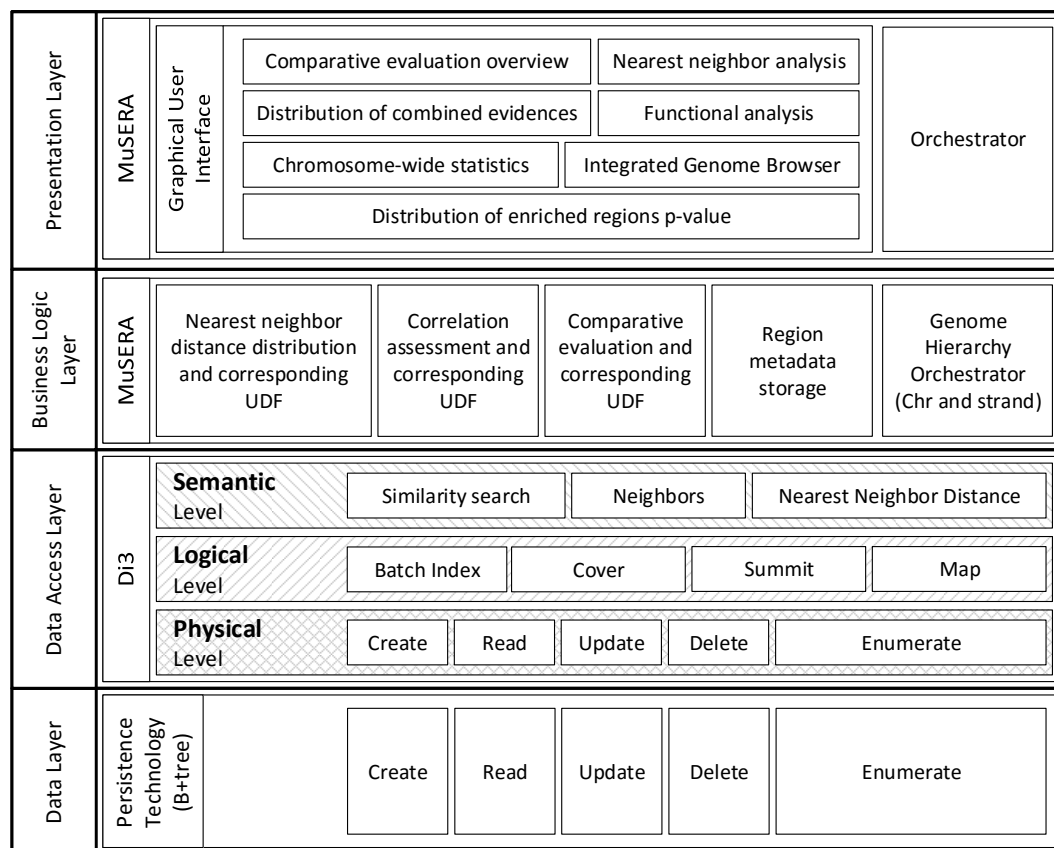


Figure 6.2: MuSERA architecture on extending Di3.

need, in this chapter, we present Multiple Sample Enriched Region Assessment (MuSERA), a novel, advanced graphical tool which extends Di3 to efficiently implement, extend and generalize MSPC, and in addition performs functional analyses on the identified ERs; furthermore, through its intuitive graphical interface it provides several graphic displays that help the user in gaining a deeper insight and biological evaluation of the analysis results. The architecture of MuSERA on extending Di3 is given in Figure 6.2 which includes only the component of Di3 that are used to implement the functions of MuSERA. The extension of Di3 operations, and business logic layer components of MuSERA for each operation are discussed in sections 6.4, 6.5, 6.6, 6.7, and 6.8, respectively on combining replicates, functional analysis, nearest feature distance distribution, correlation assessment, and integrated genome browser.

6.2 Related Works

Enrichment analysis of single ChIP-seq sample is well established; however, comparative enrichment analysis using two or more ChIP-seq samples with each other lacks a comprehensive method. In this section, we explain few available options.

6.2.1 Binary analysis

An intuitive method would be comparing two ChIP-seq samples by overlapping ERs, however this method reported to have intrinsic statistical [160] and biological problem. ChIP-seq experiments are subject to noise, and hence it is suggested to repeat an experiment at least twice [30], however, by overlapping peaks from repeated evidences, typically merely ~75% co-localization is observed, although binding profile is virtually identical and global similarity assessed by Pearson correlation coefficient (PCC) is of 0.9 and higher [160] (a phenomena known as "winner's curse" [161]). Regardless of the quality of the samples, the produced replicates tend to have a degree of dissimilarity because of possible biological variations and instrumental errors and biases. Therefore, enrichment regions above a certain threshold in one replicate, might be missing a supportive evidence from other replicate because the enriched region might be below the threshold, and vice-versa. The challenged might be solved with permissive p-value threshold, however, it is discouraged because of the high false-positive penalty. Therefore, binary analysis does not differentiate between sub-threshold binding, and truly non-binding site; and neither accounts for experimental variations defined in biological and technical replicates.

6.2.2 Alignment read Merging

An intuitive way to combine evidence in replicates is to merge the alignment reads, and use a peak caller on the combined data. As the combined dataset corresponds to the sum of the two signals, weak, co-occurring peaks should increase their significance. To this end, alignment files from replicates can be merged, and using merged backgrounds, call peaks on merged replicates. As we previously reported [162], merging the alignment files "averages" replicates with different sets of peaks, which does not necessarily give rise to true sub-thresholded binding sites. In other words, it uniformly increases statistical evidences that although reinforces binding signal, but an inevitable side effect is that it potentially increase the number of false positives. Additionally, the strategy is limited in applicability, it combines the reads of two biological/technical replicates and produces one output, disregarding the true invariance between the replicates. For instance, given three biological replicates, it is desired to combined the statistical evidences for binding sites with an objective of giving rise to sub-thresholded bindings considering a degree of dissimilarity in binding profile as experimental invariance. The merging strategy, increases evidences for binding sites, and may give rise to sub-threshold bindings, however, it does not account for biological invariance, and also the output is a merge of three replicates that requires a devious procedure to separate for each; if possible. Besides, the merging strategy has no user-defined parameter to tune the results and to weight co-occurrence and significance of ERs.

6.2.3 Irreproducibility Discovery Rate

Irreproducibility Discovery Rate (IDR) [163] is a metric quantifying the reproducibility of a peak across two ChIP-seq replicates by comparing the two lists of ERs, ranked according to their significance. In essence, after calling the peaks, the IDR pipeline uses a bivariate rank distribution to separate the signal (reproducible peaks) from noise (irreproducible peaks) in an experiment (or pairwise comparison). Each peak is associated with an IDR value, which quantifies the probability that the peak belongs to the irreproducible set. IDR is computable using the scripts provided by Anshul Kundaje.¹ The IDR [163] is a measure of the consistency of ERs identified in replicates, which has been systematically assessed in the ENCODE project. As we reported [162], the IDR method is rather stringent and generates only a small set of validated (reproducible) peaks, also we showed that a number of peaks considered “irreproducible” by IDR (with a 0.05 threshold) are validated by motif analysis. Finally, the IDR can be directly computed only for pairs of replicates, and an extension for more is ambiguous and indirect.

6.2.4 joint Analysis of Multiple ChIP-seq Datasets

The joint analysis of multiple ChIP-seq datasets (jMOSAiCS) [20] is a generic tool for joint analysis of multiple ChIP-seq samples, which can be also used to find common patterns of enrichment between ChIP-seq replicates. First, the MOSAiCS peak caller [164] pre-processes replicates and corresponding control samples by binning the mapped read counts on the genome (default width of 200 bp), and applies the MOSAiCS model fit to each replicate-control pair individually. Afterwards, the jMOSAiCS model is applied to the data fitted with MOSAiCS: region-specific enrichment patterns are determined by posterior probabilities assigned to the internal variables, and a binary variable denotes the potential enrichment of a region based on dependencies among samples. jMOSAiCS is designed to detect combinatorial patterns of enrichment in multiple ChIP-seq samples. Even if jMOSAiCS is conceived to integrate different ChIP-seq datasets that profile distinct features on the same biological sample, it can also be applied to replicates of the same ChIP-seq. As we reported [162], jMOSAiCS commonly produces a very large amount of ERs in each experiment. We checked for the enrichment of the Myc E-box in the peaks identified by jMOSAiCS, and it showed the presence of the Myc canonical E-box only in half of our test samples. The observation concludes that when jMOSAiCS applied to replicates of the same ChIP-seq experiment, has the tendency of introducing a large amount of extra peaks, which are not always validated by motif analysis. Moreover, jMOSAiCS requires significant computational resources, such that the running times for two replicates were in the order of 3 hours, with about 40 GB of memory consumption, on a server with two Intel Xeon E5-2650 processors and 64 GB of RAM, as this tool starts from alignment files and finds ERs independently. In addition, jMOSAiCS requires data fitted to MOSAiCS model, which restricts the choice of peak caller to MOSAiCS only and that may not fully adapt common pipelines used across different labs.

¹Available at the URL <https://sites.google.com/site/anshulkundaje/projects/idr>

Table 6.1: Groups of ERs based on p-values

Condition		Action	Description
$p_{ji} < T^s$		$r_{ji} \in R_j^s$	Stringent
$p_{ji} > T^w$		$r_{ji} \in R_j^b$	Background
$T^s < p_{ji} < T^w$	$K \geq C \wedge X_{ji}^2 > \gamma$	$r_{ji} \in R_j^c$	Confirmed
	Otherwise	$r_{ji} \in R_j^d$	Discarded

6.3 Definitions

Given a set of J replicates, each sample j is associated with a set R_j of I enriched regions $r_{ji} : R_j \{r_{j1}, r_{j2}, \dots, r_{ji}, \dots, r_{jI}\}$. Each region r_{ji} is defined by (chromosome _{ji} , start _{ji} , end _{ji} , p_{ji}) where p_{ji} denotes a measure of the significance of r_{ji} (i.e. its p-value). T^s is a stringent threshold on p-values, defining a set R_j^s of stringent (highly enriched) ERs; $R_j^s : r_{ji} \in R_j^s \iff p_{ji} < T^s$. Similarly, we define a set R_j^w of weak (moderately enriched) ERs, containing all regions whose Pvalue is between T^s and a weak threshold T^w , with $T^w > T^s$, i.e. $R_j^w : r_{ji} \in R_j^w \iff T^s < p_{ji} < T^w$. Clearly, $R_j^w \cap R_j^s = \emptyset$ and, if T^w is the maximum p-value allowed for an ER to be associated with sample j , $R_j^w \cup R_j^s = R_j$ (see Table 6.1, and Figure 6.3).

For each region i of each sample j , let $r_{ji,k}$ denote the region of sample k overlapping with r_{ji} , if any. If sample k has multiple regions overlapping with r_{ji} , we choose the most/least significant one, i.e. the one with the lowest/highest p-value (user defined). Let R_{ji} be the collection of $r_{ji,k}$ for $1 \leq k \leq J$, including r_{ji} itself. Let $K = |R_{ji,*j}|$ be the cardinality of $R_{ji,*}$, the set of the ERs intersecting with r_{ji} , with $1 \leq K \leq J$ by definition.

The significance of an ER is assessed through a process of combining evidences with respect to a combined evidence X^2 statistics; this process uses a Fisher's combined probability test [165], that follows a χ^2 distribution with $2K$ degrees of freedom, and a stringency threshold (γ), which defines confirmed ($R_j^c = \{r_{ji} \mid X_{ji}^2 \geq \gamma\}$) and discarded ($R_j^d = \{r_{ji} \mid X_{ji}^2 < \gamma\}$) sets of ERs. The method generates an output set (R_j^o) by applying a multiple testing correction procedure on confirmed ERs. Additionally, we define the following sets:

- i. stringent confirmed: $R_j^{sc} = \{r_{ji} \mid p_{ji} < T^s \wedge X_{ji}^2 \geq \gamma\} \subseteq R_j^c$
- ii. stringent discarded: $R_j^{sd} = \{r_{ji} \mid p_{ji} < T^s \wedge X_{ji}^2 < \gamma\} \subseteq R_j^d$
- iii. weak confirmed: $R_j^{wc} = \{r_{ji} \mid T^s \leq p_{ji} < T^w \wedge X_{ji}^2 \geq \gamma\} \subseteq R_j^c$
- iv. weak discarded: $R_j^{wd} = \{r_{ji} \mid T^s \leq p_{ji} < T^w \wedge X_{ji}^2 < \gamma\} \subseteq R_j^d$
- v. multiple-testing confirmed: $R_j^{mtc} \subseteq R_j^o$
- vi. multiple-testing discarded: $R_j^{mtd} \subseteq R_j^o$

We distinguish between technical and biological replicates of an experiment. Technical replicates aim at controlling the variability of the experimental procedure used to obtain the data and should yield exactly the same results in absence of experimental noise. In a ChIPseq experiment, this corresponds to performing multiple times the same ChIP protocol on the same biological sample, followed by independent sequencing on the same platform; we expect to observe a significant overlap between ER lists in these samples. Conversely, biological replicates are obtained by applying the same protocol on biologically equivalent samples, what could give rise to different binding profiles of a transcription factor, as in the case of tumor samples; here, the variability in the data can also stem from the “true” biological variation of the phenomenon of interest. Consequently, the lack of overlap between ERs in biological replicates does not necessarily correspond to a false positive result, as it could reflect a true biological interaction occurring only in some samples. With our method, the user is able to control for the required level of overlap and combined significance, according to the specificities of the dataset.

6.4 Combining Replicates

The main idea behind our method is that repeated evidence across replicates can compensate for a lower significance in a single sample, which is implemented through the Fisher’s method [165]. The Fisher’s method combines the p-values of each test in a global test statistics that follows a χ^2 distribution with $2K$ degrees of freedom (where K is the number of tests combined); therefore, it can be used to falsify the statement “all null hypotheses are true”, i.e. “all overlapping ERs are due to background noise”. Comparing intersecting ERs from a set of J replicates is equivalent to test the same genomic region in independent experiments against the same null hypothesis H_0 , i.e. “the number of reads in the region under study is sampled from the background distribution”, and obtaining independent probabilities of rejecting H_0 (i.e. independent p-values).

6.4.1 Authenticity of Combining Replicates

A p-value is “the probability of the observed results, plus more extreme results, if the null hypothesis were true” [166], or in algebraic notation $\text{Prob}(X \geq x \mid H_0)$ for X being “a random variable corresponding to some way of summarizing data (such as a mean or proportion), and x is the observed value of that summary in the current data.” [166] The p-value is calculated, assuming that the H_0 is true, therefore lower p-values could not be interpreted as rejecting H_0 . Comparing p-value to a significance level yields either to rejection of H_0 , or *cannot reject* H_0 which does *not* imply that H_0 is *true*. Therefore, when the p-value of an enrichment region is less than an expected value, it yields insufficient evidence to decide upon the probability of observing the enriched region only by chance, it should not be miss-interpreted that this means the region is observed only by chance. Therefore by combining corresponding p-values into one single test statistic, one increases observation, hence statistical evidence for

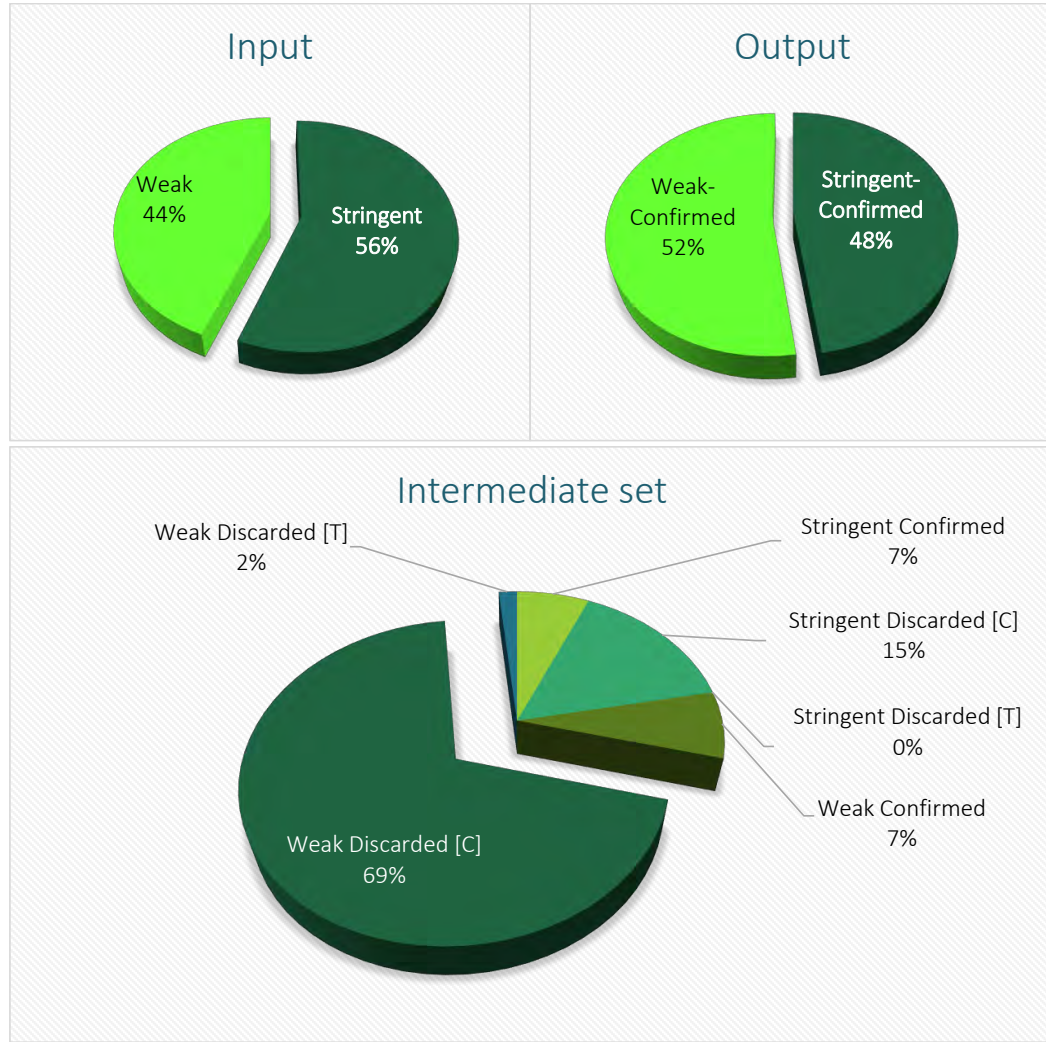


Figure 6.3: The information for these pie-charts are provided by MuSERA and MSPC and are plotted here as an example. These are the information about sample wgEncodeBroadHistoneK562CtcfStdAlnRep1, analyzed with wgEncodeBroadHistoneK562CtcfStdAlnRep2 as supporting sample, and considering them as biological samples with $T^s = 1E-8$, $T^w = 0.01$, $C = 2$, $\gamma = 1E-8$, and $\alpha = 0.05$.

[C]: refers to ERs discarded failing compiling minimum overlapping ER requirement.

[T]: refers to ERs discarded failing the combined stringency test.

enrichment at specific position on genome. Note that, p-value is neither the probability of accepting H_0 , nor rejecting alternative hypothesis (H_a) [167] [168], rather H_0 is *rejected at a given significance level* if p-value is lower than or equal to the given significance level [169]; also note that, the significance level is not determined by p-value, it's rather given. [167] [168]

6.4.2 Combining Test Statistics

Weak peaks, corresponding to a moderate enrichment of ChIP-seq reads, could arise from a random deviation from the background distribution, and they are therefore usually discarded from a single-sample analysis. However, if the same evidence is confirmed across replicates, it could correspond to a true interaction of the antibody with its target on the DNA. The significance of an enrichment is assessed via multiple independent tests, however, if one or few test results indicate individually insignificant, “yet the aggregate gives an impression that the probabilities are on the whole lower than would often have been obtained by chance.” [165] To combine the individual one-sided test statistics, we assume the following prerequisite are satisfied for the p-values:

1. Experiments are independent, accordingly, the significant of enrichment region, are independent.
2. On-sided test statistics have continuous distribution function under null hypothesis, therefore, the distribution function of test statistic is uniformly distributed under the same overall null hypothesis on $(0, 1)$, hence, the inverse of standard normal distribution function, has a standard normal distribution under same overall null hypothesis. [170]

The general form of a statistics combining p-values is:

$$\Theta = \sum_{i=1}^K w_i F(p_i) \quad (6.1)$$

where the factor w_i denotes the weight of each test statistics combined; however, this factor is not always initially proposed by the author and for augmentation reasons, it is added up in latter researches, therefore to keep methods in original and coherent format, we would decline this parameter. Some of the methods for combining test statistics are as following:

i. **Ronald Fisher’s combined probability test** [165]

$$\Theta_F = X^2 = -2 \sum_{i=1}^K \log_e p_i \quad (6.2)$$

where K is the number of independent p-values (p_i) being combined. Each of the p_i ’s is uniformly distributed, therefore the negative natural logarithm will render them exponentially distributed, and scaling each of them by two gives them a Chi-squared (χ^2) distribution with two degrees of freedom, hence their summation will be following the same distribution with $2K$ degrees of freedom. [171]

ii. **Liptak’s method for combined probability test** [172]

$$\Theta_L = \sum_{i=1}^K (\Phi^{-1}(1 - p_i)) \quad (6.3)$$

where Φ^{-1} is the inverse of standard normal distribution function.

iii. **Mudholkar and George** [173]

$$\Theta_M = \sum_{i=1}^K \left(-\log \frac{p_i}{1-p_i} \right) \quad (6.4)$$

iv. **Wilkinson** [174]

$$\Theta_W = \sum_{i=1}^K (I(p_i \leq \epsilon)) \quad (6.5)$$

This method is based upon the number of p_i 's below ϵ threshold.

v. **Truncated product method** [175]

$$\Theta_Z = \sum_{i=1}^K (I(p_i \leq \epsilon) \log_e p_i) \quad (6.6)$$

There have been plenty of efforts on assessing asymptotic optimality of distinct methods of combining independent tests of significance. Littell and Folks [176] [177] compared four different methods via exact Bahadur relative efficiency [178], (1) Fisher's method, (2) a method which is based on the mean of normal transforms of the significance levels, (3) a method based on maximum significance level, and finally (4) a method based on minimum significance level. They reasoned, "there is no uniformly best method of combining independent tests, at least from a power point of view". They showed that Fisher's method is at least as efficient as studied methods and "under certain conditions, Fisher's method is optimal among all tests based on the data, not only among methods of combining". Another and relatively new effort [179] reasoned that there exist no uniformly most powerful test. "Various combination methods were compared empirically instead of by further theoretical investigation. However, all these investigations failed to find any practically most powerful region". Another research [180] reasoned that Wilkinson's method [174] is asymptotically optimal than Fisher's method.

Considering all novel researches on comparing meta-analysis methods, to combine the significance of a set of overlapping peaks, we use the Fisher's combined probability test [165]:

$$X_{ji}^2 = -2 \sum_{r_{ji,K} \in R_{ji,*}} \ln(p_K) \quad (6.7)$$

where p_i is the p-value of the region $r_{ji,K}$, and X_{ji}^2 is a test statistic which follows a χ^2 distribution with $2K$ degrees of freedom (where K is the number of combined p-values).

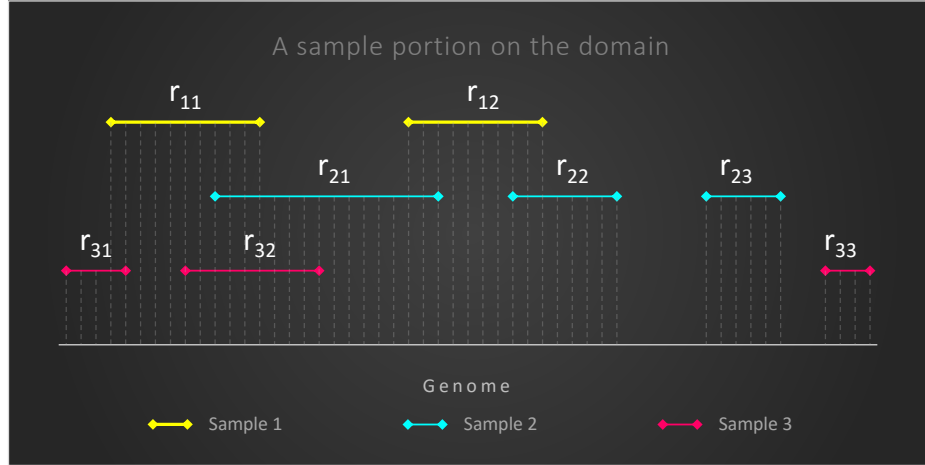


Figure 6.4: Possible overlapping conditions with three replicates.

6.4.3 Method of Combining Replicates

Background

Intersecting regions between multiple samples can be far from trivial, especially when $J > 2$. We illustrate some of the situations that can arise for $J = 3$ in Figure 6.4. A region in one sample can overlap with multiple regions in another sample, like r_{12} in sample 1 with r_{21} and r_{22} in sample 2 on Figure 6.4. As some peak callers tend to produce small, adjacent peaks that correspond to the same physical interaction between the antibody and the target, we consider only the region with the most stringent p-value in sample 2 to validate r_{12} . If both r_{21} and r_{22} are considered, the ER in the first sample would be backed by two, independent evidences in the second sample, thus leading to an overestimation of the significance of r_{12} . Similarly, r_{11} is evaluated together with r_{21} and one of the two regions r_{31} and r_{32} . Moreover, a region can overlap with distinct regions in multiple samples, which do not overlap with each other, like r_{12} , r_{21} and r_{22} .

Considering the number of regions and samples, determination of intersecting regions with a given region is computationally expensive process. In literature, there exist various efficient algorithms addressing the issue. However, depending on our specific need, the performance of these methods vary considerably. In following we emphasize the methods and their comparisons.

ERs can efficiently be sorted on genomic coordinates. Hence one trivial approach of finding intersection between J samples could be to use an algorithm based on ordered lists. Using this representation, samples could be intersected using a linear merge by scanning both lists in parallel. Such algorithms are in complexity order of $O(\sum_j |S_j|)$. These approaches are inefficient when only a small fraction of ER's intersect or the ER count in samples differ significantly [181].

Data structures such as binary search trees offer efficient search bases, Red-Black trees are an example data structures utilizing these bases [182]. Red-Black trees are type of self-balancing search trees which at every node insertion or deletion operation change the height of tree to preserve the height-balance (i.e., the height of left and right subtrees of each node are either equal or their difference is at most 1). The complexity of search operations could be of order $O(n)$ in unbalanced trees, while height-balanced trees guarantee $O(\log_2 n)$ complexity, where n is the number of items inserted in the binary tree.

Interval trees [183] [184] are an augmentation of Red-Black trees with every node representing an interval (ER in this context). Interval trees keep all the intervals with start point smaller than root start point in left-subtree of the root, and all the intervals with start point greater than or equal to the root start point in right-subtree. This property allows queries such as "given an interval I , find first occurring interval in interval tree intersecting with I " to be done in $O(\log_2 n)$ time, while preprocessing takes $O(n \log_2 n)$ where n is the number of ERs in this context. Interval trees are output sensitive algorithms, meaning that, a query for all intervals intersecting with I require $O(k \log_2 n)$, for k being the number of intersecting intervals. Alternatively, [185] reported an algorithm for retrieval of k intersecting intervals with I in $O(k + \log_2 n)$ time.

Algorithm

Interval trees are optimal data structures for our challenges, hence we use the method to find ERs overlapping a given ER. To this end, for each chromosome of sample j we create an interval tree, this allows us to parallelize process chromosome wide. Then for each ER of sample j , we determine the intersecting ER's on corresponding chromosome of samples $\{1, 2, \dots, J\} - j$. The process requires $O(kn \log_2 n)$ time for one sample and $O(Jkn \log_2 n)$ for all samples. Since k and J are considerably smaller than n , hence asymptotic behavior of the algorithm will be in order of $O(n \log_2 n)$ for one or more samples test. Unlike linear merge algorithms, this method does not necessarily require the ERs to be provided sorted, and the difference between samples in terms of ERs count and the fraction of intersecting ERs, has no impact on performance.

A flowchart of the method is given at Figure 6.5, and we discuss the detail in the following. We assign every ER r_{ji} in a given sample j to either R_j^s or R_j^w according to its significance. For a given ER, we then determine $R_{ji,*}$ as the set of ERs in the replicates that overlap with r_{ji} , including r_{ji} itself. The cardinality K of $R_{ji,*}$ represents a measure of the reproducibility of the signal in the region spanned by r_{ji} , while the significance of $r_{ji,K} \in R_{ji,*}$ is a measure of the intensity of the signal in a specific replicate K , given the background. We rigorously combine the significance of the overlapping ERs in $R_{ji,*}$ with the Fisher's method and define a new score for their combined evidence X^2 , as follows (see section 6.4.2 for details):

$$X_{ji}^2 = -2 \sum_{r_{ji,K} \in R_{ji,*}} \ln(p_K) \quad (6.8)$$

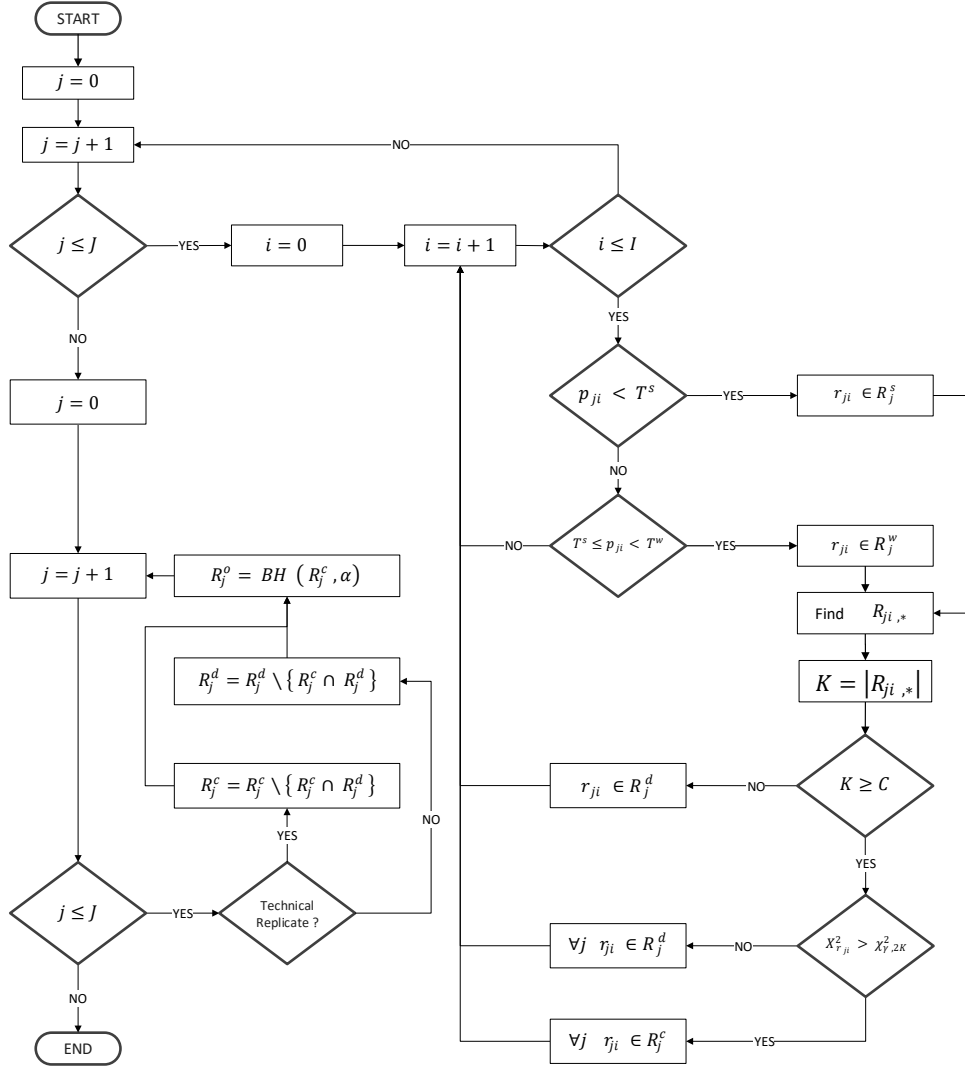


Figure 6.5: The flowchart of combining replicates. For the definition of symbols, see section 6.3; and for the description of method, see section 6.4.3.

Then, we compare X^2 with an adjustable threshold c :

1. if the desired stringency is obtained, we assign r_{ji} to the set R_j^c of confirmed peaks for sample j .
2. if the condition is not met, i.e. the combined evidence is not strong enough, we assign r_{ji} to the set R_j^d of discarded peaks for sample j .

All the ERs in $R_{ji,*}$ are assigned to the corresponding confirmed R_j^c or discarded R_j^d set, respectively. We leave the possibility to distrust a region r_{ji} , regardless of its significance, when it is not backed up by the presence of overlapping ERs in a minimum number of samples C . C is an adjustable parameter ranging between 1 and J , with different default values for

Algorithm 14 A UDF for comparative evaluation

```

1: procedure COMBINEERS( $R_{ji,*}$ )
2:   if  $|R_{ji,*}| \geq C$  then
3:      $X_{r_{ji}}^2 \leftarrow -2 \sum_{r_{ji} \in R_{ji,*}} \ln(p_{ji})$ 
4:     if  $X_{r_{ji}}^2 > \chi_{\gamma, 2K}^2$  then
5:        $\forall j \ r_{ji} \in R_j^c$ 
6:     else
7:        $\forall j \ r_{ji} \in R_j^d$ 
8:   else
9:      $r_{ji} \in R_j^d$ 
10:  return  $R_j^d$  and  $R_j^c$ 

```

biological and technical replicates. In summary, for a given sample j (see Table 6.1):

$$R_j^c = \left\{ r_{ji} \mid X_{ji}^2 \leq \gamma \wedge K \geq C \right\}$$

$$R_j^d = \left\{ r_{ji} \mid X_{ji}^2 > \gamma \vee K < C \right\}$$

We repeat this procedure for each sample. We note that an ER can participate in different sets of overlapping regions, as a consequence, it is possible that an ER is assigned to both the confirmed and discarded sets as a result of different tests. These peaks are assigned to the confirmed set if replicates are biological and to the discarded set if replicates are technical. In other words, as technical replicates are supposed to be very similar, for an ER it is enough to fail the test once to be removed from the confirmed set, while for biological replicates this condition is relaxed and it is enough to pass the test at least once for an ER to be confirmed.

6.4.4 Combining replicates using Di3

MuSERA extends the operations of Di3 through UDF to combine replicates based on the algorithm of MSPC (given in Figure 6.5). MuSERA defines a UDF for combining overlapping ERs (given in Algorithm 14), and defines a comparative evaluation function (given in Algorithm 15) that incorporates the UDF and Di3 functions to combine replicates.

6.4.5 Threshold Automatic Validation

The Fisher's method is a universal procedure to combine independent p-values. It is based on simple considerations: under the null hypothesis H_0 , p-values are uniformly distributed on the interval $[0, 1]$ and their logarithms follow an exponential distribution; the sum of the p-values, multiplied by a factor two, approximately follows a χ^2 distribution with degrees of freedom equal to twice the number of p-values combined. We confirm a peak when its combined p-value is below a definable threshold γ , i.e., $p_{ji}^{\text{comb}} < \gamma$. To increase computational

Algorithm 15 Comparative evaluation of ERs of ChIP-seq replicates

```

1: procedure COMPARATIVEEVALUATION
2:   for each  $r_{ji}$  do
3:     if  $p_{ji} < T^s$  then
4:        $r_{ji} \in R_j^s$ 
5:     else
6:       if  $T^s \leq p_{ji} < T^w$  then
7:          $r_{ji} \in R_j^w$ 
8:       else
9:         continue iteration with next  $r_{ji}$ 
10:     $f_{\text{map}}(r_{ji}, \mathfrak{D}^+, \text{CombineERs})$   $\triangleright$  Calls the MAP function of Di3 and passes:  $r_{ji}$  as
       reference interval, first resolution index, and CombineERs UDF
11:    for each sample  $j$  do
12:      if sample are technical replicates then
13:         $R_j^c \leftarrow R_j^c \setminus \{R_j^c \cap R_j^d\}$ 
14:      else
15:         $R_j^d \leftarrow R_j^d \setminus \{R_j^c \cap R_j^d\}$ 
16:         $R_j^o \leftarrow \text{BH}(R_j^c, \alpha)$   $\triangleright$  BH: Benjamini–Hochberg step-up procedure
17:    return  $R_j^o$ 

```

efficiency, we rewrite this condition on X_{ji}^2 as it follows. Let $F_K(x)$ be the right-tail cumulative probability for a χ^2 distribution with $2K$ degrees of freedom, i.e.,

$$F_k(x) = \int_x^{+\infty} \chi_{2K}^2(t) dt \quad (6.9)$$

thus, $p_{ji}^{\text{comb}} = F_K(X_{ji}^2)$. A threshold on the test-statistic equivalent to γ can be defined as $X_{\gamma, 2K}^2 : F_K(X_{\gamma, 2K}^2) = \gamma$. Hence, $p_{ji}^{\text{comb}} < \gamma \iff X_{ji}^2 > \chi_{\gamma, 2K}^2$. As the function $F_K(x)$ can be expensive to compute and the number of K overlapping peaks rarely exceed 5 – 10, for a given γ , we computed a look-up table for several values of K , and we test directly X_{ji}^2 against such look-up table values. With this procedure, the significance of a genomic region across replicates is rigorously weighted: the larger is the number of overlapping ERs, the weaker is the single-sample significance needed to confirm the peak. We note that, depending on the values of γ , K and p_k (the p-value of region k), we can avoid running the Fisher's test altogether. Let's suppose we have the same p-value $p_k = p^*$ for each k (i.e., for all K intersecting ERs); the condition $X_{ji}^2 > \chi_{\gamma, 2K}^2$ becomes:

$$-2 \sum_{r_{ji,k} \in R_{ji,*}} \ln p^* > \chi_{\gamma, 2K}^2 \quad (6.10)$$

and thus:

$$\exp\left(\frac{\chi_{\gamma, 2K}^2}{-2K}\right) > p^* \quad (6.11)$$

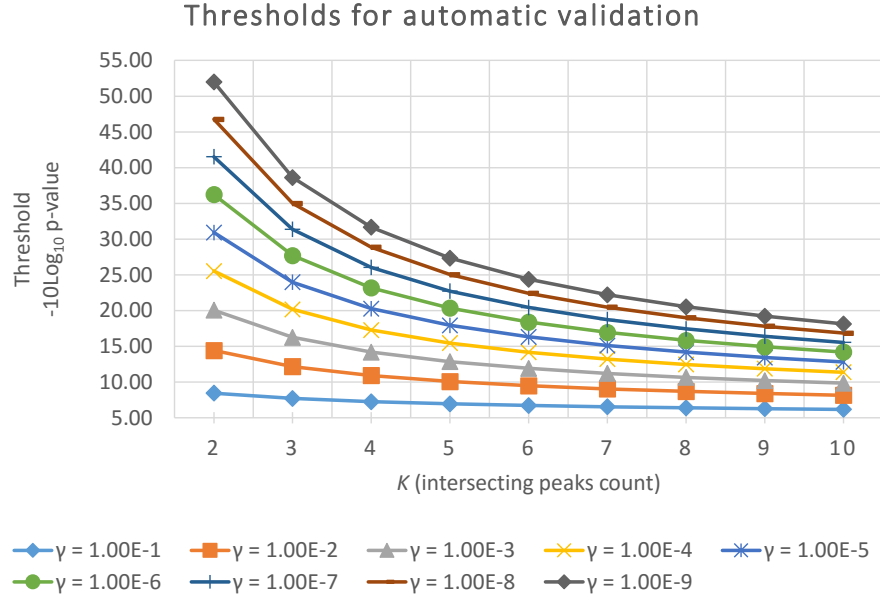


Figure 6.6: Maximum thresholds (in $-10\log_{10}$ units) for the p -value of each intersecting peak to automatically confirm all peaks (without need to run the Fisher's test), as a function of the intersecting peak number K and of the definable combined evidence threshold γ .

In other words, if the p -values of the K intersecting peaks are all below or equal to $p^* = \exp(\chi_{\gamma, 2K}^2 / -2K)$, the test is not performed and all peaks are automatically confirmed. The values of p^* for different thresholds γ are shown as a function of K in the Figure 6.6. We note that, when $\gamma = T^s$ and $C = 1$, all the strong peaks pass the test.

6.4.6 Example

We illustrate here how the ER sets generated by our method are obtained for the example in Figure 6.4. Let r_{12} , r_{32} and r_{33} be strong enriched regions (ERs) and $p_{21} < p_{22}$; the set of strong and weak ERs of each sample are defined as: $R_1^s = \{r_{12}\}$, $R_1^w = \{r_{11}\}$, $R_2^s = \emptyset$, $R_2^w = \{r_{21}, r_{22}, r_{23}\}$, $R_3^s = \{r_{32}, r_{33}\}$, and $R_3^w = \{r_{31}\}$. The corresponding ER sets generated by our method are detailed as it follows for two cases: 1) the samples are technical replicates; 2) the samples are biological replicates.

Case 1: Technical Replicates

Being the samples technical replicates, we set $C = J$ by default (C can be changed by the user). Our method begins by evaluating the first region of the first sample, r_{11} . The set of ERs overlapping with r_{11} is determined as it follows: $R_{11,*} = (R_1 \cap_{\min} R_1) \cup (R_1 \cap_{\min} R_2) \cup (R_1 \cap_{\min} R_3) = \{r_{11}, r_{21}, r_{32}\}$, where \cap_{\min} selects the intersecting region with the minimum p -value (highest significance) in each sample and $(R_1 \cap_{\min} R_1)$ includes r_{11} in $R_{11,*}$. As r_{11} intersects both with r_{31} and r_{32} from sample 3, our method considers only the most significant

ER among the two, which is r_{32} , according to our assumptions. As samples are technical replicates, our method computes $K = |R_{11,*}| = 3 = C$ and verifies that the condition $K \geq C$ holds for r_{11} . Then, it combines the p-values of the intersecting ERs using the Fisher's method as it follows:

$$X_{11}^2 = -2 \sum_{r_k \in R_{11,*}} \ln p_k = -2 \ln p_{11} - 2 \ln p_{21} - 2 \ln p_{32}$$

Depending on p_{11} , p_{21} , p_{32} and the adjustable threshold γ , the combined evidence of the intersecting ERs either confirms or discards all the ERs in $R_{11,*}$. If the ERs are confirmed, $r_{11} \in R_1^c$, $r_{21} \in R_2^c$ and $r_{32} \in R_3^c$. Otherwise, if the ERs are discarded, $r_{11} \in R_1^d$, $r_{21} \in R_2^d$ and $r_{32} \in R_3^d$.

Our method continues by assessing the second ER in the first sample, r_{12} . In this case, the set of intersecting regions is $R_{12,*} = (R_1 \cap_{\min} R_1) \cup (R_1 \cap_{\min} R_2) \cup (R_1 \cap_{\min} R_3) = \{r_{12}, r_{21}\}$. Since r_{12} does not intersect with the minimum number of required ERs (i.e., $|R_{12,*}| = K = 2 < C$), all these peaks are assigned to the discarded sets of the corresponding samples, i.e., $r_{12} \in R_1^d$ and $r_{21} \in R_2^d$.

In the second sample, the set $R_{21,*} = \{r_{21}, r_{12}, r_{32}\}$ is determined for r_{21} and, with $K = 3$, the condition on the minimum intersecting ERs is also met; thus, the p-values p_{21} , p_{12} and p_{32} are combined through the Fisher's method. The ERs are then confirmed, if their combined evidence satisfies the threshold, i.e., $r_{21} \in R_2^c$, $r_{12} \in R_1^c$ and $r_{32} \in R_3^c$; otherwise, $r_{21} \in R_2^d$, $r_{12} \in R_1^d$ and $r_{32} \in R_3^d$.

In the following, r_{22} , r_{23} and r_{31} are discarded (together with their intersecting ERs) because in these cases the number of intersecting ERs is insufficient (i.e., $K < C = J$): $R_{22,*} = \{r_{22}, r_{12}\}$ and $r_{22} \in R_2^d$, $r_{12} \in R_1^d$; $R_{23,*} = \{r_{23}\}$ and $r_{23} \in R_2^d$; $R_{31,*} = \{r_{31}, r_{11}\}$ and $r_{31} \in R_3^d$ and $r_{11} \in R_1^d$.

The ER r_{32} intersects with 2 other ERs ($R_{32,*} = \{r_{32}, r_{21}, r_{11}\}$), satisfying the $K \geq C$ requirement. If condition $X_{ji}^2 > \chi_{\gamma, 2K}^2$ is met, $r_{32} \in R_3^c$, $r_{21} \in R_2^c$ and $r_{11} \in R_1^c$; otherwise, $r_{32} \in R_3^d$, $r_{21} \in R_2^d$ and $r_{11} \in R_1^d$.

Finally, r_{33} is discarded because $K < C$: $R_{33,*} = \{r_{33}\}$ and $r_{33} \in R_3^d$.

The final outcome depends on the combined evidences X_{11}^2 , X_{21}^2 and X_{32}^2 . Let us assume that all combined evidences satisfy the condition $X_{ji}^2 > \chi_{\gamma, 2K}^2$ and therefore all the corresponding regions are confirmed. Thus, the final ER sets are the following:

$$\begin{array}{llll} R_1^{w,d} = \{r_{11}\} & R_1^{w,c} = \{r_{11}\} & R_1^{s,d} = \{r_{12}\} & R_1^{s,c} = \{r_{12}\} \\ R_2^{w,d} = \{r_{21}, r_{22}, r_{23}\} & R_2^{w,c} = \{r_{21}\} & R_2^{s,d} = \emptyset & R_2^{s,c} = \emptyset \\ R_3^{w,d} = \{r_{31}\} & R_3^{w,c} = \emptyset & R_3^{s,d} = \{r_{33}\} & R_3^{s,c} = \{r_{32}\} \end{array}$$

In the case of technical replicates, the set of confirmed peaks ($R_j^c = R_j^{w,c} \cup R_j^{s,c}$) is modified

by removing the ERs in common with the set of discarded peaks ($R_j^d = R_j^{w,d} \cup R_j^{s,d}$), resulting in the following sets: $R_1^{w,c} = \emptyset$, $R_1^{s,c} = \emptyset$, $R_2^{w,c} = \emptyset$, $R_2^{s,c} = \emptyset$, $R_3^{w,c} = \emptyset$, $R_3^{s,c} = \{r_{32}\}$. Finally, for each set the output set, R_j^o is obtained by applying the multiple testing correction on the set of confirmed peaks R_j^c , and only ERs with a false discovery rate below a definable threshold α are kept. Let us suppose that this requirement is met by r_{32} ; the final outcome becomes: -3mm

$$\begin{aligned} R_1^o &= \emptyset \\ R_2^o &= \emptyset \\ R_3^o &= \{r_{32}\} \end{aligned}$$

Case 2: Biological Replicates

In case the samples are biological replicates, for this example let us set $C = 2 < J$. First our method determines the set of intersecting ERs for the first ER in the first sample, i.e., r_{11} : $R_{11,*} = (R_1 \cap_{\min} R_1) \cup (R_1 \cap_{\min} R_2) \cup (R_1 \cap_{\min} R_3)$. The condition on the minimum number of intersecting region $K = 3 \geq C = 2$ is met, and therefore the Fisher's method is applied to evaluate the combined evidence for that ER. Here, we assume that $X_{ji}^2 > \chi_{\gamma,2K}^2$, and therefore $r_{11} \in R_1^c$, $r_{21} \in R_2^c$ and $r_{32} \in R_3^c$.

For the second ER in the first sample, r_{12} , our method obtains $R_{12,*} = \{r_{12}, r_{21}\}$ (we remind that in case of multiple intersections with ERs in the same sample, like r_{21} and r_{22} here, only the strongest ER, i.e., r_{21} , is considered), which satisfies $K = 2 \geq C = 2$. Let us assume the combined p-value confirms the ERs of $R_{12,*}$ i.e., $r_{12} \in R_1^c$, $r_{21} \in R_2^c$.

In the second sample, $R_{21,*} = \{r_{21}, r_{12}, r_{32}\}$ also satisfies the condition $K = 3 \geq C = 2$. Again, let us assume that using the Fisher's method the ERs in $R_{21,*}$ are confirmed; thus, $r_{21} \in R_2^c$, $r_{12} \in R_1^c$ and $r_{32} \in R_3^c$.

For r_{22} , the set of intersecting peaks is $R_{22,*} = \{r_{22}, r_{12}\}$. With conditions $K = 2 \geq C = 2$ and $X_{ji}^2 > \chi_{\gamma,2K}^2$ satisfied, all these ERs are confirmed, i.e., $r_{22} \in R_2^c$ and $r_{12} \in R_1^c$.

Both r_{23} and r_{33} do not intersect any other ER, therefore, $R_{23,*} = \{r_{23}\}$ and $R_{33,*} = \{r_{33}\}$ fail to satisfy the condition $K = 1 \geq C = 2$ and the corresponding ERs are discarded, i.e., $r_{23} \in R_2^d$ and $r_{33} \in R_3^d$.

In the third sample, the set $R_{32,*} = \{r_{32}, r_{11}\}$ for r_{31} satisfies $k = 2 \geq C = 2$. Assuming that $X_{ji}^2 > \chi_{\gamma,2K}^2$, these ERs are confirmed, i.e., $r_{31} \in R_3^c$ and $r_{11} \in R_1^c$.

Finally, for r_{32} we have $R_{32,*} = \{r_{32}, r_{21}, r_{11}\}$, which satisfies $K = 3 \geq C = 2$. Assuming that $X_{ji}^2 > \chi_{\gamma,2K}^2$, also these ERs are confirmed, i.e., $r_{32} \in R_3^c$, $r_{21} \in R_2^c$ and $r_{11} \in R_1^c$. Thus, the final sets of ERs are:

$$\begin{array}{llll}
 R_1^{w,d} = \emptyset & R_1^{w,c} = \{r_{11}\} & R_1^{s,d} = \emptyset & R_1^{s,c} = \{r_{12}\} \\
 R_2^{w,d} = \{r_{23}\} & R_2^{w,c} = \{r_{21}, r_{22}\} & R_2^{s,d} = \emptyset & R_2^{s,c} = \emptyset \\
 R_3^{w,d} = \emptyset & R_3^{w,c} = \{r_{31}\} & R_3^{s,d} = \{r_{33}\} & R_3^{s,c} = \{r_{32}\}
 \end{array}$$

In the case of biological replicates, the ERs which were assigned both to the discarded set ($R_j^d = R_j^{w,d} \cup R_j^{s,d}$) and to the confirmed set ($R_j^c = R_j^{w,c} \cup R_j^{s,c}$) of a sample are relocated only to the latter set: $R_j^d = R_j^d \setminus \{R_j^c \cap R_j^d\}$. In this example, this modification leaves the previous sets unmodified.

Lastly, for each sample j , the output set R_j^o is obtained by applying the multiple testing correction on the confirmed set R_j^c , and only ERs with a false discovery rate below a definable threshold α are kept. If this requirement is met by each confirmed ER, the final result becomes:

$$\begin{array}{l}
 R_1^o = \{r_{11}, r_{12}\} \\
 R_2^o = \{r_{21}, r_{22}\} \\
 R_3^o = \{r_{31}, r_{32}\}
 \end{array}$$

6.5 Functional Annotation and Analysis of Enriched Regions

6.5.1 Motivation

An ER can overlap known genomic loci, like promoters or other regulatory elements of genes. Besides, a gene might be regulated by a transcription factor bound to a DNA regulatory element far from its promoter (e.g., regulatory elements called *enhancers* [186] can be located far from transcription start, like for the Sonic hedgehog (Shh) gene in mouse [187]), even interspersed with other non-regulated genes [188] [189]. MuSERA can efficiently assign an ER to the closest up-/down-stream genomic feature (e.g., gene Transcription Start Site (TSS), promoter region, Coding DNA Sequence (CDS), or enhancer list), thanks to its optimised implementation using binned data. Furthermore, MuSERA estimates the *ER-to-feature overlap score*, by determining the number of ERs intersecting with genomic annotations (i.e., known genes, 3'/5' Untranslated Regions (UTR), CDSs, Intergenic Regions (IGR), introns, promoter regions, etc.), or with any experimentally verified binding sites uploaded in MuSERA by the user through annotations in General Transfer Format (GTF) format. Additionally, it estimates the *ER-to-feature distance distribution* between the ERs and the closest up-/down-stream features per functional group. All these options allow to better evaluate the distribution of the ERs in the genomic context.

6.5.2 Algorithm

Once replicates are combined, MuSERA automatically annotates ERs with user-provided genomic features (e.g., genes, promoters, CDSs, binding sites of other transcription factors, etc.), which is by indexing data in Di3. MuSERA is an interactive tool, where the user can tune a

few parameters to achieve better results; hence, response time to update each annotation parameter is reasonably fast. MuSERA can linearly group ERs and known binding sites/genomic annotations that overlap; however, this would require re-running the algorithm in case of any user-defined parameter is changed. To avoid this, the functional analysis algorithm of MuSERA extends Di3, and indexes ERs and known binding sites/genomic annotations once a session is selected, which significantly improves the performance at changing user-defined parameters. In general, a snapshot of Di3 bookmarks all the ERs and genomic features overlapping a position on genome; hence it enables constant access for the biological interpretation of the snapshot. This aspect avoids re-running the annotation process in case of changing any user-defined annotation parameter, such as the filter option (e.g., considering only Transcription Factor Binding Sites or Coding DNA Sequences as known binding sites/genomic annotations). Additionally, given an ER, the corresponding DNA segments (i.e., snapshots) are determined in logarithmic time, because this requires a dichotomic search on first resolution of Di3, and the element annotations are determined in constant time; therefore, an ER annotation is optimally computed in $O(\log_2 n)$ for n snapshots. A pseudo code of functional analysis algorithm using Di3 is given in Algorithm 16.

Algorithm 16 Functional analysis algorithm based on Di3

```

1: procedure FUNCTIONALANALYSIS( $r_{ji}$ )
2:    $T \leftarrow f_{\text{map}}(r_{ji}, \mathcal{D}^+, \text{null})$  ▷ Does not pass a UDF to the MAP function.
3:    $T \leftarrow \text{update } T \text{ with user-defined annotation parameters}$ 
4:   return  $T$ 

```

6.5.3 Graphical User Interface

This tab helps assessing the distances between ERs and given genomic features (see Figure 6.7). To do so, i.e., to calculate the distance from set X to set Y, MuSERA requires the user to provide the sets X and Y, where X is a set of ERs and Y is a set of genomic features. For instance, X can be the set of all weak-confirmed and weak-discarded ERs on chr2 and chrX of all the samples of the session; and Y can be a selected source of genomic features.

6.6 Nearest Neighbor Distance Distribution

6.6.1 Motivation

MuSERA can compute the ER nearest neighbour distance distribution (NND); together with the supplied functional annotation and analysis features, this is particularly useful for ChIP-seq annotation. In each analysis session consisting of at least two samples, the ERs of each sample are grouped into different sets before (*stringent* or *weak* set) and after (*stringent confirmed*, *weak confirmed*, *stringent discarded*, and *output* set) the multiple-sample analysis. To estimate the NND, after the user chooses the desired sample(s) and set(s) to be considered, for each ER MuSERA determines the distance to the nearest ER; an option is available to treat

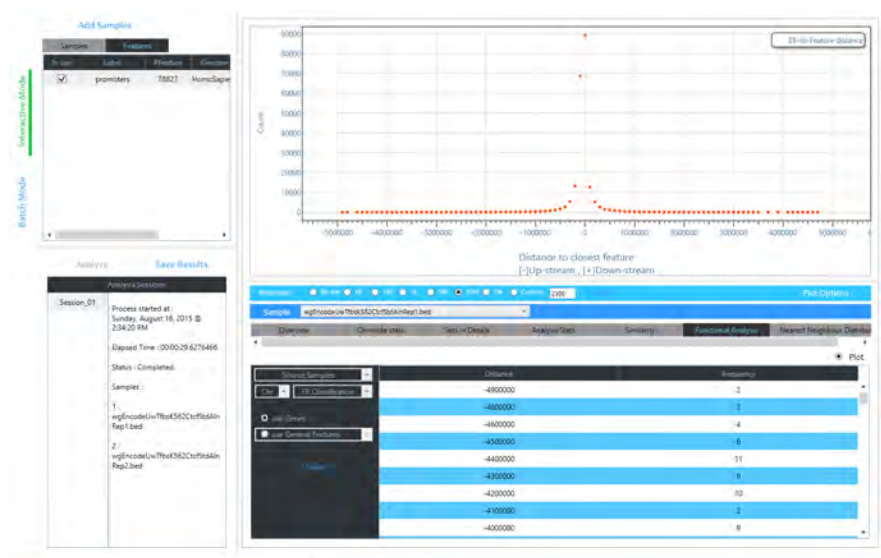


Figure 6.7: MuSERA features: functional annotation and analysis.

all selected samples and sets either as a single entity or as distinct entities. In the case of single entity, the closest neighbour of an ER could be an ER belonging to any set of any sample of the analysis session. In the case of distinct entities, the closest neighbour of an ER is determined within the same set and sample of the ER.

6.6.2 Algorithm

Once an analysis session is selected through the GUI of MuSERA, all ERs and user-provided genomic features (e.g., genes, promoters, CDSs, binding sites of other transcription factors, etc.) are automatically indexed in Di3. MuSERA is an interactive tool and aims for fastest response time for parameter changes. Therefore, to improve performance, the nearest neighbor (feature) distance distribution function of MuSERA leverages on *nearest neighbor* function of Di3. The *nearest neighbor* function of Di3 is defined at *semantic* level and is implemented over the functions of *logical* and *physical* levels. However, this design is agnostic to MuSERA which facilitates application of function; to calculate nearest feature distance distribution MuSERA defines a UDF and passes it to Di3, which is described by pseudo codes of Algorithm 17 and Algorithm 18.

6.6.3 Graphical User Interface

This tab provides features to estimate the distribution of distances between the ERs and their nearest neighbors (see Figure 6.8). The distance is the distance between an ER belonging to the selection and the closest ER also belonging to the selection. For instance, let us suppose that weak-confirmed and weak-discarded classifications are chosen. For each ER belonging to the weak-confirmed or weak-discarded classification, the distance to the closest weak-confirmed

Algorithm 17 Nearest feature distance distribution function of Di3

```

1: procedure NEAREST_NEIGHBOR_DISTANCE( $r_{ji}$ , UDF)
2:   find  $B_b | e_{b-1} < r_{ji} \leq e_b$ 
3:    $i \leftarrow 0$ 
4:   do
5:      $i \leftarrow i + 1$ 
6:      $d \leftarrow -1$ 
7:     if UDF( $@I \in B_{b-i}$ ) = true then
8:        $d \leftarrow e_b - e_{b-i}$ 
9:     if UDF( $@I \in B_{b+i}$ ) = true then
10:       $d \leftarrow \min(d, e_{b+i} - e_b)$ 
11:   while  $d \neq -1$ 
12:   return  $d$ 

```

Algorithm 18 Nearest feature distance distribution procedure of MuSERA

```

1: procedure NND_UDF( $\{@I_{ji}\}$ )
2:    $T \leftarrow \{@I_{ji}\}$ 
3:    $\triangleright @I_{ji}$  is pointer to any interval, and its type is determined using interval meta data
   which are organized by “Region metadata storage” component of MuSERA.
4:   if  $T$  contains any of user-specified features then
5:     return true
6:   else
7:     return false
8:
9: procedure NND( $r_{ji}$ )
10:  return  $f_{\text{NND}}(r_{ji}, \text{NND\_UDF})$ 

```

or weak-discarded ER will be determined. If the closest ER is upstream, the distance is between the start of the ER and the end of the neighbor. If the closest ER is downstream, the distance is between the end of the ER and the start of the neighbor. If the closest neighbor is overlapping the ER, then the distance is zero.

6.7 Global Correlation Assessment

6.7.1 Motivation

The similarity between replicates is frequently assessed either before peak calling, using genome-wide read densities, or after peak calling, using the identified enriched regions. Pearson’s product-moment correlation coefficient (PCC) [190] is a threshold-independent and scale-invariant method [160] commonly used to compute a global correlation assessment between replicates. PCC is also used after the peak calling when binned signal intensities are provided, either in a separate wiggle file per sample or as numerical vectors per identified ER (e.g., dataset chipseq mES of [191]). Besides, the Jaccard Similarity Coefficient (JSC) is a

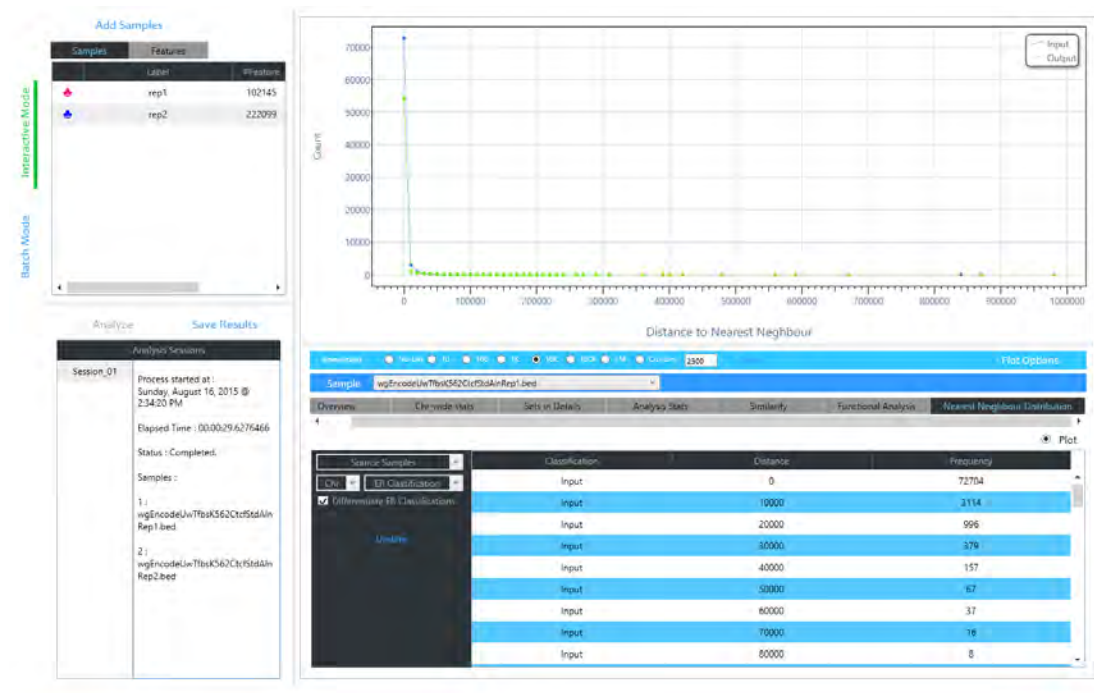


Figure 6.8: MuSERA features: Nearest Neighbor Distance Distribution.

statistical method for correlation/diversity assessment of samples, consisting on the ratio between the cardinalities of the intersection and the union of two sets; it can be used both as before peak calling (e.g., [192] increased genes detection power of RNA-seq data using JSC for global similarity filtering) and post peak calling (e.g., [193] [194]) correlation assessment procedure.

The similarity estimates help to better understand the different ER sets. For instance, a sample may contain a large number of ERs when peaks are called with a permissive p-value threshold, which are not necessarily co-localized with evidences on other replicates. Therefore, similarity between input sets is expected to be low; the similarity between discarded evidences is expected to be very low, while higher similarities are expected for confirmed ERs and output sets. A hierarchical representation of different classes of ERs along with their similarity indexes is provided in Figure 6.9 (MuSERA provides the information required to generate this figure, the figure is not generated by MuSERA itself).

MuSERA determines both region-level and base-pair-level correlations between all pairs of sets using JSC (see Figure 6.9). They are respectively computed as the ratio between the number of overlapping regions (region-level correlation), or genomic bases (base-pair-level correlation), and the total number of regions, or genomic bases, in the considered sets. Base-pair-level correlation is more stringent and is to be preferred when the position of the ERs is known with certainty, or when the experimental protocols have a low level of noise. Region-level correlation is instead more permissive, as it scores the overlap of entire regions rather than



Figure 6.9: During processing of two replicate samples, MuSERA estimates the Jaccard similarity coefficient between the two samples and for each of their computed ER sets. Values are shown for the ENCODE samples *wgEncodeSydhTfbsK562CmycIfna30StdAlnRep1* and *wgEncodeSydhTfbsK562CmycIfna30StdAlnRep2* (processed with analysis parameters: $BioRep$, $T^s = 1E-8$, $T^w = 1E-4$, $\gamma = 1E-8$, and $C = 1$), which overall show a rather low correlation (Input). In these samples the peaks are called using MACS2.0 [195] with 0.001 p-value threshold; hence, such low correlation is expected because of low signal-to-noise ratio. Initial classification of the ERs in each replicate (i.e., Stringent ERs vs. Weak ERs) confirms that in the replicates stronger evidences correlate better than weaker ones. Combining the samples, each of the two initial categories is divided into the Confirmed and Discarded sub-categories; ERs in the Confirmed sub-categories resulted to be considerably more correlated compared to their corresponding ERs in the Discarded sub-categories.

quantifying the magnitude of this overlap; this correlation measure is then to be preferred in presence of heterogeneous or noisy data sets.

Overall Similarity The overall similarity is computed across samples in terms of regions and base-pairs in common. For instance, the overall similarity between two replicates in Figure 6.10 is computed as follows:

1. **Base-pair-level:**

$$J = \frac{|\text{Rep}_1 \cap \text{Rep}_2|}{|\text{Rep}_1 \cup \text{Rep}_2|}$$

$$= \frac{(6 - 4) + (19 - 16) + (27 - 24) + (49 - 46) + (58 - 56)}{(10 - 2) + (30 - 13) + (37 - 33) + (43 - 40) + (51 - 46) + (60 - 54)} = 0.3$$

2. **Region-level:**

$$J = \frac{|\text{Rep}_1 \cap \text{Rep}_2|}{|\text{Rep}_1 \cup \text{Rep}_2|} = \frac{4 + 5}{6 + 5} = 0.81$$

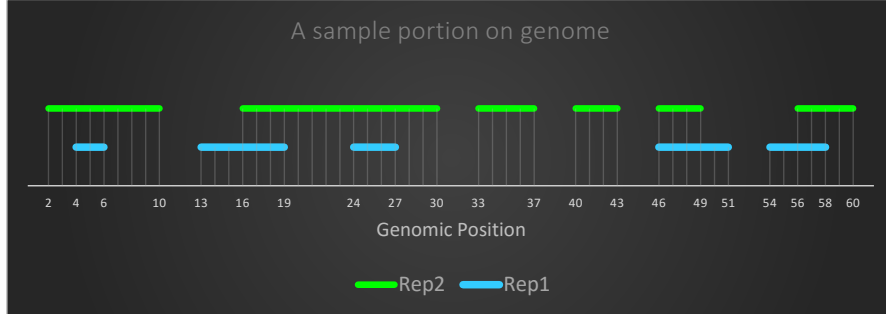


Figure 6.10: An example of a portion of genome with two replicates

Sample-wide Similarity

we compute similarities as follows:

1. How similar is Rep₁ to Rep₂ in Figure 6.10 ?

(a) **Base-pair-level:**

$$J = \frac{\sum |r_i| : r_i \in \text{Rep}_1 \wedge r_i \cap \text{Rep}_2 \neq \emptyset}{\sum |r_i| : r_i \in \text{Rep}_1}$$

$$= \frac{(6-4) + (19-16) + (27-24) + (49-46) + (58-56)}{(6-4) + (19-13) + (27-24) + (51-46) + (58-54)} = 0.65$$

(b) **Region-level:**

$$J = \frac{|\{r_i | r_i \in \text{Rep}_1 \wedge r_i \cap \text{Rep}_2 \neq \emptyset\}|}{|\text{Rep}_1|} = \frac{5}{5} = 1$$

2. How similar is Rep₂ to Rep₁ in Figure 6.10 ?

(a) **Base-pair-level:**

$$J = \frac{\sum |r_i| : r_i \in \text{Rep}_2 \wedge r_i \cap \text{Rep}_1 \neq \emptyset}{\sum |r_i| : r_i \in \text{Rep}_2}$$

$$= \frac{(6-4) + (19-16) + (27-24) + (49-46) + (58-56)}{(10-2) + (30-16) + (37-33) + (43-40) + (49-46) + (60-56)} = 0.3$$

(b) **Region-level:**

$$J = \frac{|\{r_i | r_i \in \text{Rep}_2 \wedge r_i \cap \text{Rep}_1 \neq \emptyset\}|}{|\text{Rep}_2|} = \frac{4}{6} = 0.6$$

The similarity indexes are plotted in Figure 6.11; as illustrated, for the sample-wide similarities, the similarity index of Rep₁-to-Rep₂ is higher than the index of Rep₂-to-Rep₁. These similarity indexes indicate that all Rep₁ ERs are intersecting with ERs of Rep₂ (region-level similarity of Rep₁-to-Rep₂ = 1; base-pair-level similarity of Rep₁-to-Rep₂ = 0.65 : 65% of base pairs of

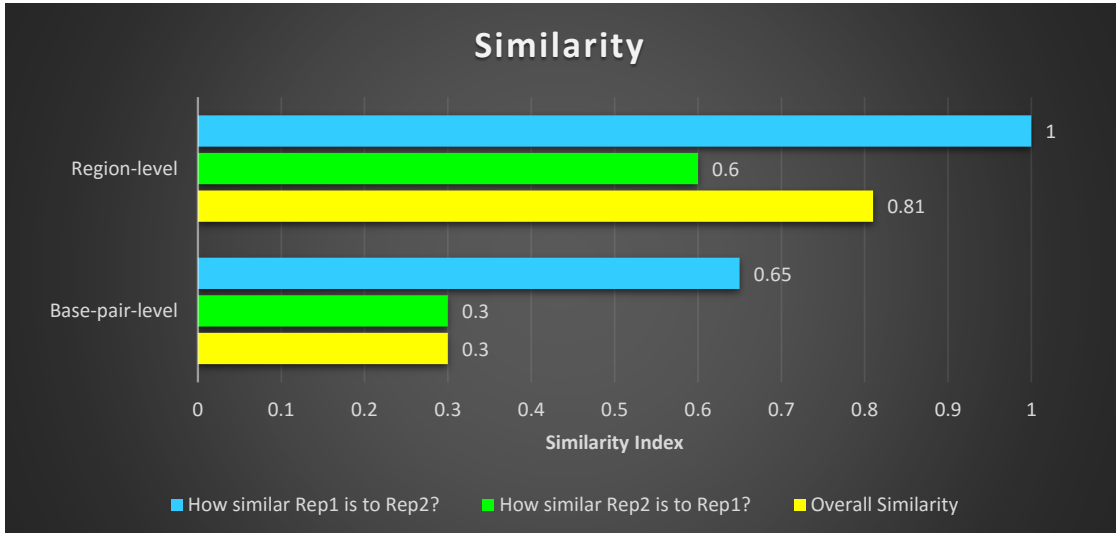


Figure 6.11: The Calculated similarity of Figure 6.10 example.

Rep₁ ERs are overlapping with Rep₂ base pairs), while Rep₂ has ERs not overlapping Rep₁ ERs (region-level similarity of Rep₂-to-Rep₁ = 0.6 : only 60% of Rep₂ ERs are overlapping Rep₁ ERs; base-pair-level similarity of Rep₂-to-Rep₁ = 0.3 indicates that only 30% the base-pairs of Rep₂ ERs are covered by Rep₁ ERs). The plot also highlights the differences between different similarity assessments.

6.7.2 Algorithm

MuSERA extends Di3 to assess global and sample-wide correlation between sample; it uses COVER and SUMMIT operations with UDFs as in the pseudo codes of Algorithm 19 and Algorithm 20.

6.8 Genome Browser

MuSERA implements a flexible and highly interactive set of plotting features based on the Dynamic Data Display [196] package, allowing real-time interactive zoom and pan on genome-

Algorithm 19 Sample-wide correlation assessment for sample j

- 1: **procedure** CORRELATION_REGIONLEVEL(j)
 - 2: $A \leftarrow \text{COVER}(J, J, \text{UDF_REGIONLEVEL})$
 - 3: $\text{return } \sum A / |S_j|$
 - 1: **procedure** CORRELATION_BASEPAIRLEVEL(j)
 - 2: $A \leftarrow \text{SUMMIT}(J, J, \text{UDF_BASEPAIRLEVEL})$
 - 3: $\text{return } \sum A / \sum_i (\bar{r}_{ji} - r_{ji})$
-

Algorithm 20 Global correlation assessment

```

1: procedure CORRELATION_REGIONLEVEL
2:    $A \leftarrow \text{COVER}(J, J, \text{UDF\_REGIONLEVEL})$ 
3:    $U \leftarrow \text{COVER}(1, J, \text{UDF\_REGIONLEVEL})$ 
    $\triangleright$  A and U are a set of numbers, where each number is cardinality of intervals at each
   region that satisfies COVER criteria.
4:   return  $\sum A / \sum U$ 

1: procedure CORRELATION_BASEPAIRLEVEL
2:    $A \leftarrow \text{SUMMIT}(J, J, \text{UDF\_BASEPAIRLEVEL})$ 
3:    $U \leftarrow \text{COVER}(1, J, \text{UDF\_BASEPAIRLEVEL})$ 
    $\triangleright$  A and U are a set of numbers, where each number is cardinality of base-pairs at
   each region that satisfies COVER/SUMMIT criteria.
4:   return  $\sum A / \sum U$ 

1: procedure UDF_REGIONLEVEL( $\{r_{ji}\}$ )
2:   return  $|\{r_{ji}\}|$ 

1: procedure UDF_BASEPAIRLEVEL( $\{r_{ji}\}$ )
2:   return  $\min \bar{r}_{ji} - \max r_{ji}$ 

```

scale samples. Having combined samples, MuSERA automatically creates bins independently for each of the determined sets (e.g., R_j^s , R_j^w , R_j^c , R_j^d , etc.), and displays in tabular format all the ERs of the sets with their corresponding information (e.g., chromosome, start, end, p-value, X^2 , etc.). By double-clicking on any of the listed ERs, MuSERA plots it together with all the ERs (in different colors according to the set they belong, i.e., stringent confirmed, stringent discarded, weak confirmed, or weak discarded) and annotations, if any, within a window of user-defined size (e.g., see Figure 6.12); then, this can be easily scrolled, panned and zoomed to interactively explore the location on the DNA also of all the other ERs and annotations.

6.8.1 Algorithm

The NEIGHBORS function of Di3 finds all intervals at d maximum distance from r reference interval, it passes the determined intervals to a UDF, and returns the output of the UDF (see Algorithm 21). Therefore, the output of NEIGHBORS function is UDF-specific, which could be a list of determined intervals, or an aggregation of an attribute of determined intervals. MuSERA uses the NEIGHBOR function to show all the ERs surrounding a given ER, hence it passes a UDF that determines a list of neighbor intervals to be plotted on the integrated genome browser.

6.9 Results

ChIP-seq enriched regions are read from data files in standard Browser Extensible Data (BED) format; besides standard ER format specifications (columns “chromosome”, “start”, “end”), we require a p-value quantifying the significance of each ER, which is usually computed by the

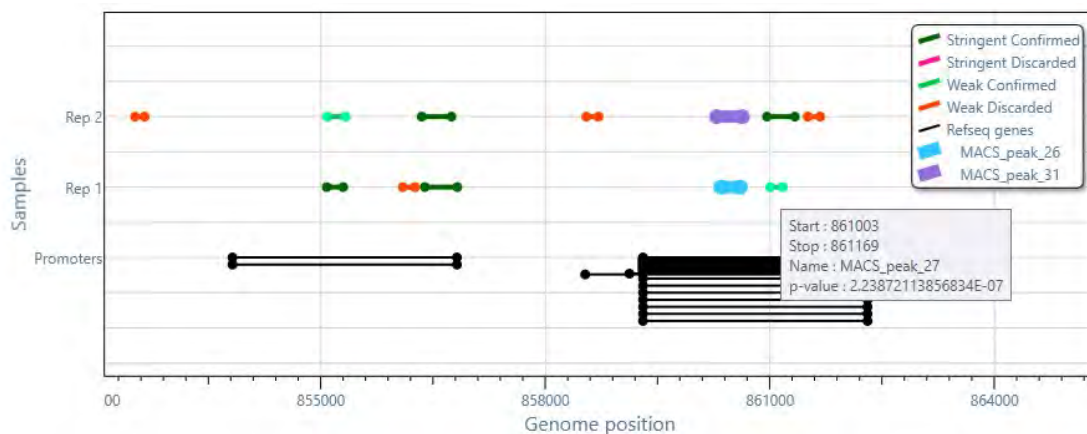


Figure 6.12: For a selected ER (e.g., the ER in light blue, named *MACS_peak_26*), the ER(s) it is combined with (e.g., the ER in purple, named *MACS_peak_31*) and all surrounding ERs (colored according to the set they belong) and available annotations are plotted; hovering the cursor on an ER, a tooltip is opened that shows the corresponding information (e.g., start, stop, name, p-value).

peak caller used to identify the ER. Binary Alignment/Map (BAM) files for the transcription factor Myc in human K562 cells (myelogenous leukaemia) were taken from the ENCODE project repository, for a total of 15 samples obtained in 7 different experiments as summarized in Table 6.2. Each experiment contained 2 or 3 biological replicates of the same ChIPseq. Technical replicates were artificially created to test our method, as they were not directly available in the ENCODE repository (see Section 6.9.1).

Peak calling was performed with the software package MACS2 [195] with the parameters

Algorithm 21 NEIGHBORS function of Di3

```

1: procedure NEIGHBORS( $r, d$ , UDF)
2:   find  $B_b | e_{b-1} < \underline{r} \leq e_b$ 
3:    $i \leftarrow 0$ 
4:    $T \leftarrow \emptyset$ 
5:   do
6:      $T \leftarrow T \cup \{@I\}_{b-i}$ 
7:      $T \leftarrow T \cup \{@I\}_{b+i}$ 
8:      $i \leftarrow i + 1$ 
9:   while  $i \leq d$ 
10:  return UDF( $T$ )

```

Algorithm 22 Integrated genome browser of MuSERA

```

1: procedure GENOMBROWSER_UDF( $T$ )
2:   return ( $T$ )

1: procedure GENOMBROWSER( $r, d$ )
2:    $T \leftarrow$  NEIGHBORS( $r, d$ , GENOMBROWSER_UDF)
3:   Plot  $T$ 

```

Chapter 6. Di3 Application in Comparative Analysis of ChIP-seq Replicates

Table 6.2: Datasets of transcription factor *Myc* in human K562 cells used for evaluation of comparative analysis.

Sample Name	Short Name	Aligned Reads	R^s	R^w
wgEncodeOpenChromChipK562CmycAlnRep1	Myc1_1	10,719,209	19,171	287,651
wgEncodeOpenChromChipK562CmycAlnRep2	Myc1_2	8,763,362	32,850	311,409
wgEncodeOpenChromChipK562CmycAlnRep3	Myc1_3	9,649,688	13,623	104,911
wgEncodeSydhTfbsK562CmycIggrabAlnRep1	Myc2_1	17,507,194	42,456	64,016
wgEncodeSydhTfbsK562CmycIggrabAlnRep2	Myc2_2	22,256,240	33,015	54,773
wgEncodeSydhTfbsK562CmycStdAlnRep1	Myc3_1	6,077,198	5,473	22,965
wgEncodeSydhTfbsK562CmycStdAlnRep2	Myc3_2	5,897,211	12,832	18,753
wgEncodeSydhTfbsK562CmycIfna30StdAlnRep1	Ifna30_1	10,115,596	1,901	13,654
wgEncodeSydhTfbsK562CmycIfna30StdAlnRep2	Ifna30_2	18,600,414	2,527	97,620
wgEncodeSydhTfbsK562CmycIfna6hStdAlnRep1	Ifna6h_1	9,377,798	5,852	12,087
wgEncodeSydhTfbsK562CmycIfna6hStdAlnRep2	Ifna6h_2	19,334,518	4,547	102,168
wgEncodeSydhTfbsK562CmycIfng30StdAlnRep1	Ifng30_1	11,602,299	8,227	13,190
wgEncodeSydhTfbsK562CmycIfng30StdAlnRep2	Ifng30_2	16,666,560	30,524	25,484
wgEncodeSydhTfbsK562CmycIfng6hStdAlnRep1	Ifng6h_1	14,019,564	2,485	13,376
wgEncodeSydhTfbsK562CmycIfng6hStdAlnRep2	Ifng6h_2	19,666,823	27,728	25,118

“-auto-bimodal”, “-p 0.0001” thus setting a p-value threshold of $1E-4$, and “-g hs”; using alignment files available in the ENCODE repository, together with the corresponding background (standard input for all samples except for Myc2, for which the input signal from rabbit IgG ChIP-seq was used). In each sample-input pair, the total number of reads was made equal by randomly down-sampling the largest alignment file. The performed call determined between $\sim 15K$ and $\sim 345K$ peaks across the different samples (see Table 6.2).

Adjustable parameters of the method are:

- T^s : maximum P-value to consider a peak as “stringent”
- T^w : maximum P-value to consider a peak as “weak”
- γ : maximum combined significance to confirm a peak
- C : minimum number of samples with intersecting peaks needed to apply the combined evidence evaluation
- α : maximum false discovery rate after the Benjamini-Hochberg correction
- the choice of “technical replicate” versus “biological replicate”

For our evaluations we used: $T^s = 1E-8$, $T^w = 1E-4$, $\gamma = 1E-8$, $\alpha = 0.05$ for all comparisons, $C = 1$ for biological replicates and $C = J$ for technical replicates. Required time was a few seconds for 2 samples with 100,000 peaks each on a standard desktop computer.

6.9.1 Simulated Technical Replicates

Artificial technical replicates were obtained by pooling alignment files of biological replicates in each experiment considered (in Table 6.2) and randomly splitting reads in two new artificial alignment files. Peaks were called with the software package MACS2 [195]. Stringent: ERs with p-value $< T^s$. Weak: ERs with $T^s \leq \text{p-value} < T^w$. $T^s = 1E-8$, $T^w = 1E-4$. Details about

Table 6.3: Statistics of Simulated Technical Replicates.

Sample Name	Short Name	Aligned Reads	R^s	R^w
wgEncodeOpenChromChipK562CmycAln_1	Myc1	14,566,130	22,928	13,770
wgEncodeOpenChromChipK562CmycAln_2	Myc1	14,566,130	23,175	13,664
wgEncodeSydhTfbsK562CmycIggrabAln_1	Myc2	19,881,717	2,457	3,858
wgEncodeSydhTfbsK562CmycIggrabAln_2	Myc2	19,881,717	2,466	3,891
wgEncodeSydhTfbsK562CmycStdAln_1	Myc3	5,987,205	1,267	1,559
wgEncodeSydhTfbsK562CmycStdAln_2	Myc3	5,987,295	1,270	1,564
wgEncodeSydhTfbsK562CmycIfna30StdAln_1	Ifna30	14,358,005	13,476	4,522
wgEncodeSydhTfbsK562CmycIfna30StdAln_2	Ifna30	14,358,005	13,364	4,581
wgEncodeSydhTfbsK562CmycIfna6hStdAln_1	Ifna6h	14,356,158	20,926	6,194
wgEncodeSydhTfbsK562CmycIfna6hStdAln_2	Ifna6h	14,356,158	21,054	6,212
wgEncodeSydhTfbsK562CmycIfng30StdAln_1	Ifng30	14,134,430	37,171	9,338
wgEncodeSydhTfbsK562CmycIfng30StdAln_2	Ifng30	14,134,430	37,146	9,333
wgEncodeSydhTfbsK562CmycIfng6hStdAln_1	Ifng6h	16,843,194	7,680	4,033
wgEncodeSydhTfbsK562CmycIfng6hStdAln_2	Ifng6h	16,843,194	7,695	3,969

technical replicates are collected in the Table 6.3. An alternative to our strategy to generate technical replicates would be to randomly split the reads in each original alignment file in replicates rather than merging biological replicates in ENCODE first. However, this procedure gives rise to a much poorer signal, preventing the identification of most ERs. The statistics of these alternative technical replicates are described in the Supplementary Table 6.4.

6.9.2 Evaluation of Technical Replicates

Technical replicates are used to evaluate and remove the noise introduced in the experimental procedure. In the case of ChIP-seq experiments, they are usually generated by performing the same ChIP protocol on the same biological sample, and then performing the sequencing independently. As the ENCODE datasets include only biological replicates, we tested our method on artificial technical replicates (see Section 6.9.1).

Results for $T^s = 1E - 8$, $T^w = 1E - 4$, $\gamma = 1E - 8$, $\alpha = 0.05$ and $C = 2$ are shown in Figure 6.13. For each replicate sample (panels A–G), we show two bars: the left bar (SS) represents the peaks called in a single sample analysis (R^s in green and R^w in red), while the right bar (MS) classifies the same peaks, according to the output of our algorithm, in the four sets described: $R^{s,c}$ (light green), $R^{w,c}$ (dark green), $R^{s,d}$ (light red) and $R^{w,d}$ (dark red).

As expected, the number of R^s stringent and R^w weak peaks called in the same technical replicates is always very similar, even if the absolute numbers differ significantly across the different conditions considered. Each output set has a consistent fraction of $R^{w,c}$ weak confirmed ERs, which ranges from 20% to 98% (mean 46%, standard deviation 30%) of the starting number of stringent peaks ($R^{w,c} / R^s$, panel H); thus by combining evidence in replicates, our method “rescues” (i.e. confirms) a large amount of weak co-localized peaks that would otherwise be discarded through a usual single sample evaluation. The percentage of stringent discarded peaks ($R^{s,d} / R^s$, panel H) is very low and varies from 0% in Myc2 to 12% in Myc3

Chapter 6. Di3 Application in Comparative Analysis of ChIP-seq Replicates

Table 6.4: Statistics of alternative simulated technical replicates.

Sample Name	Short Name	Aligned Reads	R^s	R^w
wgEncodeOpenChromChipK562CmycAlnRep1_1	Myc1_1_a	5,359,605	8,450	7,325
wgEncodeOpenChromChipK562CmycAlnRep1_2	Myc1_1_b	5,359,605	8,543	7,402
wgEncodeOpenChromChipK562CmycAlnRep2_1	Myc1_2_a	4,381,681	14,111	9,931
wgEncodeOpenChromChipK562CmycAlnRep2_2	Myc1_2_b	4,381,681	14,138	9,945
wgEncodeOpenChromChipK562CmycAlnRep3_1	Myc1_3_a	4,824,844	8,529	6,062
wgEncodeOpenChromChipK562CmycAlnRep3_2	Myc1_3_b	4,824,844	8,520	6,162
wgEncodeSydhTfbsK562CmycIggrabAlnRep1_1	Myc2_1_a	8,753,597	1,187	2,408
wgEncodeSydhTfbsK562CmycIggrabAlnRep1_2	Myc2_1_b	8,753,597	1,211	2,307
wgEncodeSydhTfbsK562CmycIggrabAlnRep2_1	Myc2_2_a	11,128,120	1,523	2,548
wgEncodeSydhTfbsK562CmycIggrabAlnRep2_2	Myc2_2_b	11,128,120	1,483	2,529
wgEncodeSydhTfbsK562CmycStdAlnRep1_1	Myc3_1_a	3,038,599	733	1,602
wgEncodeSydhTfbsK562CmycStdAlnRep1_2	Myc3_1_b	3,038,599	762	1,577
wgEncodeSydhTfbsK562CmycStdAlnRep2_1	Myc3_2_a	2,948,606	606	2,376
wgEncodeSydhTfbsK562CmycStdAlnRep2_2	Myc3_2_b	2,948,606	575	2,515
wgEncodeSydhTfbsK562CmycIfna30StdAlnRep1_1	Ifna30_1_a	5,057,798	919	1,512
wgEncodeSydhTfbsK562CmycIfna30StdAlnRep1_2	Ifna30_1_b	5,057,798	957	1,516
wgEncodeSydhTfbsK562CmycIfna30StdAlnRep2_1	Ifna30_2_a	9,300,207	14,244	3,845
wgEncodeSydhTfbsK562CmycIfna30StdAlnRep2_2	Ifna30_2_b	9,300,207	14,127	3,912
wgEncodeSydhTfbsK562CmycIfna6hStdAlnRep1_1	Ifna6h_1_a	4,688,899	3,369	1,979
wgEncodeSydhTfbsK562CmycIfna6hStdAlnRep1_2	Ifna6h_1_b	4,688,899	3,404	1,924
wgEncodeSydhTfbsK562CmycIfna6hStdAlnRep2_1	Ifna6h_2_a	9,667,259	19,815	5,796
wgEncodeSydhTfbsK562CmycIfna6hStdAlnRep2_2	Ifna6h_2_b	9,667,259	19,742	5,708
wgEncodeSydhTfbsK562CmycIfng30StdAlnRep1_1	Ifng30_1_a	5,801,150	28,287	8,958
wgEncodeSydhTfbsK562CmycIfng30StdAlnRep1_2	Ifng30_1_b	5,801,150	28,386	8,885
wgEncodeSydhTfbsK562CmycIfng30StdAlnRep2_2	Ifng30_2_a	8,333,280	17,339	5,253
wgEncodeSydhTfbsK562CmycIfng30StdAlnRep2_2	Ifng30_2_b	8,333,280	17,370	5,170
wgEncodeSydhTfbsK562CmycIfng6hStdAlnRep1_1	Ifng6h_1_a	7,009,782	2,027	1,749
wgEncodeSydhTfbsK562CmycIfng6hStdAlnRep1_2	Ifng6h_1_b	7,009,782	2,098	1,740
wgEncodeSydhTfbsK562CmycIfng6hStdAlnRep2_1	Ifng6h_2_a	9,833,412	6,030	3,408
wgEncodeSydhTfbsK562CmycIfng6hStdAlnRep2_2	Ifng6h_2_b	9,833,412	6,603	3,366

Table 6.5: P-values for the enrichment of the E-box in technical replicates.

Sample	R^s	R^o	$R^{w,c}$	$R^{s,d}$
Myc1_a	1.1e-802	4.2e-925	3.1e-120	-
Myc1_b	1.5e-847	1.3e-931	1.1e-143	-
Myc2_a	4.3e-976	4.7e-1113	2.1e-125	-
Myc2_b	2.7e-951	6.5e-1120	1.6e-125	-
Myc3_a	1.1e-495	6.6e-536	5.9e-072	-
Myc3_b	2.3e-461	2.2e-535	1.6e-075	-
Ifna30_a	6.8e-170	3.7e-220	4.9e-079	-
Ifna30_b	1.1e-169	7.5e-241	1.0e-075	-
Ifna6h_a	4.2e-296	4.8e-418	1.6e-161	-
Ifna6h_b	2.3e-306	1.0e-427	4.7e-136	-
Ifng30_a	1.7e-813	8.1e-910	3.2e-076	4.0e-021
Ifng30_b	2.1e-834	1.2e-891	6.3e-097	4.7e-023
Ifng6h_a	6.2e-895	9.9e-942	7.0e-101	1.6e-038
Ifng6h_b	1.1e-886	3.1e-965	1.6e-098	2.4e-022

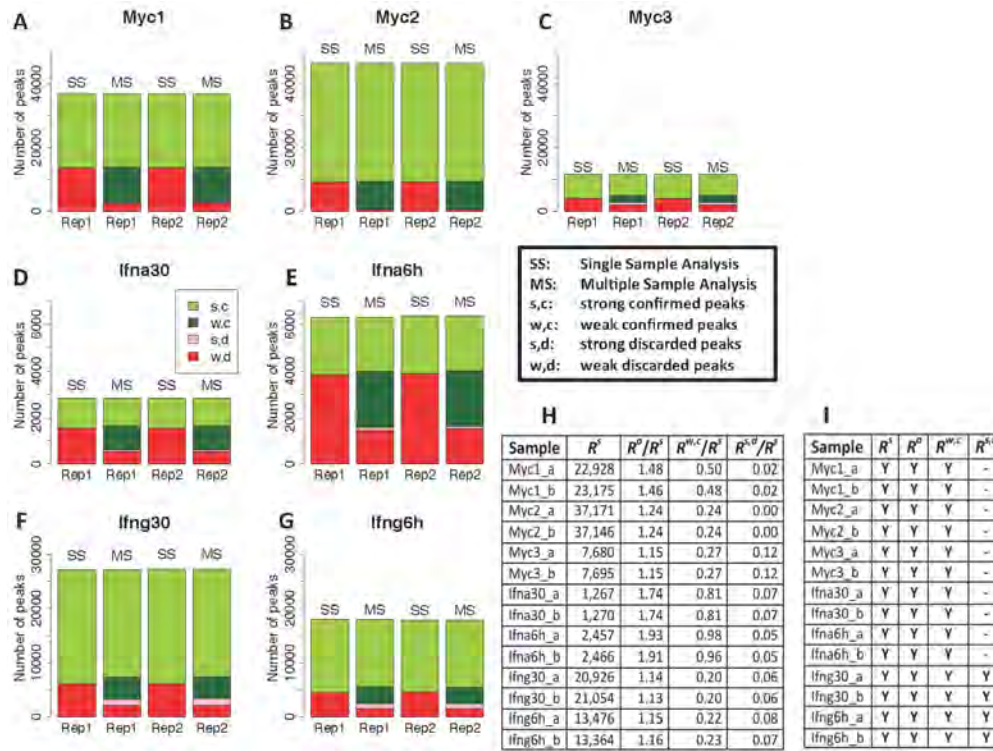


Figure 6.13: Technical replicates. For each of the 7 experiments considered, two technical replicates were obtained by pooling reads from the biological replicates of the conditions and then randomly splitting the resulting alignment files in two equal parts. **A-G:** ER sets for the technical replicates considered. **SS**, single sample analysis; **MS**, multiple sample analysis. In each panel, the **SS** stacked bars represent R^s (green) and R^w (red) in the two replicates, while the **MS** bars show the same peaks, confirmed or discarded according to the output of our method: $R^{s,c}$ (light green), $R^{w,c}$ (dark green), $R^{s,d}$ (light red) and $R^{w,d}$ (dark red). **H**, general statistics on the cardinality of the ER sets; **I**, validation of the sets with the Myc binding motif (Myc canonical E-box); “Y”, presence of the E-box; “-”, set too small to find any enriched motif. See Table 6.5 for E-box enrichment p -values.

(mean 5.6%, standard deviation 3.8%). The output set R^o corresponds to the set of confirmed peaks $R^c = R^{s,c} \cup R^{w,c}$, where the significance of each peak has been adjusted for multiple testing; combining the evidence present in replicates increases the number of obtained peaks up to almost the double of what obtained with a single sample at the same stringency (R^o/R^s , panel H).

For technical replicates, we expect the output of each replicate to be similar, and therefore the parameter C was set to $C = J = 2$ for all technical replicate comparisons. Setting $C = 1$ would be instead equivalent to trust even isolated peaks, which are not present in the other replicate. With the latter choice, and $\gamma = T^s$, no stringent peaks would be discarded.

As a preliminary evaluation of the results obtained, we considered the overlap of the peaks with the enriched regions in DNase-seq data. On average, 95.4% of the peaks in R^o , 95.4% of peaks in R^s and 94.6% of peaks in $R^{w,c}$ were in open chromatin regions, while this fraction was only 89.4% for $R^{s,d}$ and 93.0% for $R^{w,d}$ (see Table 6.6). The overlap with open chromatin,

Table 6.6: Percentages of ERs overlapping with DNase-seq data in technical replicates.

Sample	R^s	R^o	$R^{w,c}$	$R^{s,d}$	$R^{w,d}$
Myc1_a	94.5	90.2	81.1	70.4	70.4
Myc1_b	94.4	90.2	80.9	74.7	72.0
Myc2_a	98.0	97.9	97.2	96.3	97.8
Myc2_b	98.0	97.9	97.2	98.4	96.7
Myc3_a	86.1	88.7	93.0	77.8	93.8
Myc3_b	86.4	88.7	93.3	80.1	93.8
Ifna30_a	95.9	96.6	96.7	87.5	97.3
Ifna30_b	95.4	96.7	97.3	86.7	95.5
Ifna6h_a	97.6	98.0	98.0	94.6	97.7
Ifna6h_b	95.9	98.0	98.2	94.9	97.0
Ifng30_a	99.7	99.6	99.1	99.6	99.1
Ifng30_b	99.7	99.6	99.2	99.7	99.5
Ifng6h_a	96.8	96.9	96.2	94.5	95.3
Ifng6h_b	96.7	96.8	96.4	94.0	96.3

however, is not yet a validation of a specific binding event. We performed then motif analysis on the nucleotide sequences corresponding to the ERs in the four sets: R^s , R^o , $R^{w,c}$ and $R^{s,d}$. Myc is known to bind a large number of sites on the DNA, particularly with high affinity to those with the 6-nucleotide motif called Enhancer-box or E-box. This protein-binding region has the generic consensus nucleotide sequence *CANNTG* (with N representing any nucleotide [197]). In particular, Myc binds with maximum strength to the *CACGTG* motif (also called the “canonical” Myc E-box [198]). Therefore, we consider the enrichment of the Ebox in a set of peaks a sufficient condition to consider the set as containing “true” binding sites. Panel I in Figure 6.13 shows that the E-box is always enriched in the R^s stringent and R^o output sets, as well as in the $R^{w,c}$ weak confirmed set. This result confirms that in the large majority of cases the weak peaks overlapping in replicates identify real binding sites, which are missed by a stringent single-sample call. In 4 out of 14 cases, the $R^{s,d}$ stringent discarded set is enriched for the E-box, although at much lower significance (see Table 6.5), while in the remaining cases the number of peaks in the $R^{s,d}$ set is low. This analysis suggests that the default value $C = J$ used for our artificial technical replicates may be too conservative and still discards a small fraction of real binding sites.

6.9.3 Evaluation of Biological Replicates

The ENCODE data repository always includes one or more biological replicates for each ChIP-seq experiment. For the transcription factor Myc, multiple data sources are available, either obtained in independent experiments, or scored against different backgrounds (in Myc2, the input was derived from immuno-precipitating normal rabbit IgG, while in all the other samples the standard input for the K562 cell line was used). We applied our method to biological replicates obtained from each of the ENCODE experiments considered, and we also combined replicates from 2 experiments (Myc2 and Myc3). Parameters for the method were

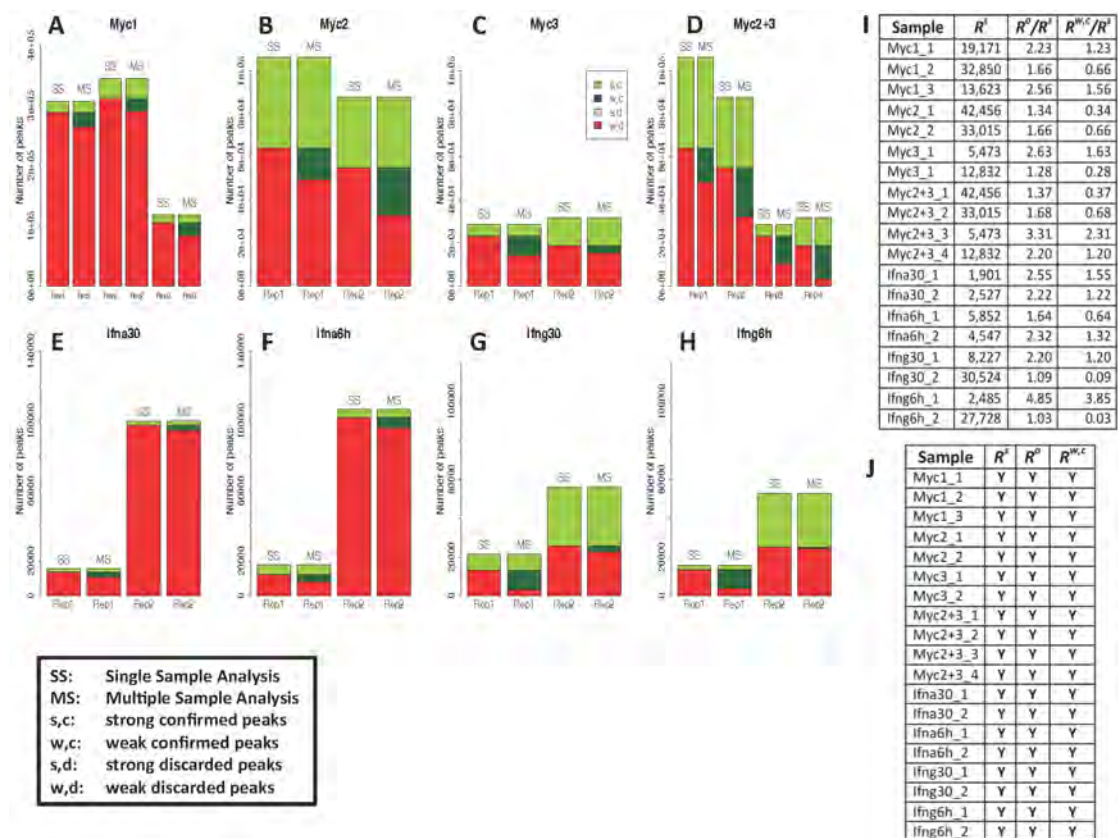


Figure 6.14: **Biological replicates.** A–H: ER sets in the biological replicates considered. SS, single sample analysis; MS, multiple sample analysis. In each panel, the SS stacked bars represent R^s (green) and R^w (red) in the replicates, while the MS bars show the same peaks, confirmed or discarded according to the output of our method: $R^{s,c}$ (light green), $R^{w,c}$ (dark green), $R^{s,d}$ (light red) and $R^{w,d}$ (dark red). I, general statistics on the cardinality of the ER sets; J, validation of the sets with the Myc binding motif (Myc canonical E-box); “Y”, presence of the E-box. See Table 6.8 for E-box enrichment P-values.

the same as for the technical replicate evaluation reported in the Section 6.9.2 (i.e. $T^s = 1E - 8$, $T^w = 1E - 4$, $\gamma = 1E - 8$, $\alpha = 0.05$); for the additional parameter C , in the case of biological replicates we adopted the permissive choice of $C = 1$ (default for the analysis of biological replicates). With these values (i.e. $\gamma = T^s$ and $C = 1$), our method never discards a stringent peak (we consider that a single strong evidence is enough for biological replicate evaluation). Results are shown in Figure 6.14.

The number of peaks in biological replicates of the same experiment can be very different (panels A–H), reflecting the different efficiency of the ChIP-seq protocol, and the number of weak peaks (R^w) is usually much larger than the number of stringent peaks (R^s). In the considered cases, the number of confirmed weak peaks ($R^{w,c}$) is often much bigger (up to ~4 times) than the number of stringent peaks (R^s) (column $R^{w,c}/R^s$ in panel I), confirming that the evidence in a “good” replicate allows the rescue of many peaks in a “bad” replicate. We observe a similar situation when we combine samples obtained with different inputs. For example, by combining together the four replicates of the Myc2 and Myc3 cases (panel D),

Chapter 6. Di3 Application in Comparative Analysis of ChIP-seq Replicates

Table 6.7: Percentages of ERs overlapping with DNase-seq data in biological replicates ($C = 1$). ChIP-seq ERs contained in each of the sets were overlapped with open chromatin regions obtained from DNase-seq experiments. The average percentage of overlapping peaks in the weak confirmed sets (91.0%) is comparable to what found for the stringent peaks (88.9%) or in the output peak sets (91.1%) and higher than what found for the weak discarded peaks (51.6%). R^s : stringent ER set. R^o : output ER set. $R^{w,c}$: weak confirmed ER set. $R^{s,d}$: stringent discarded ER set. $R^{w,d}$: weak discarded ER set.

Sample	R^s	R^o	$R^{w,c}$	$R^{w,d}$
Myc1_1	77.1	79.6	81.7	13.1
Myc1_2	70.2	73.9	79.4	16.4
Myc1_3	98.3	92.9	89.4	23.9
Myc2_1	97.5	96.3	93.0	61.8
Myc2_2	97.3	96.5	95.3	51.7
Myc3_1	61.8	79.5	90.4	36.6
Myc3_2	92.6	86.3	63.7	87.2
Myc2_3_1	97.5	96.2	92.9	61.0
Myc2_3_2	97.3	96.4	95.2	51.0
Myc2_3_3	61.8	83.1	92.3	14.8
Myc2_3_4	92.6	91.2	90.0	48.8
Ifna30_1	94.6	92.9	91.8	83.2
Ifna30_2	73.2	83.9	92.7	14.7
Ifna6h_1	98.1	97.4	96.4	93.2
Ifna6h_2	90.9	94.5	97.2	15.3
Ifng30_1	99.6	99.4	99.2	91.8
Ifng30_2	99.5	99.4	98.2	93.1
Ifng6h_1	95.5	96.5	96.7	39.6
Ifng6h_2	94.1	94.1	93.2	83.2

we increase massively the number of peaks in the output set for the samples with lower peak counts (Myc3) by confirming a number of their weak peaks much larger than in the evaluation performed without Myc2. Therefore, the presence of high-quality replicates can be of great help in improving the call on many low-quality replicates. The average overlap with open chromatin regions of the $R^{w,c}$ weak confirmed sets is 91.0% (compared with the 51.6% for the weak discarded peaks), and motif analysis confirms that in all the samples the ERs contain the canonical Myc binding site (panel J and Table 6.7 and Table 6.8).

We applied our method also using $C = 2$ on all biological replicates considered: most of the times that the $R^{s,d}$ stringent discarded peak set had a substantial size, the E-box motif was present, although the average overlap with open chromatin was only 75.9% (see Figure 6.16, Table 6.9 and Table 6.10). This further confirms that, in biological replicates, a lack of overlapping with peaks in other replicates does not necessarily correspond to an artifactual interaction.

Table 6.8: *P*-values for the enrichment of the E-box in biological replicates ($C = 1$). *P*-values for the enrichment of the E-box in the different peak sets, as estimated with the DREME package [199]. As the E-box is represented by several Position Weight Matrices (PWMs) in the Jaspar Core Vertebrata database (see Figure 6.15), we chose the best *p*-value among those obtained for the PWMs considered. R^s : stringent ER set. R^o : output ER set. $R^{w,c}$: weak, confirmed ER set.

Sample	R^s	R^o	$R^{w,c}$
Myc1_1	3.1e-627	1.5e-886	1.4e-283
Myc1_2	7.5e-563	3.5e-795	2.8e-235
Myc1_3	6.3e-508	4.5e-788	4.8e-326
Myc2_1	7.3e-987	1.3e-1198	3.0e-240
Myc2_2	5.7e-832	8.8e-1164	3.0e-334
Myc3_1	4.9e-266	1.9e-479	1.8e-216
Myc3_2	7.3e-548	1.4e-603	1.4e-083
Myc2_3_1	7.3e-987	5.8e-1099	7.3e-244
Myc2_3_2	5.7e-832	2.0e-1123	2.2e-345
Myc2_3_3	4.9e-266	7.1e-547	6.1e-270
Myc2_3_4	7.3e-548	6.7e-806	3.4e-313
Ifna30_1	1.9e-222	2.1e-370	2.0e-146
Ifna30_2	1.6e-183	1.1e-330	1.8e-156
Ifna6h_1	1.1e-483	1.1e-654	1.1e-188
Ifna6h_2	1.2e-374	3.2e-561	1.1e-188
Ifng30_1	6.3e-711	8.9e-1038	1.9e-354
Ifng30_2	4.2e-786	5.4e-876	7.3e-144
Ifng6h_1	2.0e-232	5.4e-479	6.4e-254
Ifng6h_2	2.6e-1321	1.5e-1276	5.5e-014

Table 6.9: Percentages of ERs overlapping with DNase-seq data in biological replicates ($C = 2$). ChIP-seq ERs contained in each of the sets were overlapped with open chromatin regions obtained from DNase-seq experiments. The average percentage of overlapping peaks in the weak confirmed sets (92.2%) is comparable to what found for the stringent peaks (91.5%) or in the output peak sets (94.5%) and higher than what found for the stringent discarded peaks (75.9%) and weak discarded peaks (50.0%). R^s : stringent ER set. R^o : output ER set. $R^{w,c}$: weak confirmed ER set. $R^{s,d}$: stringent discarded ER set. $R^{w,d}$: weak discarded ER set.

Sample	R^s	R^o	$R^{w,c}$	$R^{s,d}$	$R^{w,d}$
Myc1_1	77.1	95.7	92.9	42.9	13.1
Myc1_2	98.3	95.6	92.3	93.4	16.4
Myc1_3	98.3	93.1	89.4	84.0	23.8
Myc2_1	97.5	96.7	93.0	93.5	61.8
Myc2_2	97.3	96.9	95.3	83.5	51.7
Myc3_1	61.8	83.8	90.4	16.1	36.6
Myc3_2	92.6	81.8	63.7	96.6	87.2
Ifna30_1	94.6	93.3	91.8	88.9	83.2
Ifna30_2	73.2	93.2	92.7	41.9	14.8
Ifna6h_1	98.1	97.4	96.4	97.3	93.2
Ifna6h_2	90.9	97.5	97.2	62.8	15.2
Ifng30_1	99.6	99.4	99.2	97.1	91.8
Ifng30_2	99.5	99.4	98.2	99.4	93.1
Ifng6h_1	95.5	97.0	96.7	48.5	39.6
Ifng6h_2	94.1	97.0	93.2	92.8	83.2

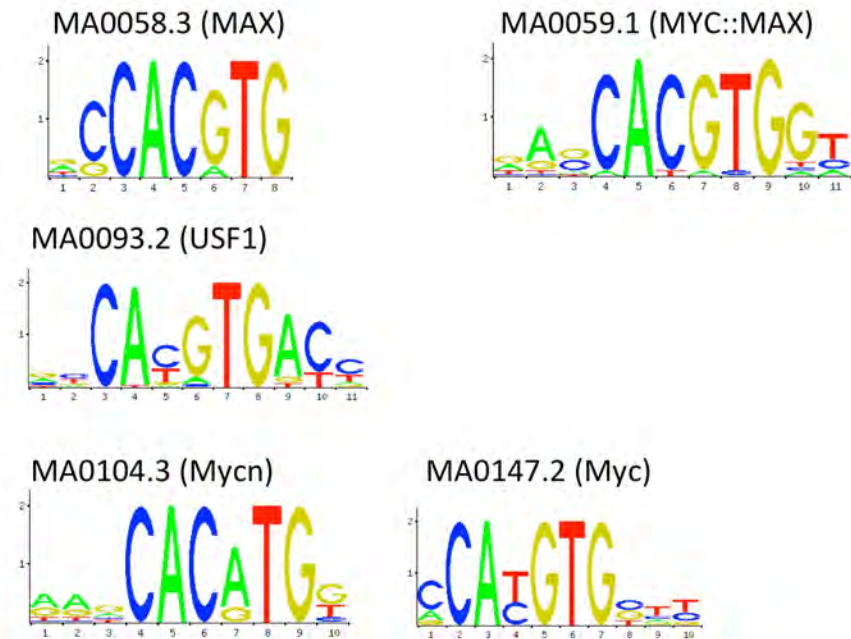


Figure 6.15: Sequence logos for the Position Weight Matrices (PWMs) from the Jaspas Core Vertebrata database used to validate Myc peaks.

Table 6.10: P-values for the enrichment of the E-box in biological replicates ($C = 2$). P-values for the enrichment of the E-box in the different peak sets, as estimated with the DREME package [199]. As the E-box is represented by several Position Weight Matrices (PWMs) in the Jaspas Core Vertebrata database (see Figure 6.15), we chose the best p-value among those obtained for the PWMs considered. R^s : stringent ER set. R^o : output ER set. $R^{w,c}$: weak confirmed ER set. $R^{s,d}$: stringent discarded ER set. “-”: set too small to find any enriched motif. “N”: E-box not enriched.

Sample	R^s	R^o	$R^{w,c}$	$R^{s,d}$
Myc1_1	3.1e-627	3.7e-878	-	-
Myc1_2	7.5e-563	1.7e-792	2.8e-235	N
Myc1_3	2.9e-503	9.5e-780	2.2e-321	2.9e-008
Myc2_1	6.3e-508	2.0e-784	4.8e-326	8.0e-013
Myc2_1	7.3e-987	2.5e-1125	6.6e-245	4.7e-126
Myc2_2	5.7e-832	6.6e-1131	3.0e-334	2.4e-027
Myc3_2	4.9e-266	2.5e-473	1.8e-216	-
Ifna30_2	1.6e-183	5.6e-348	4.2e-161	7.8e-016
Ifna30_1	1.9e-222	1.8e-336	2.0e-146	3.9e-031
Ifna6h_1	1.1e-483	8.0e-583	1.1e-188	2.9e-065
Ifna6h_2	1.2e-374	1.1e-555	3.4e-223	2.9e-023
Ifng30_1	6.3e-711	4.5e-984	1.9e-354	N
Ifng30_2	4.2e-786	2.6e-839	1.3e-113	2.9e-175
Ifng6h_1	2.0e-232	1.3e-447	6.4e-254	-
Ifng6h_2	2.6e-1321	3.3e-442	5.5e-014	2.0e-893

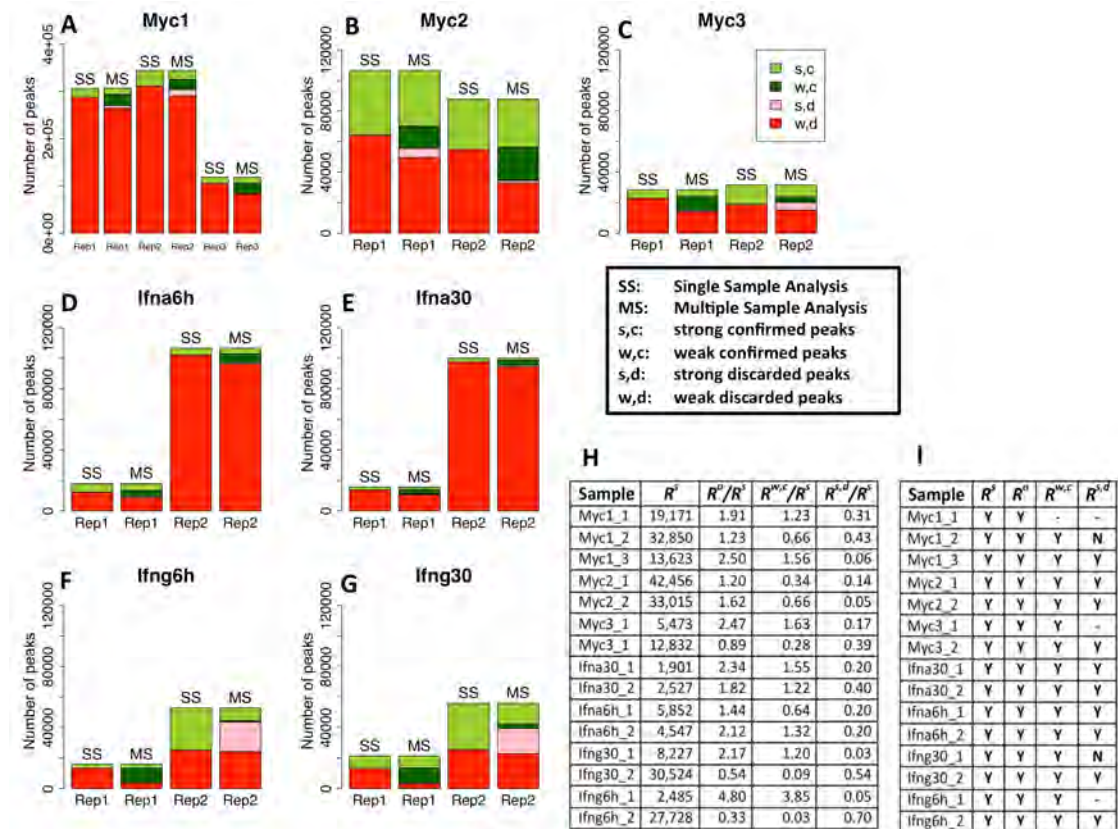


Figure 6.16: A-G: ER sets in the biological replicates considered, obtained with $C = 2$. SS: single sample analysis; MS: multiple sample analysis. In each panel, the SS stacked bars represent R^s (light green) and R^w (red) in the replicates, while the MS bars show the same peaks, confirmed or discarded according to the output of our method: $R^{s,c}$ (light green), $R^{w,c}$ (dark green), $R^{s,d}$ (pink) and $R^{w,d}$ (red). H: general statistics on the cardinality of the ER sets. I: validation of the sets with the Myc binding motif (Myc canonical E-box); “Y”: presence of the E-box; “N”: absence of the E-box; “-”: set too small to find any enriched motif (see Supplementary Table 8 for E-box enrichment p -values).

7 Toward Google-Style Search in Genomics

7.1 Introduction

Understanding genome regulation is a main target in cancer and other complex diseases. Many different combinations of chromatin conformation, binding, histone modifications, and gene expression patterns, determine genome's response to the environment and to internal stimuli. We call these combinations genomic regulatory signatures. Genomic regulatory signatures, detected as a combination of high-throughput experimental signals, may appear incomplete or even biologically irrelevant if not considered in their context. The context is represented by a physical (including mechanical) or chemical stimulus, affecting the signature characteristics and thus the genomic response to it. The context can be both external, when a stimulus is applied from outside, such as drug induction (i.e. ligand-receptor interaction); or internal, due to the activity of a trans-agents (e.g. transcription factor/cofactor), the presence of an active distal regulatory element (e.g. an enhancer), or a cis-regulatory element, such as a promoter-binding motif. Depending on the context, biological signals composing a signature may vary in relative distances, amplitude, statistical significance, and signal shape. For instance, H3K4me1 signal is typically unimodal and very high at enhancers, but bimodal surrounding promoters; while H3K4me3 signal is unimodal and high at active promoters, but almost absent at enhancers, misregarding the activation status. Another example is represented by those trans-agents, like Mediator, capable of mediating the promoter-enhancer interaction, or the capability of some factors, like CTCF, of creating chromatin domains. Beside basic known functions, genomic signals are still not fully understood, especially considering their possible interactions (i.e. the most informative part).

Since the heterogeneous semantic of different high-throughput experiments, and their intrinsic noise, often difficult to divide from true signal, a robust method capable of recognizing signal patterns (i.e. signatures) above the background is highly requested by the community. Given the complexity of the problem of recognizing informative signatures in the genome (i.e. signatures that could explain genomic regulation), a divide-et-impera method is often the winning strategy. Explain a subset of regulatory functions allow us to better reduce the impact

of noise on true signal (signal-to-noise ratio is a local property), and to better define a less complex and more clear mechanism. We call these local patterns as landmarks, and we take them as examples (“training”) for our similarity search.

For instance, consider a dataset of related samples (e.g., replicates, or samples of a specific cell-line) of Transcription Factors (TF), Histone Modifications (HM) and RNASeqs, targeting a particular experiment, where the samples are pre-processed to preserve only the highly reliable evidences. The samples may have a degree of similarity between each other, and between binding sites (i.e., in binding profile); for instance the samples may in general show relatively high activity on some promoter regions. In other words, the binding pattern could possibly be replicated through the samples, and the replicated regions are the landmarks defining regions of interest to which the process should be scoped. It is of high interest to identify the set of samples in repository (e.g., ENCODE data) that have high degree of similarity in binding profile to the query dataset. This step is currently done manually and through visual inspection on few predicted samples that are selected based a researcher’s personal experience. Also, this procedure currently requires the candidate datasets to be processed (cleaned) using similar pipeline that applied on query dataset. Note that, the cleaning procedure could possibly differ from one study to another targeting different regions of interest that possibly would necessitate pre-processing the candidates according to query pre-processing setup.

So the research question would be: identify datasets in the repository that have similar binding profile to query dataset with a degree of similarity, without the necessity of cleaning the repository the same way the query dataset was cleaned. Additionally, it should be noted that there exist a many-to-many relationship between samples of query and target, i.e., one reference interval can be close to multiple intervals from other samples, and vice versa. Moreover, the samples of the query may possibly be assigned a weight for each indicating the importance of contribution of the binding profile from that sample to overall binding signature.

7.2 Pattern Finding Queries on Di4

Present section provides means to define regions and metrics of interest, and to search for regions and samples that satisfy the criteria using a query. For this purpose, two orthogonal approaches are defined, *Samples Pattern Rank* (SPR) (see Section 7.2.1), and *Regions Pattern Rank* (RPR) (see Section 7.2.2). The former ranks regions based on a defined pattern, while the latter ranks samples based on another defined pattern. For formal definition of RPR and SPR we use following syntax:

1. A sample ID:
`<SID> := <reference to sample>`
2. Coordinates of regions are defined as follows:

- a. Left-end (start) is an integer referring to a base-pair on genome:
`<L> := <integer for left-end>`
- b. Right-end (stop) is an integer referring to a base-pair of genome:
`<R> := <integer for right-end>`
- c. Summit is an integer referring to a base-pair on genome:
`<S> := <integer for summit>`
3. A region:
`REGION> := (<L> <R> [<S>])`
4. A window that defines extension to left and right of a region:
`<WINDOW> := (<int> <int>) | ()`
5. A set of regions (where an empty set could mean every region):
`<REGION_LIST> := (<REGION>+)`
6. A sample list: `<SAMPLES_LIST> := (<SID>+) | ((<SID><SID>+)+)`
7. A select statement is defined as follows:
`<SELECT_STATEMENT> :=
 (SELECT ((<attribute> <value>)*)
 [(AGGREGATE <AGGREGATE_FUNCTION>)]
 [(PATTERN <PATTERN_FUNCTION> [<PATTERN_FUNCTION>])]
 [(WEIGHT <float> [<float>])])`
8. Rank statement is defined as follows:
`<RANK_STATEMENT> :=
 RANK_SAMPLES_INDEPENDENT |
 RANK_ALL_BY_PEARSON |
 RANK_ALL_BY_COSINE`
9. We define a comparison operator for rank statement as follows:
`<int> ← (<COMPARISON>(<float>+) (<float>+))`
 The function compares first and second vectors and returns -1 if first proceeds the second, 0 if are equal, and +1 if first follows the second.
10. A pattern function could be a selection of predefined functions, or a custom function, such that:
`<PATTERN_FUNCTION> := JACCARD | EXPRESSION_LEVEL`
 A custom pattern function has following signature:
`<float> ← (<NAME> <REGION_LIST> [<REGION_LIST>] [(<parameter> <value>)]*)`
 If the second region list is missing, the function is computed with respect to the reference. Possible values of <NAME> could be JACCARD, DENSE, COOCCURRENCE, and etc., which could possibly be extended by user to best adapt the application.

11. An aggregate function is a selection from a set predefined functions, formally defined as follows:

```
<AGGREGATE_FUNCTION> :=  
    COVER |  
    SUMMIT |  
    MERGE |  
    COMPLEMENT |  
    DICHOTOMY |  
    ACCUMULATION_HISTOGRAM
```

With reference to this syntax, SPR and RPR are defined in following subsections.

7.2.1 Sample Pattern Rank

A query defines the pattern of interest, and Di4 returns a ranked set of samples. In general, let define SPR query as follows:

```
SAMPLE_LIST ← SPR (<REGION_LIST>  
    <WINDOW>  
    <SELECT_STATEMENT>+  
    <RANK_STATEMENT>)
```

Example 1

Function DENSE defines pattern on interest for promoter regions, and function JACCARD defines a pattern of interest for enhancer regions. A set of reference regions as candidate promoter and enhancer regions are given ((10 20) (35 65)). We are interested in set of samples that satisfy this property (see Figure 7.1).

```
SAMPLE_LIST ← SPR (  
    ((10 20) (35 65))  
    ( )  
    ((SELECT (Type Enhancer)) (PATTERN DENSE))  
    ((SELECT (Type Promoter)) (PATTERN JACCARD))  
    RANK_SAMPLES_INDEPENDENT)
```

Explanation: two reference regions (10 20) and (35 65) are given, which (10 20) could be an Enhancer region, and (35 65) be a Promoter region. Since we are interested in evaluation of promoters and enhancers together, the BODY of SPR is composed of two statements where:

- The first statement selects *Enhancers* from repository and evaluates their regions overlapping reference REGION_LIST using DENSE. Suppose the selection, select S_1 , S_2 , and S_3 from repository, and DENSE evaluates the regions of each sample as $\langle 12, 0 \rangle$, $\langle 37, 0.1 \rangle$, $\langle 8, 0 \rangle$ respectively.

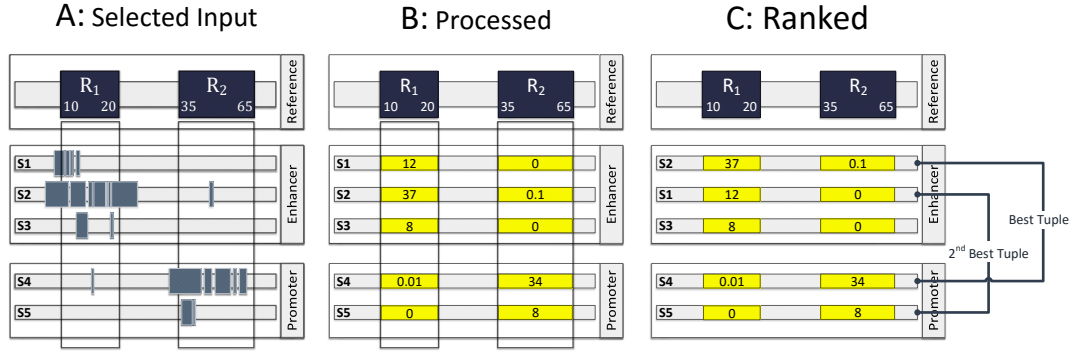


Figure 7.1: Sample Pattern Rank, Example 1.

- The second statement selects *Promoters* from repository and evaluates their regions intersecting reference REGION_LIST using JACCARD. Suppose the selection, selects S₄ and S₅, and JACCARD evaluates the regions of each sample as $\langle 0.01, 34 \rangle$, $\langle 0, 8 \rangle$ respectively.

The *Ranking method* is set to Rank_Sample_Independent, this means:

1. The samples from first statement (i.e., *Enhancers*) is ranked independent from the second statement (i.e., *Promoters*).
2. The output of SPR is a list of tuples (in this example a pair (because we defined two statements)).

Accordingly, the samples are evaluated as follows: S₁, S₂, and S₃ are ranked independently and we obtain following ranked list: (S₂ S₁ S₃); also S₄ and S₅ are evaluated independently and we obtain following ranked list: (S₄ S₅). The output of SPR is produced with the composition of the two sets (in general: n -dimensional cross product) and then ranking the set, an example would be:

SAMPLE_LIST = ((S2 S4) (S1 S5))

Example 2

Function COOCCURRENCE defines pattern of interest on a set of given reference regions ((10 20) (35 65)), and we are interested in finding samples from data type *Enhancer* and *Promoter* that best match the pattern (see Figure 7.2).

```
SAMPLE_LIST ← SPR (
  ((10 20) (35 65))
  ( )
  (SELECT ((Cell-line ENHANCER)(Cell-line PROMOTER))(PATTERN COOCCURRENCE))
  RANK_ALL_BY_PEARSON)
```

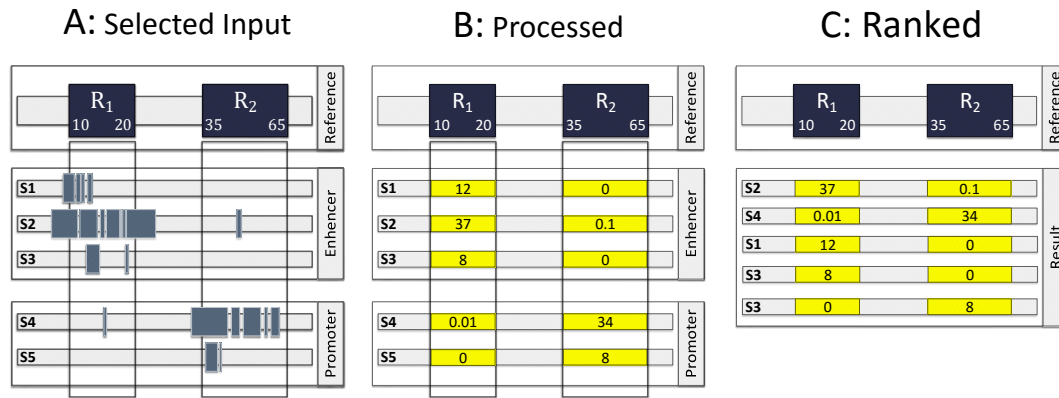


Figure 7.2: Sample Pattern Rank, Example 2.

Explanation: Two reference regions are given, and regions overlapping the reference from all sample belonging to data type *Enhancer* and *Promoter* are determined and each sample is evaluated using COOCCURRENCE function on the two reference regions. After this, we have multiple vectors representing each sample, then the choice RANK_ALL_BY_PEARSON specifies that the vectors to be ranked for closest match to the query, which outputs a ranked set of samples (NOT a set of tuples of samples as in previous example); for instance:

SAMPLE_LIST = (S₃ S₈ S₁ S₂)

Example 3

Referring to Example 1 (Section 7.2.1), incorporate the *Summit* of reference regions and add an extension of 10bp to up-stream and 53bp to down-stream. Also evaluate the regions differently for up-stream and down-stream to *Summit* by replacing DENSE with DENSE and SPARSE for up-stream and down-stream respectively, and JACCARD with DENSE and JACCARD for up-stream and down-stream respectively (see Figure 7.3).

```
SAMPLE_LIST ← SPR (
    ((10 20 15) (35 65 38))
    (10 53)
    (SELECT (Type Enhancer) (PATTERN DENSE SPARSE))
    (SELECT (Type Promoter) (PATTERN DENSE JACCARD))
    RANK_SAMPLES_INDEPENDENT)
```

Example 4

Given a pattern function DENSE that applies to enrichment at TFBS (reference regions: ((10 20) (35 65))) that are highly active at certain condition, we are interest to see, in general, in which conditions we have similar binding activity at these TFBS. Target conditions of interest are: (1) cell-line CL1, Antibody AB1, Lab LB1, (2) cell-line CL2, Antibody AB2, Lab LB2. At each

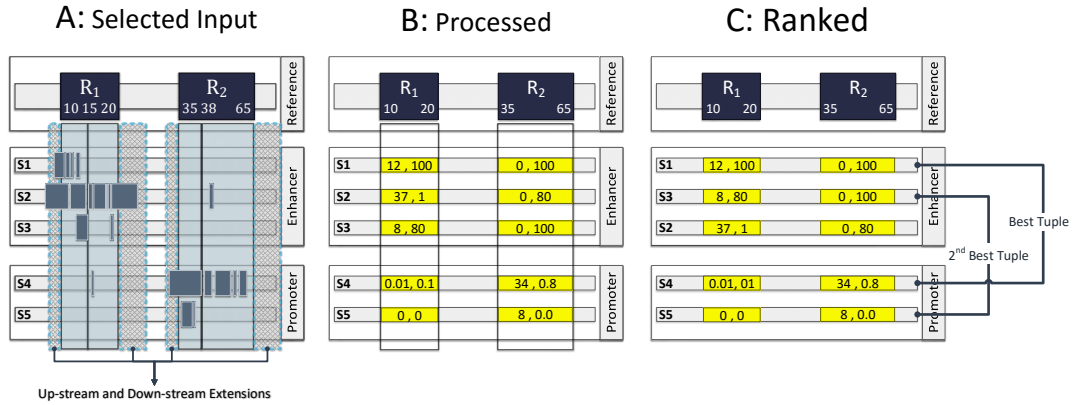


Figure 7.3: Sample Pattern Rank, Example 3.

condition, we might have multiple samples, that we consider them as replicates and merge their regions using MERGE function (see Figure 7.4).

```
SAMPLE_LIST ← SPR (
  ((10 20) (35 65))
  ( )
  (
    SELECT ((Cell-line CL1) (Lab LB1) (Antibody AB1))
    (AGGREGATE MERGE)
    (PATTERN DENSE)
  )
  (
    SELECT ((Cell-line CL2) (Lab LB2) (Antibody AB2))
    (AGGREGATE MERGE)
    (PATTERN DENSE)
  )
  RANK_ALL_BY_PEARSON)
```

7.2.2 Region Pattern Rank

A query defines the pattern of interest, and Di4 returns a ranked set of regions. In general, let define RPR query as follows:

```
SAMPLE_LIST ← RPR (<REGION_LIST>
  <WINDOW>
  <SELECT_STATEMENT>+
  <RANK_STATEMENT>)
```

Note that, rank statement for RPR does not include the Rank_Sample_Independent option as is available for SPR.

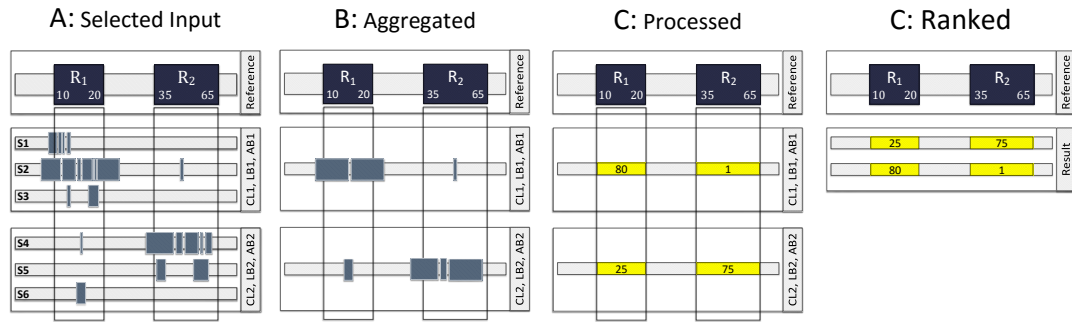


Figure 7.4: Sample Pattern Rank, Example 4.

Example 1

We are given a set of genes that are involved in a specific disease. We are asked to compare the given set of genes with genes involved in multiple similar conditions; and rank the reference genes such that a reference gene that is highly active in all the similar conditions is assigned with higher rank compared to another reference gene that has varying activity across similar conditions or low activity on all the similar conditions. For instance, we are given two genes which have coordinates (10, 20) and (35, 65). According to our experimental setup, samples with antibody AB1 and AB2 are considered similar to the reference, so we are interested in the enrichment of the reference genes in samples with AB1 and AB2. However, there could be multiple samples with AB1 in repository, which we would like to MERGE them to single sample. Additionally, there could be multiple samples with AB2 in repository, which we would like to aggregate them to single sample using COVER (2, ANY). Also, we define enrichment (i.e., pattern of interest) with pattern function ACTIVE. Finally, we would like to rank the enrichment of reference genes on the similar conditions using Pearson correlation (see Figure 7.5). To answer this question, we define RPR query as follows:

```
SAMPLE_LIST ← RPR (
  ((10 20) (35 65))
  ( )
  (
    SELECT (Antibody AB1)
    (AGGREGATE MERGE)
    (PATTERN ACTIVE)
  )
  (
    SELECT (Antibody AB2)
    (AGGREGATE (COVER 2 ANY))
    (PATTERN ACTIVE)
  )
  RANK_ALL_BY_PEARSON)
```

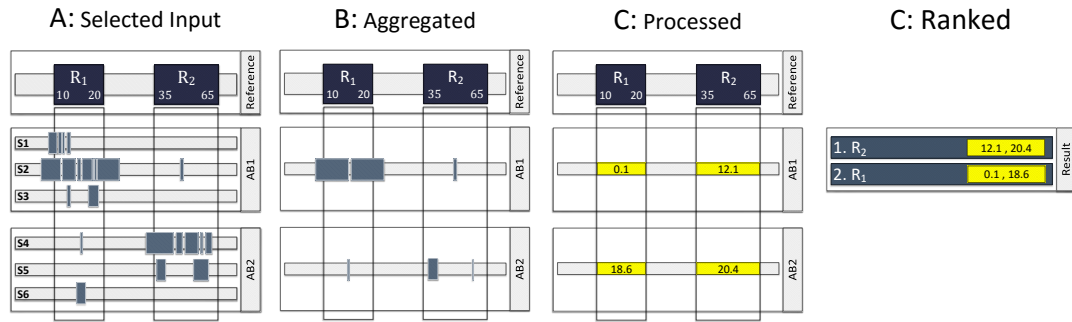


Figure 7.5: Region Pattern Rank, Example 1.

The query might calculate function ACTIVE for reference gene at (10, 20) with 0.1 on AB1 and 18.6 on AB2; and for reference gene at (35, 65) with 12.1 and 20.4 on AB1 and AB2 respectively. After final step that is ranking, the gene at (35, 65) with evaluations 12.1 and 20.4 shows relatively high activity on similar conditions than the gene at (10, 20) with evaluations 0.1 and 18.6.

Conclusion and Future Works

Comparative evaluation of evidences from a large collection of homo/heterogeneous genomic data is a paramount work that enables assessment of variety of aspects of the data. The big data size and the need for efficient retrieval and inferences, encourage adapting data indexing techniques. However, the diversity in comparative analysis requirements, heterogeneity in data types, and the large size of datasets, introduce analytical and computational challenges. Accordingly, a retrieval technique has to have *(i)* Tera-byte scalability, *(ii)* offer comprehensive and orthogonal region calculus, *(iii)* support a wide range of analytical operations that inherit from diverse research questions, and *(iv)* adapt to different application scenarios with varying computational infrastructure. Present dissertation discusses two indexing frameworks, 1D intervals inverted index (Di3) and 1D intervals incremental inverted index (Di4); these two offer aforementioned features, discussed generally as follows.

Multi-resolution data structure

Di3 and Di4 are single-dimension indexing frameworks that organize intervals based on coordinate attribute. Therefore, Di3 and Di4 excel coordinate-based operations. However, some inferences on data may operate upon an aggregated aspect of coordinates or meta-data of intervals. Such operations are supported by the multi-resolution design of Di3 and Di4, which enables an abstract view over a set of intervals through custom aggregates of coordinate attribute (indexed) or non-indexed attributes of intervals (e.g., “p-value”). Therefore, the support of a retrieval function that operates upon a non-indexed attribute is twofold. First, a second resolution is created by aggregating an attribute of a set of intervals using a user-defined function (discussed in Section 4.6). Second, the retrieval function uses the items of second-resolution to determine position on domain which satisfy function criteria (e.g., see COVER function discussed in Section 4.8.2). Based on the multi-resolution design, Di3 and Di4 operate optimally on non-indexed attributes.

Agnostic to persistence technology

Di3 and Di4 are agnostic to persistence technology. This design enables adaptation of the Di3 and Di4 data model and functionality to different persistence technologies spanning classical B+tree to local or distributed Big data key-value pair storage technologies such as LevelDB, Apache Cassandra, and Riak [200]; in general, any Dynamo [201] storage system.

This is achieved through the physical layer of Di3 and Di4. This layer defines CRUD (Create, Read, Update, and Delete) and enumerate operations, which bridge between the Di3 and Di4 functions, and persistence-technology-specific data access functionality. In other words, the physical layer abstracts the persistence technology to the functions of Di3 and Di4. This design makes Di3 and Di4 operations independent from persistence technology and minimizes the effort of adapting different storage technologies to manipulation of the physical layer only (which is simplified as implementing the physical layer interface for CRUD and enumerate operations based on persistence technology functionalities).

Agnostic to business logic layer and presentation scenarios

Di3 and Di4 are agnostic to business logic layer and presentation scenarios. This design facilitates adaptation to variety of disciplines and interval-based data models with application specific operations, defined at business logic layer. Additionally, it supports any user-interface spanning console-level accessibility and stand-alone graphical user interface (e.g., MSPC [162] and MuSERA [202] as discussed in Chapter 6), to web-based applications. This extensibility is due to the generic design of Di3 and Di4 model on interval-based data, which has two key advantages described as follows.

- i. It adapts the model to any interval-based data. In general, Di3 and Di4 organize intervals based on the primary attribute —coordinate, and are scoped lower and upper bounds which are the least common characteristics of interval-based data. This extends the model to different applications with varying coordinate characteristics. For instance, genomics locate an interval with chromosome and strand, in addition to lower and upper bounds. To address this, Di3 and Di4 are extensible to multiple-dimension data structure through additional resolutions defined at the data access layer (within business logic layer) per each extra coordination characteristic; which is one instance of Di3 or Di4 per each additional coordinate attribute. For instance in genomics, one Di3 or Di4 instance for *Chr1*, *+strand*, another Di3 or Di4 instance for *Chr1*, *-strand*, another Di3 or Di4 instance for *Chr2*, *+strand*, and so on so forth (see Figure 7.6). Note that, different resolutions are considered independent, and any domain specific inter-resolution process or retrieval is entrusted to the domain specific application (e.g., “if the sum of p-values of intervals overlapping *GeneX* on *Chr1* is above a certain threshold, then find intervals overlapping given *GeneY* on *Chr2*”).
- ii. Di3 and Di4 support heterogeneous data types; this is motivated by the wide-range of information in a discipline that can be abstracted by an interval-based representation. For instance, a gene, a promoter, a transcription factor binding site, and an un-annotated ChIP-seq peak, all represent different genomic concepts which are abstracted by an interval-based representation on the genome. Different data types share coordination characteristics, and differ in additional attributes. For instance, while an un-annotated ChIP-seq peak has “p-value” attribute, a gene has “GeneID” attribute instead. However, being generic on data type, Di3 and Di4 store a pointer to descriptive meta-data per each

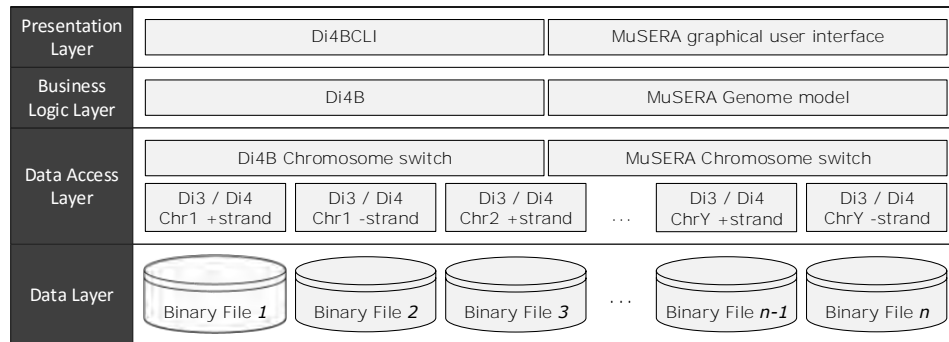


Figure 7.6: The multi-dimensional architecture of Di3 and Di4 for Chromosome and strand coordinate attributes.

indexed interval. In this case, an external storage may organize genes in a *table* which has “GeneID” attribute, and store un-annotated ChIP-seq peaks on another *table* which has “p-value” attribute. This design enables Di3 and Di4 to support heterogeneous data types; and descriptive information of interval-based data are fetched from external storage upon request by a function, using the pointer stored per each interval.

Extensive domain-specific operations

Extensive aspects of data is assessed using domain specific aggregates applied on coordinates and/or descriptive metadata of intervals. Di3 and Di4 determine a set of snapshots and the composing intervals as the result of a search query; then they apply a user-defined function on determined information to make a custom inference. For instance, in genomics, statistical significance of a “ChIP-seq peak” (interval) is verified by combining the “p-values” (descriptive metadata) of co-localized intervals of replicates (search result) using a data fusion function such as Fisher’s method (user-defined function). This assessment is extensively studied in MSPC [162] and MuSERA [202], and is discussed in Chapter 6. Such extensibility on domain-specific and application-specific functions is facilitated by the design of Di3 and Di4 through *design pattern* of functions (a composition of strategy and factory method patterns [122])¹. Such design provides extensive flexibility to Di3 and Di4 functions, and facilitates adaption to different disciplines with various inference requirements.

Future work

Di3 and Di4 index the primary attribute of an interval, i.e., coordinate, and provide logarithmic access to this attribute while offering a linear access (with the pruning of second-resolution) to additional attributes. In general, first, candidate intervals that satisfy the coordinate criteria

¹Domain and application specific aggregate algorithm, encapsulated in “strategy” object, is passed to Di3 and Di4 functions, and applied on determined snapshots and intervals. Finally, a “factory” determined output object is returned, which enables functions to return any user-defined output.

of a query are determined; then candidates are linearly filtered for the intervals complying the criteria of additional attributes. The efficiency of this method depends on the selectivity of coordinate criteria. Such that, a selective coordinate criteria, selects a relatively small number of indexed intervals which creates a significantly small number of candidates for the filter of additional attributes criteria. In contrast, a coordinate criteria with low or none selectivity, takes all or a big percentage of indexed intervals as candidates for the filter of additional attributes criteria; which has a cost proportional to linearly scanning Di3 and Di4 data structure. For instance, the query “find intervals overlapping a known genomic region that has p-value above X threshold” has coordinate criteria, which is “overlapping a known genomic region”; however, this criteria has limited selectivity because a majority of indexed intervals overlap known genomic regions. Therefore, finding intervals that satisfy the “p-value above X threshold” criteria is as expensive as a linear scan on Di3 and Di4. Additionally, regardless of the selectivity of coordinate criteria, creating candidate intervals is a performance penalty and may lead to a considerable *memory footprint*. Therefore, it is generally preferred to minimize the usage and size of candidate intervals.

To address these challenges, Di3 and Di4 can be extended as multi-attribute index; such that, the (commonly used) meta-data of intervals are also indexed in Di3 and Di4 as additional dimensions. Therefore, the intervals satisfying coordinate and additional attributes criteria of a query, are the intervals at the intersection of retrievals on multiple-dimension. This design has two advantages; first, the selectivity of coordinate attribute becomes equally important to the selectivity of indexed additional attributes. Second, minimizes the size of candidate intervals, and limits the generation of candidate intervals to queries with select criteria of non-indexed attributes, or an aggregation of indexed attributes.

Bibliography

- [1] Allan M Maxam and Walter Gilbert. A new method for sequencing dna. *Proceedings of the National Academy of Sciences*, 74(2):560–564, 1977.
- [2] Raja Jothi, Suresh Cuddapah, Artem Barski, Kairong Cui, and Keji Zhao. Genome-wide identification of in vivo protein–dna binding sites from chip-seq data. *Nucleic acids research*, 36(16):5221–5231, 2008.
- [3] Dominic Schmidt, Michael D Wilson, Christiana Spyrou, Gordon D Brown, James Hadfield, and Duncan T Odom. Chip-seq: Using high-throughput sequencing to discover protein–dna interactions. *Methods*, 48(3):240–248, 2009.
- [4] Peter J Park. Chip–seq: advantages and challenges of a maturing technology. *Nature Reviews Genetics*, 10(10):669–680, 2009.
- [5] Anton Valouev, David S Johnson, Andreas Sundquist, Catherine Medina, Elizabeth Anton, Serafim Batzoglou, Richard M Myers, and Arend Sidow. Genome-wide analysis of transcription factor binding sites based on chip-seq data. *Nature methods*, 5(9):829–834, 2008.
- [6] Shirley Pepke, Barbara Wold, and Ali Mortazavi. Computation for chip-seq and rna-seq studies. *Nature methods*, 6:S22–S32, 2009.
- [7] Bart Deplancke, Arnab Mukhopadhyay, Wanyuan Ao, Ahmed M Elewa, Christian A Grove, Natalia J Martinez, Reynaldo Sequerra, Lynn Doucette-Stamm, John S Reece-Hoyes, Ian A Hope, et al. A gene-centered c. elegans protein-dna interaction network. *Cell*, 125(6):1193–1205, 2006.
- [8] Reut Shalgi, Daniel Lieber, Moshe Oren, and Yitzhak Pilpel. Global and local architecture of the mammalian microRNA–transcription factor regulatory network. *PLoS computational biology*, 3(7):e131, 2007.
- [9] Job Dekker, Karsten Rippe, Martijn Dekker, and Nancy Kleckner. Capturing chromosome conformation. *science*, 295(5558):1306–1311, 2002.
- [10] Melissa J Fullwood and Yijun Ruan. Chip-based methods for the identification of long-range chromatin interactions. *Journal of cellular biochemistry*, 107(1):30–39, 2009.

Bibliography

- [11] Melissa J Fullwood, Mei Hui Liu, You Fu Pan, Jun Liu, Han Xu, Yusoff Bin Mohamed, Yuriy L Orlov, Stoyan Velkov, Andrea Ho, Poh Huay Mei, et al. An oestrogen-receptor- α -bound human chromatin interactome. *Nature*, 462(7269):58–64, 2009.
- [12] Dna sequencing costs.
- [13] Yuichi Kodama, Martin Shumway, and Rasko Leinonen. The sequence read archive: explosive growth of sequencing data. *Nucleic acids research*, 40(D1):D54–D56, 2012.
- [14] Tanya Barrett, Dennis B Troup, Stephen E Wilhite, Pierre Ledoux, Carlos Evangelista, Irene F Kim, Maxim Tomashevsky, Kimberly A Marshall, Katherine H Phillippy, Patti M Sherman, et al. Ncbi geo: archive for functional genomics data sets—10 years on. *Nucleic acids research*, 39(suppl 1):D1005–D1010, 2011.
- [15] ENCODE Project Consortium et al. The encode (encyclopedia of dna elements) project. *Science*, 306(5696):636–640, 2004.
- [16] Rhoda J Kinsella, Andreas Kähäri, Syed Haider, Jorge Zamora, Glenn Proctor, Giulietta Spudich, Jeff Almeida-King, Daniel Staines, Paul Derwent, Arnaud Kerhornou, et al. Ensembl biomarts: a hub for data retrieval across taxonomic space. *Database*, 2011:bar030, 2011.
- [17] José Caldas, Nils Gehlenborg, Eeva Kettunen, Ali Faisal, Mikko Rönty, Andrew G Nicholson, Sakari Knuutila, Alvis Brazma, and Samuel Kaski. Data-driven information retrieval in heterogeneous collections of transcriptomics data links sim2s to malignant pleural mesothelioma. *Bioinformatics*, 28(2):246–253, 2012.
- [18] Zhang Zhang, Jeffrey P Townsend, Jun Yu, Kei-Hoi Cheung, and Vladimir B Bajic. *Data integration in bioinformatics: current efforts and challenges*. INTECH Open Access Publisher, 2011.
- [19] Lin Dai, Xin Gao, Yan Guo, Jingfa Xiao, Zhang Zhang, et al. Bioinformatics clouds for big data manipulation. *Biology direct*, 7(1):43, 2012.
- [20] Xin Zeng, Rajendran Sanalkumar, Emery H Bresnick, Hongda Li, Qiang Chang, and Sündüz Keleş. jmosaics: joint analysis of multiple chip-seq datasets. *Genome biology*, 14(4):R38, 2013.
- [21] Stratos Idreos, Olga Papaemmanouil, and Surajit Chaudhuri. Overview of data exploration techniques. In *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*, pages 277–281. ACM, 2015.
- [22] Mark A Musen, Blackford Middleton, and Robert A Greenes. Clinical decision-support systems. In *Biomedical informatics*, pages 643–674. Springer, 2014.
- [23] Duncan Hull, Katy Wolstencroft, Robert Stevens, Carole Goble, Mathew R Pocock, Peter Li, and Tom Oinn. Taverna: a tool for building and running workflows of services. *Nucleic acids research*, 34(suppl 2):W729–W732, 2006.

-
- [24] Philippe Cudré-Mauroux, Hideaki Kimura, K-T Lim, Jennie Rogers, Roman Simakov, Emad Soroush, Pavel Velikhov, Daniel L Wang, Magdalena Balazinska, Jacek Becla, et al. A demonstration of scidb: a science-oriented dbms. *Proceedings of the VLDB Endowment*, 2(2):1534–1537, 2009.
- [25] Rebecca Taft, Manasi Vartak, Nadathur Rajagopalan Satish, Narayanan Sundaram, Samuel Madden, and Michael Stonebraker. Genbase: a complex analytics genomics benchmark. In *Proceedings of the 2014 ACM SIGMOD international conference on Management of data*, pages 177–188. ACM, 2014.
- [26] Jing Han, E Haihong, Guan Le, and Jian Du. Survey on nosql database. In *Pervasive computing and applications (ICPCA), 2011 6th international conference on*, pages 363–366. IEEE, 2011.
- [27] Bogdan George Tudorica and Cristian Bucur. A comparison between several nosql databases with comments and notes. In *Roedunet International Conference (RoEduNet), 2011 10th*, pages 1–5. IEEE, 2011.
- [28] John Vlissides, Richard Helm, Ralph Johnson, and Erich Gamma. Design patterns: Elements of reusable object-oriented software. *Reading: Addison-Wesley*, 49(120):11, 1995.
- [29] Maria D Chikina and Olga G Troyanskaya. An effective statistical evaluation of chipseq dataset similarity. *Bioinformatics*, 28(5):607–613, 2012.
- [30] Stephen G Landt, Georgi K Marinov, Anshul Kundaje, Pouya Kheradpour, Florencia Pauli, Serafim Batzoglou, Bradley E Bernstein, Peter Bickel, James B Brown, Philip Cayting, et al. Chip-seq guidelines and practices of the encode and modencode consortia. *Genome research*, 22(9):1813–1831, 2012.
- [31] David S Johnson, Ali Mortazavi, Richard M Myers, and Barbara Wold. Genome-wide mapping of in vivo protein-dna interactions. *Science*, 316(5830):1497–1502, 2007.
- [32] Tarjei S Mikkelsen, Manching Ku, David B Jaffe, Biju Issac, Erez Lieberman, Georgia Giannoukos, Pablo Alvarez, William Brockman, Tae-Kyung Kim, Richard P Koche, et al. Genome-wide maps of chromatin state in pluripotent and lineage-committed cells. *Nature*, 448(7153):553–560, 2007.
- [33] Xi Chen, Han Xu, Ping Yuan, Fang Fang, Mikael Huss, Vinsensius B Vega, Eleanor Wong, Yuriy L Orlov, Weiwei Zhang, Jianming Jiang, et al. Integration of external signaling pathways with the core transcriptional network in embryonic stem cells. *Cell*, 133(6):1106–1117, 2008.
- [34] Elizabeth D Wederell, Mikhail Bilenky, Rebecca Cullum, Nina Thiessen, Melis Dagpinar, Allen Delaney, Richard Varhol, YongJun Zhao, Thomas Zeng, Bridget Bernier, et al. Global analysis of in vivo foxa2-binding sites in mouse adult liver using massively parallel sequencing. *Nucleic acids research*, 36(14):4549–4564, 2008.

Bibliography

- [35] Aaron R Quinlan and Ira M Hall. Bedtools: a flexible suite of utilities for comparing genomic features. *Bioinformatics*, 26(6):841–842, 2010.
- [36] Shane Neph, M Scott Kuehn, Alex P Reynolds, Eric Haugen, Robert E Thurman, Audra K Johnson, Eric Rynes, Matthew T Maurano, Jeff Vierstra, Sean Thomas, et al. Bedops: high-performance genomic feature operations. *Bioinformatics*, 28(14):1919–1920, 2012.
- [37] W James Kent, Charles W Sugnet, Terrence S Furey, Krishna M Roskin, Tom H Pringle, Alan M Zahler, and David Haussler. The human genome browser at ucsc. *Genome research*, 12(6):996–1006, 2002.
- [38] Jeremy Goecks, Anton Nekrutenko, James Taylor, et al. Galaxy: a comprehensive approach for supporting accessible, reproducible, and transparent computational research in the life sciences. *Genome Biol*, 11(8):R86, 2010.
- [39] Ewan Birney, D Andrews, Mario Cáccamo, Yuan Chen, Laura Clarke, Guy Coates, Tony Cox, Fiona Cunningham, Val Curwen, Tim Cutts, et al. Ensembl 2006. *Nucleic acids research*, 34(suppl 1):D556–D561, 2006.
- [40] Belinda Giardine, Laura Elnitski, Cathy Riemer, Izabela Makalowska, Scott Schwartz, Webb Miller, and Ross C Hardison. Gala, a database for genomic sequence alignments and annotations. *Genome research*, 13(4):732–741, 2003.
- [41] Alexander V Alekseyenko and Christopher J Lee. Nested containment list (nclist): a new algorithm for accelerating interval query of genome alignment and interval databases. *Bioinformatics*, 23(11):1386–1393, 2007.
- [42] Lincoln D Stein, Christopher Mungall, ShengQiang Shu, Michael Caudy, Marco Mangone, Allen Day, Elizabeth Nickerson, Jason E Stajich, Todd W Harris, Adrian Arva, et al. The generic genome browser: a building block for a model organism system database. *Genome research*, 12(10):1599–1610, 2002.
- [43] Ryan M Layer, Kevin Skadron, Gabriel Robins, Ira M Hall, and Aaron R Quinlan. Binary interval search: a scalable algorithm for counting interval intersections. *Bioinformatics*, 29(1):1–7, 2013.
- [44] Gabriel Renaud, Pedro Neves, Edson Luiz Folador, Carlos Gil Ferreira, and Fabio Passetti. Segtor: rapid annotation of genomic coordinates and single nucleotide variations using segment trees. *PloS one*, 6(11):e26715, 2011.
- [45] Heng Li, Bob Handsaker, Alec Wysoker, Tim Fennell, Jue Ruan, Nils Homer, Gabor Marth, Goncalo Abecasis, Richard Durbin, et al. The sequence alignment/map format and samtools. *Bioinformatics*, 25(16):2078–2079, 2009.
- [46] Antonin Guttman. *R-trees: a dynamic index structure for spatial searching*, volume 14. ACM, 1984.

-
- [47] Joel E Richardson. Fjoin: simple and efficient computation of feature overlaps. *Journal of Computational Biology*, 13(8):1457–1464, 2006.
 - [48] Michael Ian Shamos and Dan Hoey. Geometric intersection problems. In *Foundations of Computer Science, 1976., 17th Annual Symposium on*, pages 208–215. IEEE, 1976.
 - [49] Jon L Bentley and Thomas A Ottmann. Algorithms for reporting and counting geometric intersections. *Computers, IEEE Transactions on*, 100(9):643–647, 1979.
 - [50] Bernard Chazelle and Herbert Edelsbrunner. An optimal algorithm for intersecting line segments in the plane. *Journal of the ACM (JACM)*, 39(1):1–54, 1992.
 - [51] Mark McKenney and Tynan McGuire. A parallel plane sweep algorithm for multi-core systems. In *Proceedings of the 17th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, pages 392–395. ACM, 2009.
 - [52] Michael T Goodrich, Jyh-Jong Tsay, Darren Erik Vengroff, and Jeffrey Scott Vitter. External-memory computational geometry. In *Foundations of Computer Science, 1993. Proceedings., 34th Annual Symposium on*, pages 714–723. IEEE, 1993.
 - [53] Hans-Peter Kriegel, Thomas Brinkhoff, and Ralf Schneider. The combination of spatial access methods and computational geometry in geographic database systems. In *Advances in Spatial Databases*, pages 5–21. Springer, 1991.
 - [54] Heng Li. Tabix: fast retrieval of sequence features from generic tab-delimited files. *Bioinformatics*, 27(5):718–719, 2011.
 - [55] Matthias Zytnecki, YuFei Luo, and Hadi Quesneville. Efficient comparison of sets of intervals with nc-lists. *Bioinformatics*, 29(7):933–939, 2013.
 - [56] Ryan M Layer and Aaron R Quinlan. A parallel algorithm for-way interval set intersection.
 - [57] Thomas H Cormen, Charles E Leiserson, Ronald L Rivest, and Clifford Stein. *Section 14.3: Interval trees*, chapter 14, pages 348–354. MIT press Cambridge, third edition edition.
 - [58] Jon Louis Bentley. Solutions to klee’s rectangle problems. Technical report, Technical report, Carnegie-Mellon Univ., Pittsburgh, PA, 1977.
 - [59] Jon Louis Bentley. Decomposable searching problems. *Information Processing Letters*, 8(5):244–251, 1979.
 - [60] Peter M Fenwick. A new data structure for cumulative frequency tables. *Software: Practice and Experience*, 24(3):327–336, 1994.
 - [61] Utz Westermann and Ramesh Jain. Toward a common event model for multimedia applications. *IEEE MultiMedia*, (1):19–29, 2007.

Bibliography

- [62] Michael Worboys and Kathleen Hornsby. From objects to events: Gem, the geospatial event model. In *Geographic Information Science*, pages 327–343. Springer, 2004.
- [63] Tuukka Ruotsalo and Eero Hyvönen. *An event-based approach for semantic metadata interoperability*. Springer, 2007.
- [64] MJ Soomer and Geert Jan Franx. Scheduling aircraft landings using airlines' preferences. *European Journal of Operational Research*, 190(1):277–291, 2008.
- [65] Sheng-Peng Yu, Xian-Bin Cao, and Jun Zhang. A real-time schedule method for aircraft landing scheduling problem based on cellular automation. *Applied Soft Computing*, 11(4):3485–3493, 2011.
- [66] Andrew Lim, Brian Rodrigues, and Yi Zhu. Airport gate scheduling with time windows. *Artificial Intelligence Review*, 24(1):5–31, 2005.
- [67] Ulrich Dorndorf, Andreas Drexl, Yury Nikulin, and Erwin Pesch. Flight gate scheduling: State-of-the-art and recent developments. *Omega*, 35(3):326–334, 2007.
- [68] Cynthia Barnhart, Amy M Cohn, Ellis L Johnson, Diego Klabjan, George L Nemhauser, and Pamela H Vance. Airline crew scheduling. In *Handbook of transportation science*, pages 517–560. Springer, 2003.
- [69] Joyce W Yen and John R Birge. A stochastic programming approach to the airline crew scheduling problem. *Transportation Science*, 40(1):3–14, 2006.
- [70] Ahmed Abdelghany, Khaled Abdelghany, and Ram Narasimhan. Scheduling baggage-handling facilities in congested airports. *Journal of Air Transport Management*, 12(2):76–81, 2006.
- [71] Jose L Tongzon. Determinants of port performance and efficiency. *Transportation Research Part A: Policy and Practice*, 29(3):245–252, 1995.
- [72] Chris N Potts and Vitaly A Strusevich. Fifty years of scheduling: a survey of milestones. *Journal of the Operational Research Society*, pages S41–S68, 2009.
- [73] Nicholas G Hall, Chris N Potts, and Chelliah Sriskandarajah. Parallel machine scheduling with a common server. *Discrete Applied Mathematics*, 102(3):223–243, 2000.
- [74] Daniel Costa. An evolutionary tabu search algorithm and the nhl scheduling problem. *Infor-Information Systems and Operational Research*, 33(3):161–178, 1995.
- [75] Graham Kendall, Sigrid Knust, Celso C Ribeiro, and Sebastián Urrutia. Scheduling in sports: An annotated bibliography. *Computers & Operations Research*, 37(1):1–19, 2010.
- [76] Leslie A Hall. Approximability of flow shop scheduling. *Mathematical Programming*, 82(1-2):175–190, 1998.

-
- [77] TC Edwin Cheng and Zhaohui Liu. Approximability of two-machine no-wait flowshop scheduling with availability constraints. *Operations Research Letters*, 31(4):319–322, 2003.
- [78] Reuven Bar-Yehuda, Magnús M Halldórsson, Joseph Naor, Hadas Shachnai, and Irina Shapira. Scheduling split intervals. *SIAM Journal on Computing*, 36(1):1–15, 2006.
- [79] Richard Snodgrass and Ilsoo Ahn. A taxonomy of time databases. In *ACM Sigmod Record*, volume 14, pages 236–246. ACM, 1985.
- [80] Siegfried M Rump. *Algebraic computation, numerical computation and verified inclusions*. Springer, 1988.
- [81] Marc Daumas and Guillaume Melquiond. Generating formally certified bounds on values and round-off errors. In *Real Numbers and Computers*, pages 55–70, 2004.
- [82] Thomas C Hales. A proof of the kepler conjecture. *Annals of mathematics*, pages 1065–1185, 2005.
- [83] Michael Hutchings, Frank Morgan, Manuel Ritoré, and Antonio Ros. Proof of the double bubble conjecture. *Annals of Mathematics*, pages 459–489, 2002.
- [84] Z Galias and P Zgliczyński. Computer assisted proof of chaos in the lorenz equations. *Physica D: Nonlinear Phenomena*, 115(3):165–188, 1998.
- [85] LA Rastrigin. Systems of extremal control, 1974.
- [86] Rainer Storn and Kenneth Price. Differential evolution—a simple and efficient heuristic for global optimization over continuous spaces. *Journal of global optimization*, 11(4):341–359, 1997.
- [87] Norman W Paton and Oscar Díaz. Active database systems. *ACM Computing Surveys (CSUR)*, 31(1):63–103, 1999.
- [88] Jennifer Widom and Stefano Ceri. *Active database systems: Triggers and rules for advanced database processing*. Morgan Kaufmann, 1996.
- [89] Sharma Chakravarthy, Vidhya Krishnaprasad, Eman Anwar, and Seung-Kyum Kim. Composite events for active databases: Semantics, contexts and detection. In *VLDB*, volume 94, pages 606–617, 1994.
- [90] Donna J Peuquet and Niu Duan. An event-based spatiotemporal data model (estdm) for temporal analysis of geographical data. *International journal of geographical information systems*, 9(1):7–24, 1995.
- [91] Samuel R Madden, Michael J Franklin, Joseph M Hellerstein, and Wei Hong. Tinydb: an acquisitional query processing system for sensor networks. *ACM Transactions on database systems (TODS)*, 30(1):122–173, 2005.

Bibliography

- [92] Zhao Cao, Yanlei Diao, and Prashant Shenoy. Architectural considerations for distributed rfid tracking and monitoring. In *Proceedings of the ACM Workshop on Networking Meets Databases (NetDB)*, 2009.
- [93] Venkatesh Raghavan, Elke Rundensteiner, John Woycheese, and Abhishek Mukherji. Firestream: Sensor stream processing for monitoring fire spread. In *2007 IEEE 23rd International Conference on Data Engineering*, pages 1507–1508. IEEE, 2007.
- [94] Lexing Xie, Hari Sundaram, and Murray Campbell. Event mining in multimedia streams. *Proceedings of the IEEE*, 96(4):623–647, 2008.
- [95] James F Allen. Maintaining knowledge about temporal intervals. *Communications of the ACM*, 26(11):832–843, 1983.
- [96] Richard E Fikes and Nils J Nilsson. Strips: A new approach to the application of theorem proving to problem solving. *Artificial intelligence*, 2(3):189–208, 1972.
- [97] Earl D Sacerdoti. A structure for plans and behavior. Technical report, DTIC Document, 1975.
- [98] Bertram C Bruce. A model for temporal references and its application in a question answering program. *Artificial intelligence*, 3:1–25, 1972.
- [99] Gary G Hendrix. Modeling simultaneous actions and continuous processes. *Artificial Intelligence*, 4(3):145–180, 1974.
- [100] Kenneth Kahn and G Anthony Gorry. Mechanizing temporal knowledge. *Artificial intelligence*, 9(1):87–108, 1977.
- [101] Drew McDermott. A temporal logic for reasoning about processes and plans*. *Cognitive science*, 6(2):101–155, 1982.
- [102] Elisa Bertino, Elena Ferrari, and Giovanna Guerrini. An approach to model and query event-based temporal data. In *Temporal Representation and Reasoning, 1998. Proceedings. Fifth International Workshop on*, pages 122–131. IEEE, 1998.
- [103] Maximilian Dylla, Iris Miliaraki, and Martin Theobald. A temporal-probabilistic database model for information extraction. *Proceedings of the VLDB Endowment*, 6(14):1810–1821, 2013.
- [104] Rita de Caluwe, Guy De Tré, and Gloria Bordogna. *Spatio-temporal databases: flexible querying and reasoning*. Springer Science & Business Media, 2013.
- [105] Yannis Manolopoulos, Alexandros Nanopoulos, Apostolos N Papadopoulos, and Yannis Theodoridis. *R-trees: Theory and Applications*. Springer Science & Business Media, 2010.

-
- [106] Willem Robert Van Hage, Véronique Malaisé, Roxane Segers, Laura Hollink, and Guus Schreiber. Design and use of the simple event model (sem). *Web Semantics: Science, Services and Agents on the World Wide Web*, 9(2):128–136, 2011.
 - [107] Ruslan Mavlyutov and Philippe Cudre-Mauroux. Cintia: A distributed, low-latency index for big interval data. In *Big Data (Big Data), 2015 IEEE International Conference on*, pages 619–628. IEEE, 2015.
 - [108] Benjamin Schlegel, Thomas Willhalm, and Wolfgang Lehner. Fast sorted-set intersection using simd instructions. In *ADMS@ VLDB*, pages 1–8, 2011.
 - [109] Bolin Ding and Arnd Christian König. Fast set intersection in memory. *Proceedings of the VLDB Endowment*, 4(4):255–266, 2011.
 - [110] Marcus Fontoura, Maxim Gurevich, Vanja Josifovski, and Sergei Vassilvitskii. Efficiently encoding term co-occurrences in inverted indexes. In *Proceedings of the 20th ACM international conference on Information and knowledge management*, pages 307–316. ACM, 2011.
 - [111] Hiroshi Inoue, Moriyoshi Ohara, and Kenjiro Taura. Faster set intersection with simd instructions by reducing branch mispredictions. *Proceedings of the VLDB Endowment*, 8(3):293–304, 2014.
 - [112] Joshua Brody, Amit Chakrabarti, Ranganath Kondapally, David P Woodruff, and Grigory Yaroslavlsev. Beyond set disjointness: the communication complexity of finding the intersection. In *Proceedings of the 2014 ACM symposium on Principles of distributed computing*, pages 106–113. ACM, 2014.
 - [113] Weixiong Rao, Lei Chen, Pan Hui, and Sasu Tarkoma. Bitlist: New full-text index for low space cost and efficient keyword search. *Proceedings of the VLDB Endowment*, 6(13):1522–1533, 2013.
 - [114] Yufei Tao and Dimitris Papadias. The mv3r-tree: A spatio-temporal access method for timestamp and interval queries. In *Proceedings of Very Large Data Bases Conference (VLDB), 11-14 September, Rome, 2001*.
 - [115] Theodoros Tzouramanis, Yannis Manolopoulos, and Nikos Lorentzos. Overlapping b+-trees: an implementation of a transaction time access method. *Data & Knowledge Engineering*, 29(3):381–404, 1999.
 - [116] Kristian Ovaska, Lauri Lyly, Biswajyoti Sahu, Olli A Janne, and Sampsa Hautaniemi. Genomic region operation kit for flexible processing of deep sequencing data. *IEEE/ACM Transactions on Computational Biology and Bioinformatics (TCBB)*, 10(1):200–206, 2013.
 - [117] Curtis Dyreson, Fabio Grandi, Wolfgang Käfer, Nick Kline, Nikos Lorentzos, Yannis Mitsopoulos, Angelo Montanari, Daniel Nonen, Elisa Peressi, Barbara Pernici, et al. A consensus glossary of temporal database concepts. *ACM Sigmod Record*, 23(1):52–64, 1994.

Bibliography

- [118] Goetz Graefe. Sorting and indexing with partitioned b-trees. In *CIDR*, volume 3, pages 5–8, 2003.
- [119] Thanaa M Ghanem, Rahul Shah, Mohamed F Mokbel, Walid G Aref, and Jeffrey S Vitter. Bulk operations for space-partitioning trees. In *Data Engineering, 2004. Proceedings. 20th International Conference on*, pages 29–40. IEEE, 2004.
- [120] Martin Erwig and Markus Schneider. Spatio-temporal predicates. *Knowledge and Data Engineering, IEEE Transactions On*, 14(4):881–901, 2002.
- [121] Panagiotis Papapetrou, George Kollios, Stan Sclaroff, and Dimitrios Gunopulos. Mining frequent arrangements of temporal intervals. *Knowledge and Information Systems*, 21(2):133–171, 2009.
- [122] Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides. *Design patterns: elements of reusable object-oriented software*. Pearson Education, 1994.
- [123] Roded Sharan, Ivan Ovcharenko, Asa Ben-Hur, and Richard M Karp. Creme: a framework for identifying cis-regulatory modules in human-mouse conserved segments. *Bioinformatics*, 19(suppl 1):i283–i291, 2003.
- [124] Priya Sudarsanam, Yitzhak Pilpel, and George M Church. Genome-wide co-occurrence of promoter elements reveals a cis-regulatory cassette of rna transcription motifs in *saccharomyces cerevisiae*. *Genome research*, 12(11):1723–1731, 2002.
- [125] Yasuhiko Morimoto. Mining frequent neighboring class sets in spatial databases. In *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 353–358. ACM, 2001.
- [126] Rakesh Agrawal, Tomasz Imieliński, and Arun Swami. Mining association rules between sets of items in large databases. In *ACM SIGMOD Record*, volume 22, pages 207–216. ACM, 1993.
- [127] Karthik Ganesan Pillai, Rafal Angryk, Juan M Banda, Michael Schuh, Tim Wylie, et al. Spatio-temporal co-occurrence pattern mining in data sets with evolving regions. In *Data Mining Workshops (ICDMW), 2012 IEEE 12th International Conference on*, pages 805–812. IEEE, 2012.
- [128] Xiangye Xiao, Xing Xie, Qiong Luo, and Wei-Ying Ma. Density based co-location pattern discovery. In *Proceedings of the 16th ACM SIGSPATIAL international conference on Advances in geographic information systems*, page 29. ACM, 2008.
- [129] Vladimir Estivill-Castrol and Alan T Murray. Discovering associations in spatial data—an efficient medoid based approach. In *Research and Development in Knowledge Discovery and Data Mining*, pages 110–121. Springer, 1998.

-
- [130] Yuhan Cai and Raymond Ng. Indexing spatio-temporal trajectories with chebyshev polynomials. In *Proceedings of the 2004 ACM SIGMOD international conference on Management of data*, pages 599–610. ACM, 2004.
 - [131] Berkay Aydin, Dustin Kempton, Vijay Akkineni, Shaktidhar Reddy Gopavaram, Karthik Ganesan Pillai, and Rafal Angryk. Spatiotemporal indexing techniques for efficiently mining spatiotemporal co-occurrence patterns. In *Big Data (Big Data), 2014 IEEE International Conference on*, pages 1–10. IEEE, 2014.
 - [132] V Prasad Chakka, Adam C Everspaugh, and Jignesh M Patel. Indexing large trajectory data sets with seti. *Ann Arbor*, 1001:48109–2122, 2003.
 - [133] Hanan Samet. *Foundations of multidimensional and metric data structures*. Morgan Kaufmann, 2006.
 - [134] Mohamed F Mokbel, Thanaa M. Ghanem, and Walid G. Aref. Spatio-temporal access methods. *IEEE Data Eng. Bull.*, 26(2):40–49, 2003.
 - [135] Long-Van Nguyen-Dinh, Walid G Aref, and Mohamed Mokbel. Spatio-temporal access methods: Part 2 (2003-2010). 2010.
 - [136] Rakesh Agrawal, Ramakrishnan Srikant, et al. Fast algorithms for mining association rules. In *Proc. 20th int. conf. very large data bases, VLDB*, volume 1215, pages 487–499, 1994.
 - [137] Yan Huang, Shashi Shekhar, and Hui Xiong. Discovering colocation patterns from spatial data sets: a general approach. *Knowledge and Data Engineering, IEEE Transactions on*, 16(12):1472–1485, 2004.
 - [138] Yan Huang, Jian Pei, and Hui Xiong. Mining co-location patterns with rare events from spatial data sets. *Geoinformatica*, 10(3):239–260, 2006.
 - [139] Vladimir Estivill-Castro and Ickjai Lee. Data mining techniques for autonomous exploration of large volumes of geo-referenced crime data. In *Proc. of the 6th International Conference on Geocomputation*, pages 24–26. Citeseer, 2001.
 - [140] Junmei Wang, Wynne Hsu, and Mong Li Lee. A framework for mining topological patterns in spatio-temporal databases. In *Proceedings of the 14th ACM international conference on Information and knowledge management*, pages 429–436. ACM, 2005.
 - [141] Jin Soung Yoo, Shashi Shekhar, John Smith, and Julius P Kumquat. A partial join approach for mining co-location patterns. In *Proceedings of the 12th annual ACM international workshop on Geographic information systems*, pages 241–249. ACM, 2004.
 - [142] Shashi Shekhar, Pusheng Zhang, Yan Huang, and Ranga Raju Vatsavai. Trends in spatial data mining. *Data mining: Next generation challenges and future directions*, pages 357–380, 2003.

Bibliography

- [143] Shashi Shekhar and Yan Huang. Discovering spatial co-location patterns: A summary of results. In *Advances in Spatial and Temporal Databases*, pages 236–256. Springer, 2001.
- [144] Jin Soung Yoo and Shashi Shekhar. A joinless approach for mining spatial colocation patterns. *Knowledge and Data Engineering, IEEE Transactions on*, 18(10):1323–1337, 2006.
- [145] Jiawei Han, Jian Pei, and Yiwen Yin. Mining frequent patterns without candidate generation. In *ACM SIGMOD Record*, volume 29, pages 1–12. ACM, 2000.
- [146] Jiawei Han, Jian Pei, Yiwen Yin, and Runying Mao. Mining frequent patterns without candidate generation: A frequent-pattern tree approach. *Data mining and knowledge discovery*, 8(1):53–87, 2004.
- [147] Mete Celik, Shashi Shekhar, James P Rogers, James Shine, et al. Mixed-drove spatiotemporal co-occurrence pattern mining. *Knowledge and Data Engineering, IEEE Transactions on*, 20(10):1322–1335, 2008.
- [148] Zhanquan Wang, Xuanhuang Peng, Chunhua Gu, and Bingqiang Huang. Mining at most top-k% mixed-drove spatiotemporal co-occurrence patterns. In *Control Conference (ASCC), 2013 9th Asian*, pages 1–5. IEEE, 2013.
- [149] Yan Huang, Liqin Zhang, and Pusheng Zhang. A framework for mining sequential patterns from spatio-temporal event data sets. *Knowledge and Data Engineering, IEEE Transactions on*, 20(4):433–448, 2008.
- [150] Joachim Gudmundsson and Marc van Kreveld. Computing longest duration flocks in trajectory data. In *Proceedings of the 14th annual ACM international symposium on Advances in geographic information systems*, pages 35–42. ACM, 2006.
- [151] Dina Q Goldin, Todd D Millstein, and Ayferi Kutlu. Bounded similarity querying for time-series data. *Information and Computation*, 194(2):203–241, 2004.
- [152] Mete Celik, Shashi Shekhar, James P Rogers, James Shine, Jin Soung Yoo, et al. Mixed-drove spatio-temporal co-occurrence pattern mining: A summary of results. In *Data Mining, 2006. ICDM'06. Sixth International Conference on*, pages 119–128. IEEE, 2006.
- [153] Erwin L van Dijk, Hélène Auger, Yan Jaszczyszyn, and Claude Thermes. Ten years of next-generation sequencing technology. *Trends in genetics*, 30(9):418–426, 2014.
- [154] Yiwen Chen, Nicolas Negre, Qunhua Li, Joanna O Mieczkowska, Matthew Slattery, Tao Liu, Yong Zhang, Tae-Kyung Kim, Housheng Hansen He, Jennifer Zieba, et al. Systematic evaluation of factors influencing chip-seq fidelity. *Nature methods*, 9(6):609–614, 2012.
- [155] Mark B Gerstein, Zhi John Lu, Eric L Van Nostrand, Chao Cheng, Bradley I Arshinoff, Tao Liu, Kevin Y Yip, Rebecca Robilotto, Andreas Rechtsteiner, Kohta Ikegami, et al. Integrative analysis of the caenorhabditis elegans genome by the modencode project. *Science*, 330(6012):1775–1787, 2010.

-
- [156] Maya Kasowski, Fabian Grubert, Christopher Heffelfinger, Manoj Hariharan, Akwasi Asabere, Sebastian M Waszak, Lukas Habegger, Joel Rozowsky, Minyi Shi, Alexander E Urban, et al. Variation in transcription factor binding among humans. *science*, 328(5975):232–235, 2010.
- [157] Matthew T Maurano, Eric Haugen, Richard Sandstrom, Jeff Vierstra, Anthony Shafer, Rajinder Kaul, and John A Stamatoyannopoulos. Large-scale identification of sequence variants influencing human transcription factor occupancy in vivo. *Nature genetics*, 2015.
- [158] Wei Zheng, Hongyu Zhao, Eugenio Mancera, Lars M Steinmetz, and Michael Snyder. Genetic analysis of variation in transcription factor binding in yeast. *Nature*, 464(7292):1187–1191, 2010.
- [159] Elizabeth G Wilbanks and Marc T Facciotti. Evaluation of algorithm performance in chip-seq peak detection. *PloS one*, 5(7):e11471, 2010.
- [160] Anaïs F Bardet, Qiye He, Julia Zeitlinger, and Alexander Stark. A computational pipeline for comparative chip-seq analyses. *Nature protocols*, 7(1):45–61, 2012.
- [161] Kirk E Lohmueller, Celeste L Pearce, Malcolm Pike, Eric S Lander, and Joel N Hirschhorn. Meta-analysis of genetic association studies supports a contribution of common variants to susceptibility to common disease. *Nature genetics*, 33(2):177–182, 2003.
- [162] Vahid Jalili, Matteo Matteucci, Marco Masseroli, and Marco J. Morelli. Using combined evidence from replicates to evaluate chip-seq peaks. *Bioinformatics*, 31(17):2761–2769, 2015.
- [163] Qunhua Li, James B Brown, Haiyan Huang, and Peter J Bickel. Measuring reproducibility of high-throughput experiments. *The annals of applied statistics*, pages 1752–1779, 2011.
- [164] Pei Fen Kuan, Dongjun Chung, Guangjin Pan, James A Thomson, Ron Stewart, and Sündüz Keleş. A statistical framework for the analysis of chip-seq data. *Journal of the American Statistical Association*, 106(495):891–903, 2011.
- [165] Ronald Aylmer Fisher. Statistical methods for research workers. 1934.
- [166] Steven Goodman. A dirty dozen: twelve p-value misconceptions. In *Seminars in hematology*, volume 45, pages 135–140. Elsevier, 2008.
- [167] Mark J Schervish. P values: what they are and what they are not. *The American Statistician*, 50(3):203–206, 1996.
- [168] Jonathan AC Sterne and George Davey Smith. Sifting the evidence—what’s wrong with significance tests? *Physical Therapy*, 81(8):1464–1469, 2001.

Bibliography

- [169] Ronald A Fisher. The arrangement of field experiments. In *Breakthroughs in Statistics*, pages 82–91. Springer, 1992.
- [170] Joachim Hartung. A note on combining dependent tests of significance. Technical report, Technical Report, SFB 475: Komplexitätsreduktion in Multivariaten Datenstrukturen, Universität Dortmund, 1998.
- [171] Morton B Brown. 400: A method for combining non-independent, one-sided tests of significance. *Biometrics*, pages 987–992, 1975.
- [172] T Liptak. On the combination of independent tests. *Magyar Tud Akad Mat Kutato Int Kozl*, 3:171–197, 1958.
- [173] Govind S Mudholkar and E Olusegun George. The logit statistic for combining probabilities-an overview. Technical report, DTIC Document, 1977.
- [174] Bryan Wilkinson. A statistical consideration in psychological research. *Psychological Bulletin*, 48(2):156, 1951.
- [175] Dmitri V Zaykin, Lev A Zhivotovsky, Peter H Westfall, and Bruce S Weir. Truncated product method for combining p-values. *Genetic epidemiology*, 22(2):170–185, 2002.
- [176] Ramon C Littell and J Leroy Folks. Asymptotic optimality of fisher’s method of combining independent tests. *Journal of the American Statistical Association*, 66(336):802–806, 1971.
- [177] Ramon C Littell and J Leroy Folks. Asymptotic optimality of fisher’s method of combining independent tests ii. *Journal of the American Statistical Association*, 68(341):193–194, 1973.
- [178] RR Bahadur. Stochastic comparison of tests. *Annals of Mathematical Statistics*, 31(2):276–295, 1960.
- [179] Sungho Won, Nathan Morris, Qing Lu, and Robert C Elston. Choosing an optimal method to combine p-values. *Statistics in medicine*, 28(11):1537, 2009.
- [180] Dmitri V Zaykin, Lev A Zhivotovsky, Wendy Czika, Susan Shao, and Russell D Wolfinger. Combining p-values in large-scale genomics experiments. *Pharmaceutical statistics*, 6(3):217–226, 2007.
- [181] Frank K. Hwang and Shen Lin. A simple algorithm for merging two disjoint linearly ordered sets. *SIAM Journal on Computing*, 1(1):31–39, 1972.
- [182] Rudolf Bayer. Symmetric binary b-trees: Data structure and maintenance algorithms. *Acta informatica*, 1(4):290–306, 1972.
- [183] Edward M McCreight. Priority search trees. *SIAM Journal on Computing*, 14(2):257–276, 1985.

-
- [184] Herbert Edelsbrunner. *Dynamic data structures for orthogonal intersection queries*. Technische Universität Graz/Forschungszentrum Graz. Institut für Informationsverarbeitung, 1980.
- [185] Franco P Preparata and Michael Shamos. *Computational geometry: an introduction*. Springer Science & Business Media, 2012.
- [186] Michael Bulger and Mark Groudine. Enhancers: the abundance and function of regulatory sequences beyond promoters. *Developmental biology*, 339(2):250–257, 2010.
- [187] Laura A Lettice, Simon JH Heaney, Lorna A Purdie, Li Li, Philippe de Beer, Ben A Oostra, Debbie Goode, Greg Elgar, Robert E Hill, and Esther de Graaff. A long-range shh enhancer regulates expression in the developing limb and fin and is associated with preaxial polydactyly. *Human molecular genetics*, 12(14):1725–1735, 2003.
- [188] William J Glassford and Mark Rebeiz. Assessing constraints on the path of regulatory sequence evolution. *Philosophical Transactions of the Royal Society B: Biological Sciences*, 368(1632):20130026, 2013.
- [189] Kyle L MacQuarrie, Abraham P Fong, Randall H Morse, and Stephen J Tapscott. Genome-wide transcription factor binding: beyond direct target regulation. *Trends in Genetics*, 27(4):141–148, 2011.
- [190] Karl Pearson. Note on regression and inheritance in the case of two parents. *Proceedings of the Royal Society of London*, pages 240–242, 1895.
- [191] Xiaobei Zhao, Eivind Valen, Brian J Parker, and Albin Sandelin. Systematic clustering of transcription start site landscapes. *PLoS One*, 6(8):e23409, 2011.
- [192] Andrea Rau, Mélina Gallopin, Gilles Celeux, and Florence Jaffrézic. Data-based filtering for replicated high-throughput transcriptome sequencing experiments. *Bioinformatics*, 29(17):2146–2152, 2013.
- [193] Eugenia G Giannopoulou and Olivier Elemento. An integrated chip-seq analysis platform with customizable workflows. *BMC bioinformatics*, 12(1):277, 2011.
- [194] Haitham Ashoor, Aurélie Hérault, Aurélie Kamoun, François Radvanyi, Vladimir B Bajic, Emmanuel Barillot, and Valentina Boeva. Hmcan: a method for detecting chromatin modifications in cancer samples using chip-seq data. *Bioinformatics*, 29(23):2979–2986, 2013.
- [195] Yong Zhang, Tao Liu, Clifford A Meyer, Jérôme Eeckhoutte, David S Johnson, Bradley E Bernstein, Chad Nusbaum, Richard M Myers, Myles Brown, Wei Li, et al. Model-based analysis of chip-seq (macs). *Genome biology*, 9(9):R137, 2008.
- [196] Microsoft. Dynamic data display, 2009.

Bibliography

- [197] Cornelis Murre, Patrick Schonleber McCaw, and David Baltimore. A new dna binding and dimerization motif in immunoglobulin enhancer binding, daughterless, myod, and myc proteins. *Cell*, 56(5):777–783, 1989.
- [198] Albertha JM Walhout, JM Gubbels, R Bernards, PC Van der Vliet, and H Th M Timmers. c-myc/max heterodimers bind cooperatively to the e-box sequences located in the first intron of the rat ornithine decarboxylase (odc) gene. *Nucleic acids research*, 25(8):1493–1501, 1997.
- [199] Timothy L Bailey. Dreme: motif discovery in transcription factor chip-seq data. *Bioinformatics*, 27(12):1653–1659, 2011.
- [200] Rusty Klophaus. Riak core: building distributed applications without shared state. In *ACM SIGPLAN Commercial Users of Functional Programming*, page 14. ACM, 2010.
- [201] Giuseppe DeCandia, Deniz Hastorun, Madan Jampani, Gunavardhan Kakulapati, Avinash Lakshman, Alex Pilchin, Swaminathan Sivasubramanian, Peter Voshall, and Werner Vogels. Dynamo: amazon’s highly available key-value store. In *ACM SIGOPS Operating Systems Review*, volume 41, pages 205–220. ACM, 2007.
- [202] Vahid Jalili, Matteo Matteucci, Marco J. Morelli, and Marco Masseroli. Musera: Multiple sample enriched region assessment. *Briefings in Bioinformatics*, 2016.

Genome analysis

Using combined evidence from replicates to evaluate ChIP-seq peaks

Vahid Jalili¹, Matteo Matteucci¹, Marco Masseroli¹ and Marco J. Morelli^{2,*}

¹Dipartimento di Elettronica, Informazione e Bioingegneria, Politecnico di Milano, 20133, Milan, Italy and ²Center for Genomic Science of IIT@SEMM, Istituto Italiano di Tecnologia (IIT), 20139 Milan, Italy

*To whom correspondence should be addressed.

Associate Editor: Alfonso Valencia

Received on August 6, 2014; revised on April 24, 2015; accepted on May 4, 2015

Abstract

Motivation: Chromatin Immunoprecipitation followed by sequencing (ChIP-seq) detects genome-wide DNA–protein interactions and chromatin modifications, returning enriched regions (ERs), usually associated with a significance score. Moderately significant interactions can correspond to true, weak interactions, or to false positives; replicates of a ChIP-seq experiment can provide co-localised evidence to decide between the two cases. We designed a general methodological framework to rigorously combine the evidence of ERs in ChIP-seq replicates, with the option to set a significance threshold on the repeated evidence and a minimum number of samples bearing this evidence.

Results: We applied our method to Myc transcription factor ChIP-seq datasets in K562 cells available in the ENCODE project. Using replicates, we could extend up to 3 times the ER number with respect to single-sample analysis with equivalent significance threshold. We validated the ‘rescued’ ERs by checking for the overlap with open chromatin regions and for the enrichment of the motif that Myc binds with strongest affinity; we compared our results with alternative methods (IDR and jMOSAICS), obtaining more validated peaks than the former and less peaks than latter, but with a better validation.

Availability and implementation: An implementation of the proposed method and its source code under GPLv3 license are freely available at <http://www.bioinformatics.deib.polimi.it/MSPC/> and <http://mspc.codeplex.com/>, respectively.

Contact: marco.morelli@iit.it

Supplementary information: [Supplementary Material](#) are available at *Bioinformatics* online.

1 Introduction

Chromatin immunoprecipitation followed by next-generation sequencing (ChIP-seq) is the most commonly used method to study genome-wide chromatin modifications or protein–DNA interactions. Computational tools like MACS (Zhang *et al.*, 2008) or ZINBA (Rashid *et al.*, 2011) are applied on aligned ChIP-seq reads to detect enriched regions (ERs) over the genome (often called ‘peaks’), where the local accumulation of sequencing fragments exceeds that of a background distribution, typically estimated from randomly fragmented chromatin or by performing the ChIP-seq protocol with a

control antibody (Bailey *et al.*, 2013). As the protocol is subjected to multiple sources of noise (Chen *et al.*, 2012), some low-intensity accumulation of reads is possible even in absence of a true interaction with the target. These low-intensity peaks, which are usually present in large amounts, contain therefore a mixture of false positives and true, although weak, interactions; they are typically discarded when stringent thresholds on the peak call are used. This approach leads to the discovery of the strongest interactions only, and might distort the genome-wide picture of the genomic locations of the transcription factor binding sites or histone modifications of interest.

Given the intrinsic noise of the ChIP-seq protocol, it is good practice to repeat every experiment at least twice, as the guidelines of the ENCODE project indicate (Landt *et al.*, 2012). The information contained in replicates can then be used to assess the validity of the peaks obtained from a single sample, especially of those with low-intensity.

In this paper, we propose a novel method to rigorously combine the results of peak calls in ChIP-seq replicates and to obtain new, sample-specific, peak lists taking into account their combined evidence. Our method takes as input, for each replicate, a list of enriched genomic regions and a measure of their individual significance in terms of a *P*-value. Starting from a permissive call, we divide the initial ERs in ‘stringent’ (highly significant) and ‘weak’ (moderately significant), and we assess the presence of overlapping enriched regions across multiple replicates. Non-overlapping regions can be penalised or discarded according to specific needs. The significance of the overlapping regions is rigorously combined with the Fisher’s method to obtain a global score. Finally, this score is assessed against an adjustable threshold on the combined evidence, and peaks in each replicate are either confirmed or discarded (a schematic view of our method is given in Fig. 1 and a visualisation of the results of the method on data from the ENCODE project is shown in Fig. 2). In other words, we are able to ‘rescue’ weak peaks, which would probably be discarded in a single-sample analysis, when their combined evidence across multiple samples is sufficiently strong.

We applied our method to ENCODE datasets from ChIP-seq experiments of the Myc transcription factor in K562 human cells, for which multiple samples with replicates are available. As Myc preferentially binds to a well-defined motif, the choice of this TF allowed us to validate our results through motif analysis and DNase-seq data; finally, we compared our findings with other state-of-art methods. The strong aspects of our method, besides the validity and relevance of the results that it provides, are its simplicity and flexibility, together with its efficiency (few minutes for 2–3 replicates with a few tens of thousands of peaks each).

2 Methods

Here, a brief description of the method and datasets used is given. For more details, see the Extended Methods section in the Supplementary Material.

2.1 Data collection and peak calling

ChIP-seq enriched regions are read from data files in standard Browser Extensible Data (BED) format; besides standard ER format specifications (columns ‘chromosome’, ‘start’, ‘end’, ‘ID’), we require a *P*-value quantifying the significance of each ER, which is usually computed by the peak caller used to identify the ER.

Binary Alignment/Map (BAM) files for the transcription factor Myc in human K562 cells (myelogenous leukaemia) were taken from the ENCODE project repository, for a total of 15 samples obtained in 7 different experiments as summarised in Table 1. Each experiment contained 2 or 3 biological replicates of the same ChIP-seq. Technical replicates were artificially created to test our method, as they were not directly available in the ENCODE repository. Technical replicates were obtained by merging the ENCODE alignment files relative to biological replicates for each of the seven conditions considered above, and then by randomly splitting their reads in two new alignment files. Details about technical replicates, and the process used to generate them, are collected in the Supplementary Table S1.

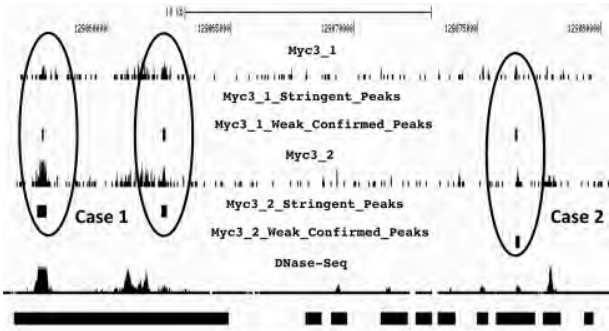


Fig. 2. Genome browser view of a result of the proposed method. Tracks for two ChIP-seq replicates are shown along with the position of the stringent peaks, the weak peaks confirmed by our method and the open chromatin regions (measured as DNase-seq enriched regions). Case 1 refers to a weak peak rescued by a stringent peak, while Case 2 refers to two weak peaks validating each other

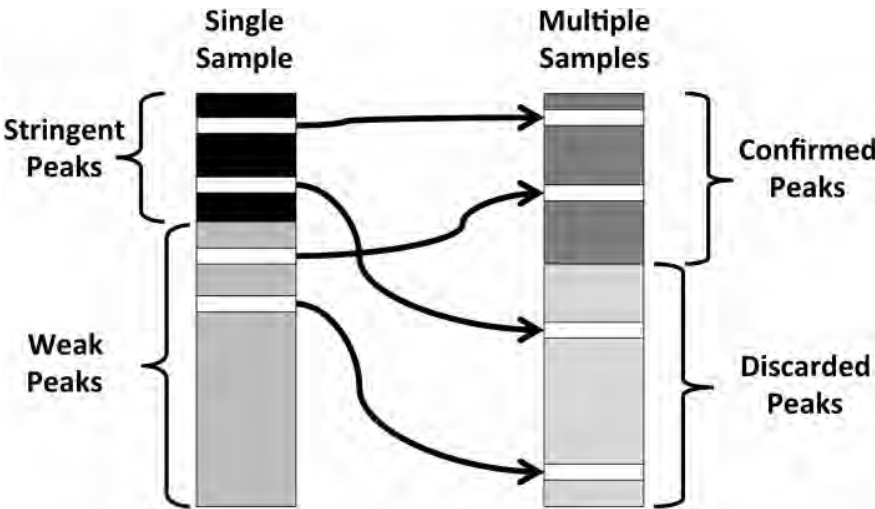


Fig. 1. Pictorial schematic view of the proposed method. First, with a permissive call, we divide peaks of a single individual sample in stringent and weak. Then, combining the evidence of multiple replicates, the peaks in each replicate are confirmed or discarded

Table 1. ENCODE alignment files used and their quantitative features

Sample name	Short name	Aligned reads	R^s	R^w
wgEncodeOpenChromChip K562CmycAlnRep1	Myc1_1	10 719 209	19 171	287 651
wgEncodeOpenChromChip K562CmycAlnRep2	Myc1_2	8 763 362	32 850	311 409
wgEncodeOpenChromChip K562CmycAlnRep3	Myc1_3	9 649 688	13 623	104 911
wgEncodeSydhTfbs K562CmycIggrabAlnRep1	Myc2_1	17 507 194	42 456	64 016
wgEncodeSydhTfbs K562CmycIggrabAlnRep2	Myc2_2	22 256 240	33 015	54 773
wgEncodeSydhTfbs K562CmycStdAlnRep1	Myc3_1	6 077 198	5473	22 965
wgEncodeSydhTfbs K562CmycStdAlnRep2	Myc3_2	5 897 211	12 832	18 753
wgEncodeSydhTfbs K562CmycIfna30StdAlnRep1	Ifna30_1	10 115 596	1901	13 654
wgEncodeSydhTfbs K562CmycIfna30StdAlnRep2	Ifna30_2	18 600 414	2527	97 620
wgEncodeSydhTfbs K562CmycIfna6hStdAlnRep1	Ifna6h_1	9 377 798	5852	12 087
wgEncodeSydhTfbs K562CmycIfna6hStdAlnRep2	Ifna6h_2	19 334 518	4547	102 168
wgEncodeSydhTfbs K562CmycIfng30StdAlnRep1	Ifng30_1	11 602 299	8227	13 190
wgEncodeSydhTfbs K562CmycIfng30StdAlnRep2	Ifng30_2	16 666 560	30 524	25 484
wgEncodeSydhTfbs K562CmycIfng6hStdAlnRep1	Ifng6h_1	14 019 564	2485	13 376
wgEncodeSydhTfbs K562CmycIfng6hStdAlnRep2	Ifng6h_2	19 666 823	27 728	25 118

Peaks were called with the software package MACS2. R^s : stringent ER set (ERs with P -value $< T^s$). R^w : weak ER set (ERs with $T^s \leq P$ -value $< T^w$). $T^s = 10^{-8}$, $T^w = 10^{-4}$.

Peak calling was performed with the software package MACS2 (Zhang *et al.*, 2008) with the following parameters: ‘-auto-bimodal -p 0.0001 -g hs’ (thus setting a p -value threshold of 10^{-4}), using alignment files available in the ENCODE repository, together with the corresponding background (standard input for all samples except for Myc2, for which the input signal from rabbit IgG ChIP-seq was used). In each sample-input pair, the total number of reads was made equal by randomly down-sampling the largest alignment file. The performed call determined between ~ 15 k and ~ 345 k peaks across the different samples (see Table 1).

2.2 Definitions

Given a set of J replicates, each sample j is associated with a set R_j of I enriched regions r_{ji} : $R_j = \{r_{j1}, r_{j2}, \dots, r_{ji}, \dots, r_{jI}\}$. Each region r_{ji} is defined by (*chromosome* _{ji} , *start* _{ji} , *end* _{ji} , *ID* _{ji} , p_{ji}), where p_{ji} denotes a measure of the significance of r_{ji} (i.e. its P -value). T^s is a stringent threshold on P -values, defining a set R_j^s of stringent (highly enriched) ERs; $R_j^s: r_{ji} \in R_j^s$ iff $p_{ji} < T^s$. Similarly, we define a set R_j^w of weak (moderately enriched) ERs, containing all regions whose P -value is between T^s and a weak threshold T^w , with $T^w > T^s$, i.e. $R_j^w: r_{ji} \in R_j^w$ iff $T^s \leq p_{ji} < T^w$. Clearly, $R_j^w \cap R_j^s = \emptyset$ and, if T^w is the maximum P -value allowed for an ER to be associated with sample j , $R_j^w \cup R_j^s = R_j$.

For each region i of each sample j , let $r_{ji,k}$ denote the region of sample k overlapping with r_{ji} , if any. If sample k has multiple regions overlapping with r_{ji} , we choose the most significant one, i.e. the one with the lowest p -value. Let R_{ji} be the collection of $r_{ji,k}$ for $k \in \{1, \dots, J\}$, including r_{ji} itself. Let $K = |R_{ji,*}|$ be the cardinality of $R_{ji,*}$, the set of the ERs intersecting with r_{ji} , with $1 \leq K \leq J$ by definition.

We distinguish between technical and biological replicates of an experiment. Technical replicates aim at controlling the variability of the experimental procedure used to obtain the data and should yield exactly the same results in absence of experimental noise. In a ChIP-seq experiment, this corresponds to performing multiple times the same ChIP protocol on the same biological sample, followed by independent sequencing on the same platform; we expect to observe a significant overlap between ER lists in these samples. Conversely, biological replicates are obtained by applying the same protocol on biologically equivalent samples, what could give rise to different binding profiles of a transcription factor, as in the case of tumor samples; here, the variability in the data can also stem from the ‘true’ biological variation of the phenomenon of interest. Consequently, the lack of overlap between ERs in biological replicates does not necessarily correspond to a false positive result, as it could reflect a true biological interaction occurring only in some samples. With our method, the user is able to control for the required level of overlap and combined significance, according to the specificities of the dataset.

2.3 Algorithm: overall procedure

The main idea behind our method is that repeated evidence across replicates can compensate for a lower significance in a single sample, which is implemented through the Fisher’s method. The Fisher’s method combines the P -values of each test in a global test statistics that follows a chi-squared distribution with $2k$ degrees of freedom (where k is the number of tests combined); therefore, it can be used to falsify the statement ‘all null hypotheses are true’, i.e. ‘all overlapping ERs are due to background noise’. Comparing intersecting ERs from a set of J replicates is equivalent to test the same genomic region in independent experiments against the same null hypothesis H_0 , i.e. ‘the number of reads in the region under study is sampled from the background distribution’, and obtaining independent probabilities of rejecting H_0 (i.e. independent P -values). Here, we briefly outline the structure and motivation of our algorithm, following the flowchart given in Figure 3, while we discuss its details in the Extended Methods (data structures, search algorithms and combining test statistics sections).

We assign every ER r_{ji} in a given sample j to either R_j^s or R_j^w according to its significance. For a given ER, we then determine $R_{ji,*}$ as the set of ERs in the replicates that overlap with r_{ji} , including r_{ji} itself (see Definitions subsection). The cardinality K of $R_{ji,*}$ represents a measure of the reproducibility of the signal in the region spanned by r_{ji} , while the significance of $r_{ji,k} \in R_{ji,*}$ is a measure of the intensity of the signal in a specific replicate k , given the background. We rigorously combine the significance of the overlapping ERs in $R_{ji,*}$ with the Fisher’s method, as described in the Extended Methods, and define a new score for their combined evidence p_{ji}^{comb} . Then, we compare this new score with an adjustable threshold γ : if the desired stringency is obtained, we assign r_{ji} to the set R_j^c of confirmed peaks for sample j ; if the condition is not met, i.e. the combined evidence is not strong enough, we assign r_{ji} to the set R_j^d of discarded peaks for sample j . All the ERs in $R_{ji,*}$ are assigned to the corresponding confirmed R_k^c or discarded R_k^d set, respectively.

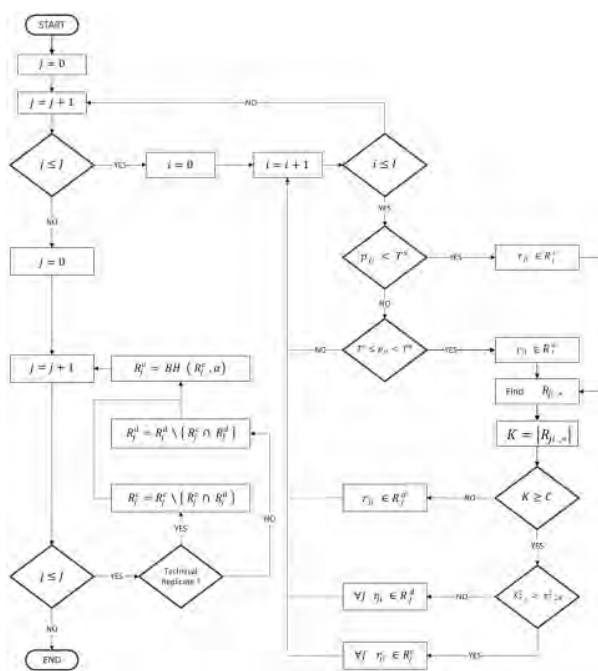


Fig. 3. Flowchart of the proposed method. For the definition of the symbols, see the Definitions subsection of the Methods

We leave the possibility to distrust a region r_{ji} , regardless of its significance, when it is not backed up by the presence of overlapping ERs in a minimum number of samples C . C is an adjustable parameter ranging between 1 and J , with different default values for biological and technical replicates. In summary, for a given sample j , $R_j^c = \{r_{ji} \mid p_{ji}^{comb} \leq \gamma \wedge (K \geq C)\}$ and $R_j^d = \{r_{ji} \mid (p_{ji}^{comb} > \gamma) \vee (K < C)\}$. We repeat this procedure for each sample.

We note that an ER can participate in different sets of overlapping regions, as we discuss in detail in the Extended Methods. As a consequence, it is possible that an ER is assigned to both the confirmed and discarded sets as a result of different tests. These peaks are assigned to the confirmed set if replicates are biological and to the discarded set if replicates are technical. In other words, as technical replicates are supposed to be very similar, for an ER it is enough to fail the test once to be removed from the confirmed set, while for biological replicates this condition is relaxed and it is enough to pass the test at least once for an ER to be confirmed.

After applying the proposed method, each peak has two properties: its initial significance, which can be either *stringent* (s) or *weak* (w), and the result of the multiple replicate comparison, which can be either *confirmed* (c) or *discarded* (d). Then, for each sample we define four mutually exclusive sets on the basis of these property values: $R^{s,c}$, $R^{w,c}$, $R^{s,d}$, $R^{w,d}$, with $R^c = R^{s,c} \cup R^{w,c}$ and $R^d = R^{s,d} \cup R^{w,d}$. The final output set R_j^o of each sample j is obtained by applying the Benjamini–Hochberg correction procedure to R_j^c , independently from the choice of the other parameters, in order to account for multiple testing (Benjamini and Hochberg, 1995), and keeping only the ERs with false discovery rate smaller than an adjustable threshold α .

2.4 Validation

In order to validate the peaks obtained after combining replicates, we first checked whether the peaks we rescued fell within open chromatin by intersecting their genomic coordinates with enriched

regions in DNase-seq data obtained from ENCODE (see Extended Methods). Then, we looked for enriched motifs in the nucleotide sequences spanned by the sets of ERs that we obtained. To perform motif analysis, we used the software package DREME (Bailey, 2011) with parameters ‘-e 0.00001 -m 10’. Results were scored against the JASPAR database (Mathelier et al., 2014) using the software package TOMTOM (Gupta et al., 2007). Myc strongly binds to one specific motif, called the canonical Enhancer-box or E-box, corresponding to the Position Weight Matrices (PWMs) MA0058 (MAX), MA0059 (MYC::MAX), MA0093 (USF1), MA0104 (Mycn) and MA0147 (Myc) in the Jaspas Core Vertebrata database (the corresponding sequence logos are shown in the Supplementary Fig. S1); thus, an ER set is validated when at least one of these PWMs is found significantly enriched in the ER set. We note that Myc has a weaker affinity for other versions of the E-box, but we chose to exclude these other motifs from the validation to achieve maximum stringency.

2.5 Comparison with other methods

Irreproducibility Discovery Rate (IDR) (Li et al., 2011). It is a metric quantifying the reproducibility of a peak across two ChIP-seq replicates by comparing the two lists of ERs, ranked according to their significance. In essence, after calling the peaks, the IDR pipeline uses a bivariate rank distribution to separate the signal (reproducible peaks) from noise (irreproducible peaks) in an experiment (or pairwise comparison). Each peak is associated with an IDR value, which quantifies the probability that the peak belongs to the irreproducible set. IDR was computed for our validation with the scripts provided by Anshul Kundaje at the URL <https://sites.google.com/site/anshulkundaje/projects/idr>

jMOSAICS (Zeng et al., 2013). It is a generic tool for joint analysis of multiple ChIP-seq samples, which can be also used to find common patterns of enrichment between ChIP-seq replicates. First, the MOSAiCS peak caller pre-processes replicates and corresponding control samples by binning the mapped read counts on the genome (default width of 200 bp), and applies the MOSAiCS model fit to each replicate-control pair individually. Afterwards, the jMOSAICS model is applied to the data fitted with MOSAiCS: region-specific enrichment patterns are determined by posterior probabilities assigned to the internal variables, and a binary variable denotes the potential enrichment of a region based on dependencies among samples. jMOSAICS was executed with default parameter values as described at <http://www.bioconductor.org/packages/release/bioc/vignettes/jmosaics/inst/doc/jmosaics.R>. The complete script is included in the Extended Methods.

3 Results

In this section, we report the results obtained by applying our method on either technical or biological ChIP-seq replicates. The method takes as input the genomic coordinates and a measure of significance (i.e. the P -value) of each of the ERs for each replicate considered. For each input sample j , it outputs the lists of confirmed (R_j^o) and discarded (R_j^d) ERs, as well as the lists of stringent confirmed ($R^{s,c}$), weak confirmed ($R^{w,c}$), stringent discarded ($R^{s,d}$) and weak discarded ($R^{w,d}$) ERs.

Adjustable parameters of the method are: T^s (maximum P -value to consider a peak as ‘stringent’), T^w (maximum P -value to consider a peak as ‘weak’), C (minimum number of samples with intersecting peaks needed to apply the combined evidence evaluation), γ (maximum combined significance to confirm a peak), α (maximum false

discovery rate after the Benjamini-Hochberg correction), together with the choice of ‘technical replicate’ versus ‘biological replicate’ mode. For our evaluations we used: $T^s = 10^{-8}$, $T^w = 10^{-4}$, $\gamma = 10^{-8}$, $\alpha = 0.05$ for all comparisons, $C = 1$ for biological replicates and $C = J$ for technical replicates. Required time was a few minutes for 2 samples with 100 000 peaks each on a standard desktop computer.

3.1 Technical replicates

Technical replicates are used to evaluate and remove the noise introduced in the experimental procedure. In the case of ChIP-seq experiments, they are usually generated by performing the same ChIP protocol on the same biological sample, and then performing the sequencing independently. As the ENCODE datasets include only biological replicates, we tested our method on artificial technical replicates, simulated as described in the Methods section. Details about these samples can be found in the [Supplementary Table S1](#). An alternative to our strategy to generate technical replicates would be to randomly split the reads in each original alignment file in replicates rather than merging biological replicates in ENCODE first. However, this procedure gives rise to a much poorer signal, preventing the identification of most ERs. The statistics of these alternative technical replicates are described in the [Supplementary Table S2](#).

Results for $T^s = 10^{-8}$, $T^w = 10^{-4}$, $\gamma = T^s$, $\alpha = 0.05$ and $C = 2$ are shown in [Figure 4](#). For each replicate sample (panels A–G), we show two bars: the left bar (SS) represents the peaks called in a single-sample analysis (R^s in light gray and R^w in dark grey), while the right bar (MS) classifies the same peaks, according to the output of our algorithm, in the four sets described above: $R^{s,c}$ (light gray), $R^{w,c}$ (medium-light grey), $R^{s,d}$ (medium-dark grey) and $R^{w,d}$ (dark grey).

As expected, the number of R^s stringent and R^w weak peaks called in the same technical replicates is always very similar, even if the absolute numbers differ significantly across the different conditions considered. Each output set has a consistent fraction of $R^{w,c}$ weak confirmed ERs, which ranges from 20% to 98% (mean 46%, standard deviation 30%) of the starting number of stringent peaks ($R^{w,c} / R^s$, panel H); thus by combining evidence in replicates, our method ‘rescues’ (i.e. confirms) a large amount of weak co-localised peaks that would otherwise be discarded through a usual single sample evaluation. The percentage of stringent discarded peaks ($R^{s,d} / R^s$, panel H) is very low and varies from 0% in Myc2 to 12% in Myc3 (mean 5.6%, standard deviation 3.8%). The output set R^o corresponds to the set of confirmed peaks $R^c = R^{s,c} \cup R^{w,c}$, where the significance of each peak has been adjusted for multiple testing; combining the evidence present in replicates increases the number of obtained peaks up to almost the double of what obtained with a single sample at the same stringency (R^o / R^s , panel H).

For technical replicates, we expect the output of each replicate to be similar, and therefore the parameter C was set to $C = J = 2$ for all technical replicate comparisons. Setting $C = 1$ would be instead equivalent to trust even isolated peaks, which are not present in the other replicate. With the latter choice, and $\gamma = T^s$, no stringent peaks would be discarded.

As a preliminary evaluation of the results obtained, we considered the overlap of the peaks with the enriched regions in DNase-seq data. On average, 95.4% of the peaks in R^o , 95.4% of peaks in R^s and 94.6% of peaks in $R^{w,c}$ were in open chromatin regions, while this fraction was only 89.4% for $R^{s,d}$ and 93.0% for $R^{w,d}$ ([Supplementary Table S3](#)). The overlap with open chromatin, however, is not yet a validation of a specific binding event. We performed then motif analysis on the nucleotide sequences corresponding to the ERs in the four sets: R^s , R^o ,

$R^{w,c}$ and $R^{s,d}$. Myc is known to bind a large number of sites on the DNA, particularly with high affinity to those with the 6-nucleotide motif called *Enhancer-box* or *E-box*. This protein-binding region has the generic consensus nucleotide sequence CANNTG (with N representing any nucleotide; [Murre et al., 1989](#)). In particular, Myc binds with maximum strength to the CACGTG motif (also called the ‘canonical’ Myc E-box [[Walhout et al., 1997](#)]). Therefore, we consider the enrichment of the E-box in a set of peaks a sufficient condition to consider the set as containing ‘true’ binding sites. Panel I in [Figure 4](#) shows that the E-box is always enriched in the R^s stringent and R^o output sets, as well as in the $R^{w,c}$ weak confirmed set. This result confirms that in the large majority of cases the weak peaks overlapping in replicates identify real binding sites, which are missed by a stringent single-sample call. In 4 out of 14 cases, the $R^{s,d}$ stringent discarded set is enriched for the E-box, although at much lower significance ([Supplementary Table S5](#)), while in the remaining cases the number of peaks in the $R^{s,d}$ set is low. This analysis suggests that the default value $C = J$ used for our artificial technical replicates may be too conservative and still discards a small fraction of real binding sites.

3.2 Biological replicates

The ENCODE data repository always includes one or more biological replicates for each ChIP-seq experiment. For the transcription factor Myc, multiple data sources are available, either obtained in independent experiments, or scored against different backgrounds (in Myc2, the input was derived from immuno-precipitating normal rabbit IgG, while in all the other samples the standard input for the K562 cell line was used). We applied our method to biological replicates obtained from each of the ENCODE experiments considered, and we also combined replicates from 2 experiments (Myc2 and Myc3). Parameters for the method were the same as for the technical replicate evaluation reported in the previous section (i.e. $T^s = 10^{-8}$, $T^w = 10^{-4}$, $\gamma = T^s$, $\alpha = 0.05$); for the additional parameter C , in the case of biological replicates we adopted the permissive choice of $C = 1$ (default for the analysis of biological replicates). With these values (i.e. $\gamma = T^s$ and $C = 1$), our method never discards a stringent peak (we consider that a single strong evidence is enough for biological replicate evaluation). Results are shown in [Figure 5](#).

The number of peaks in biological replicates of the same experiment can be very different (panels A–H), reflecting the different efficiency of the ChIP-seq protocol, and the number of weak peaks (R^w) is usually much larger than the number of stringent peaks (R^s). In the considered cases, the number of confirmed weak peaks ($R^{w,c}$) is often much bigger (up to ~ 4 times) than the number of stringent peaks (R^s) (column $R^{w,c}/R^s$ in panel I), confirming that the evidence in a ‘good’ replicate allows the rescue of many peaks in a ‘bad’ replicate. We observe a similar situation when we combine samples obtained with different inputs. For example, by combining together the four replicates of the Myc2 and Myc3 cases (panel D), we increase massively the number of peaks in the output set for the samples with lower peak counts (Myc3) by confirming a number of their weak peaks much larger than in the evaluation performed without Myc2. Therefore, the presence of high-quality replicates can be of great help in improving the call on many low-quality replicates. The average overlap with open chromatin regions of the $R^{w,c}$ weak confirmed sets is 91.0% (compared with the 51.6% for the weak discarded peaks), and motif analysis confirms that in all the samples the ERs contain the canonical Myc binding site (panel J and [Supplementary Tables S6 and S8](#)).

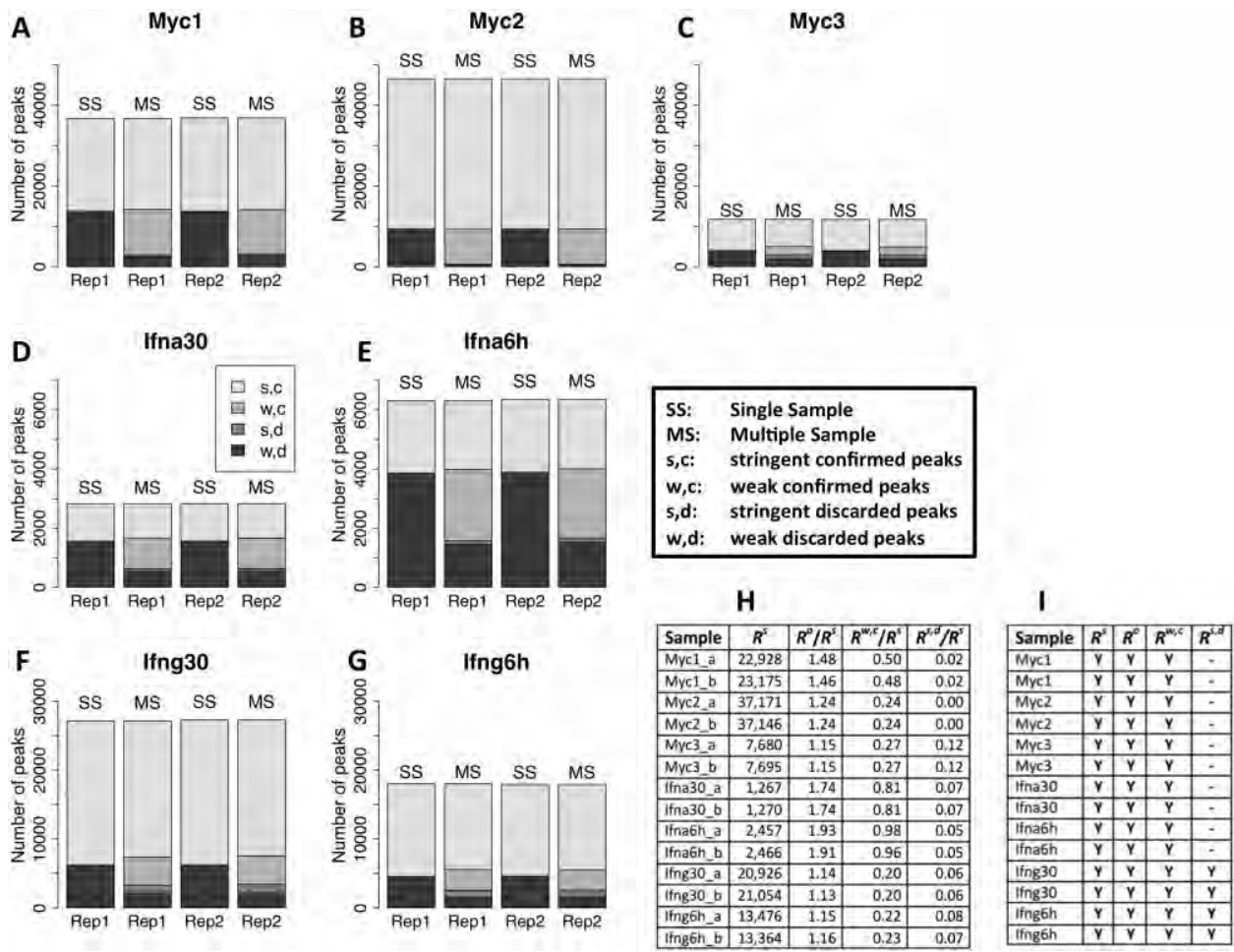


Fig. 4. Technical replicates. For each of the 7 experiments considered, two technical replicates were obtained by pooling reads from the biological replicates of the conditions and then randomly splitting the resulting alignment files in two equal parts. A-G: ER sets for the technical replicates considered. SS, single sample analysis; MS, multiple sample analysis. In each panel, the SS stacked bars represent R^s (light gray) and R^w (dark gray) in the two replicates, while the MS bars show the same peaks, confirmed or discarded according to the output of our method: $R^{s,c}$ (light gray), $R^{w,c}$ (medium-light gray), $R^{s,d}$ (medium-dark gray) and $R^{w,d}$ (dark gray). H, general statistics on the cardinality of the ER sets; I, validation of the sets with the Myc binding motif (Myc canonical E-box); 'Y', presence of the E-box; '-', set too small to find any enriched motif. See [Supplementary Table S5](#) for E-box enrichment *P*-values

We applied our method also using $C=2$ on all biological replicates considered: most of the times that the $R^{s,d}$ stringent discarded peak set had a substantial size, the E-box motif was present, although the average overlap with open chromatin was only 75.9% ([Supplementary Fig. S2](#) and [Tables S9](#) and [S11](#)). This further confirms that, in biological replicates, a lack of overlapping with peaks in other replicates does not necessarily correspond to an artifactual interaction.

3.3 Comparison with alternative strategies

3.3.1 Alignment read merging

An intuitive way to combine evidence in replicates is to merge the alignment reads, and use a peak caller on the combined data. As the combined dataset corresponds to the sum of the two signals, weak, co-occurring peaks should increase their significance. We have merged alignment files from replicates for each considered experiment, using merged backgrounds when available (Myc3 only), and considered only peaks with *P*-value smaller than $T^* = 10^{-8}$. In almost all the cases, the number of peaks called from the merged replicates was substantially lower than the number of peaks obtained by

our algorithm ([Table 2](#), third and fourth columns). Moreover, a large fraction of the peaks detected in the merged samples overlapped with at least one of the peaks obtained by our method ([Table 2](#), fifth column). The only cases when this fraction was below 70% were those where the replicates exhibited a very large difference in the number of called peaks (Myc3, Ifng30, Ifng6h). In these cases, the output set of the replicate with the higher number of peaks always showed a very high overlap with the merged sample peaks. Merging the alignment files therefore 'averages' replicates with different sets of peaks, whereas our method 'rescues' a sample with few ERs with the help of a sample with many ERs. Besides, the merging strategy has no user-defined parameter to tune the results, whereas our method provides a rigorous way to weight co-occurrence and significance of ERs.

3.3.2 Irreproducibility discovery rate

The IDR ([Li et al., 2011](#)) is a measure of the consistency of ERs identified in replicates, which has been systematically assessed in the ENCODE project. We computed the IDR for the ERs in our samples and used an IDR threshold of 0.05. The results are shown in the

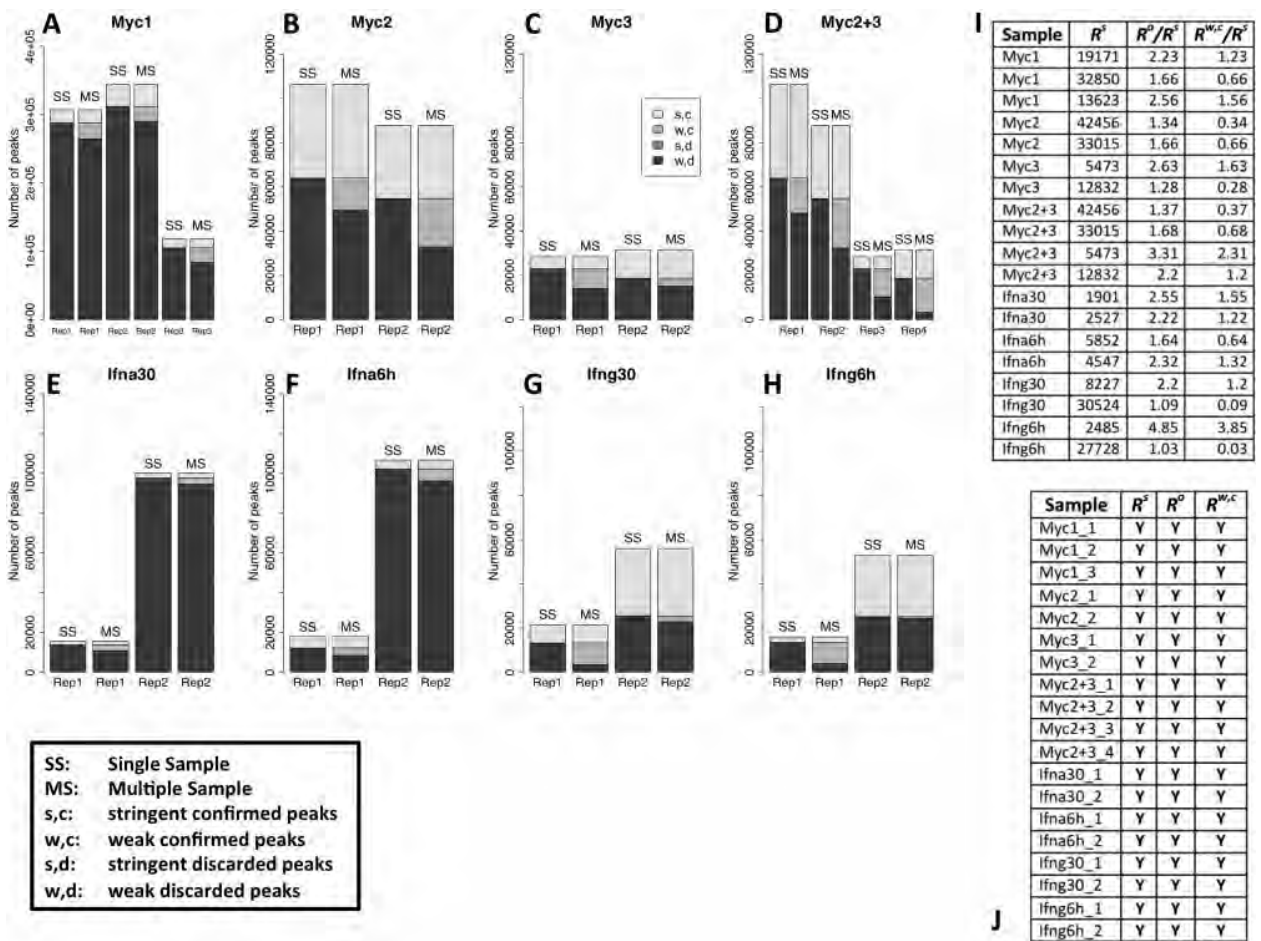


Fig. 5. Biological replicates. A–H: ER sets in the biological replicates considered. SS, single sample analysis; MS, multiple sample analysis. In each panel, the SS stacked bars represent R^s (light gray) and R^w (dark gray) in the replicates, while the MS bars show the same peaks, confirmed or discarded according to the output of our method: $R^{s,c}$ (light gray), $R^{w,c}$ (medium-light gray), $R^{s,d}$ (medium-dark gray) and $R^{w,d}$ (dark gray). I, general statistics on the cardinality of the ER sets; J, validation of the sets with the Myc binding motif (Myc canonical E-box); 'Y', presence of the E-box. See [Supplementary Table S8](#) for E-box enrichment P -values

Table 2. Comparison with merged alignment files and IDR

Sample	R^s	R^o	Merged	$R^o \cap$ Merged	IDR < 0.05	$R^o \cap$ IDR < 0.05
Myc1_1	19 171	42 663	27 958	20 359 (73%)	3097	3097 (100%)
Myc1_2	32 850	54 420	27 958	22 265 (80%)	4618	4618 (100%)
Myc1_3	13 623	34 858	27 958	20 112 (72%)	4966	4964 (99%)
Myc2_1	42 456	56 989	39 805	38 958 (98%)	24 066	23 767 (99%)
Myc2_2	33 015	54 889	39 805	37 765 (95%)	24 066	23 767 (99%)
Myc3_1	5473	14 411	15 404	9 252 (60%)	2356	2237 (95%)
Myc3_2	12 832	16 401	15 404	12 483 (81%)	2356	2237 (95%)
Ifna30_1	1901	4839	3650	2918 (80%)	1171	811 (69%)
Ifna30_2	2527	5605	3650	2914 (80%)	1171	823 (70%)
Ifna6h_1	5852	9570	6633	5913 (89%)	2274	2078 (91%)
Ifna6h_2	4547	10 547	6633	5671 (85%)	2274	2078 (91%)
Ifng30_1	8227	18 112	32 363	15 828 (49%)	5586	5586 (100%)
Ifng30_2	30 524	33 203	32 363	30 307 (94%)	5586	5586 (100%)
Ifng6h_1	2485	12 052	21 145	8506 (40%)	5181	4352 (84%)
Ifng6h_2	27 728	28 564	21 145	20 187 (96%)	5181	5079 (98%)

Comparison of the output set R^o (3rd column) with ERs obtained by merging replicates (4th column) and with ERs having Irreproducibility Discovery Rate (IDR) < 0.05 (6th column). The number of overlapping peaks between R^o and the two other methods (5th and 7th columns, respectively) is shown, together with the fraction of the other method peaks overlapping with peaks in R^o . In general, R^o is larger and comprises the majority of the peaks obtained by the other methods.

Table 3. Comparison with jMOSAiCS

Sample	R^s	R^o	jMOSAiCS	R^o jMOSAiCS	E-box	jMOS AiCS\(R^o	E-box
Myc1_1	19 171	42 663	50 539	7649	Y	31 750	N
Myc1_3	13 623	34 858	33 867	4156	Y	17 249	N
Myc2_1	42 456	56 989	91 252	1346	Y	46 979	N
Myc2_2	33 015	54 889	92 423	799	Y	50 428	Y
Myc3_1	5473	14 411	18 244	2801	Y	9061	N
Myc3_2	12 832	16 401	27 116	1390	Y	13 181	Y
Ifna30_1	1901	4839	32 711	111	-	28 752	Y
Ifna30_2	2527	5605	30 695	527	-	26 616	Y
Ifna6h_1	5852	9570	38 517	114	Y	29 703	N
Ifna6h_2	4547	10 547	36 258	408	Y	28 028	Y
Ifng30_1	8227	18 112	49 843	257	Y	33 763	N
Ifng30_2	30 524	33 203	69 128	150	-	39 996	N
Ifng6h_1	2485	12 052	31 160	483	Y	22 678	Y
Ifng6h_2	27 728	28 564	70 888	98	-	45 601	Y

Comparison of the output set R^o (3rd column) with ERs obtained by jMOSAiCS (4th column). The 5th and 6th columns show the number of peaks that are present in R^o but not in the jMOSAiCS output, and the enrichment of the Myc canonical E-box in this last set, respectively. Vice versa, the 7th and 8th columns show the number of peaks present in the jMOSAiCS output but not in R^o , and the corresponding enrichment of the E-box, respectively. For Myc1, the comparison was done only with replicates 1 and 3. jMOSAiCS outputs a large number of ERs, including most of the peaks identified by our method, but only a fraction of the jMOSAiCS-specific peaks contains the Myc binding motif. Each E-box column refers to the set described in the previous column and is marked as follows: ‘Y’: presence of the E-box; ‘N’: absence of the E-box; ‘-’: set too small to find any enriched motif.

sixth and seventh columns of Table 2. First of all, the IDR-validated peaks are few and often entirely contained in our output sets. This highlights that the IDR method is rather stringent and generates only a small set of validated (reproducible) peaks. We repeated our analysis for the Myc2 sample for a lower T^w threshold and found similar results (see Supplementary Material). Our method does not score the reproducibility of a peak, but it rather combines evidence in replicates, and has the option to accept very stringent peaks even if they are not found in other replicates. We conclude that our method confirms weak peaks that are considered ‘irreproducible’ by IDR (with a 0.05 threshold), and are validated by motif analysis (Figs 4 and 5, Supplementary Tables S3–S11). Finally, differently from our method, which accepts any number of replicates, the IDR can be directly computed only for pairs of replicates.

3.3.3 jMOSAiCS

We compared our results also with those obtained with jMOSAiCS (Zeng et al., 2013), a tool designed to detect combinatorial patterns of enrichment in multiple ChIP-seq samples. Even if jMOSAiCS is conceived to integrate different ChIP-seq datasets that profile distinct features on the same biological sample, it can also be applied to replicates of the same ChIP-seq. Applying jMOSAiCS to our biological replicates (Table 3) resulted in a very large amount of ERs in each experiment, which were on average around 5 times larger than our peaks (ER average size data are not shown). These sets of peaks contained by far the majority of the ERs identified by our method. We checked for the enrichment of the Myc E-box in the peaks identified by our method, but not by jMOSAiCS and vice versa (peaks identified by jMOSAiCS, but not by our method). While in the former case the Myc binding motif was enriched in all the sets with an enough number of peaks to detect any enriched motif, the latter case showed the presence of the Myc canonical E-box only in half of the samples. Moreover, the running times of jMOSAiCS for two replicates were in the order of 3 hours, with about 40 GB of memory consumption, on a server with two Intel Xeon E5-2650 processors and 64 GB of RAM, as this tool starts from alignment files and finds ERs independently. On the same platform, our method ran in a few

minutes; the preliminary peak calling step needed to obtain the sets of enriched regions required about 40 min for each of the replicates with MACS2 (Zhang et al., 2008). We conclude that jMOSAiCS, when applied to replicates of the same ChIP-seq experiment, has the tendency of introducing a large amount of extra peaks, which are not always validated by motif analysis, and it requires significant computational resources. On the other hand our method is much faster, as it allows to start from pre-determined ER lists, and at least equally specific.

4 Discussion

We introduced a novel and rigorous method to combine evidence in ChIP-seq replicates and we applied it to several ENCODE datasets for the transcription factor Myc in the K562 cell line. Our results confirmed that a considerable number of ERs, which display a weak significance in single-sample analysis, can be ‘rescued’ by the help of co-localised evidence in multiple replicates. We proved that the nucleotide sequences spanned by these weak peaks are almost always found in open chromatin regions and enriched for the Myc canonical E-box motif, for which the Myc protein has the highest affinity. Surprisingly, even in the case of technical replicates, where reproducibility should be high, we found that discarding ERs only on the base of the lack of overlap often results in the dismissal of true binding sites. This can be due to the fact that our technical replicates were simulated using a computational procedure starting from the biological replicates available; nonetheless, we recommend to be careful in setting the overlapping parameter to high stringency (i.e. $C = J$).

We stress that our method works as a post-processing of a permissive peak call and it does not question the reliability of the output of the peak caller (any peak caller providing a P -value score can be used; we recommend using the same peak caller with the same parameters on all the replicates). The method has three main strengths: (i) rigour: single-sample evidence from each replicate is combined through the Fisher’s method; (ii) versatility: with the choice of a few parameters, it can be decided to weight co-localisation (C) and combined significance (γ) differently; (iii) efficiency: typically, the

required time is in the order of few minutes on a standard desktop computer and does not require special hardware.

Comparing our method with two other common approaches (replicate merging and IDR) confirmed the stable identification of a core of stringent, reproducible peaks. Besides this, our tests demonstrated that less stringent evidence consistently present across replicates can be combined, leading to the ‘rescue’ of sets of ERs corresponding to real binding sites, e.g. of the transcription factor (TF) Myc. In particular, our results are compatible with those found with IDR, a method widely used in ENCODE to assess the consistency of each detected peak: IDR works by comparing peak rankings and inferring the proportion of reproducible and irreproducible signal in the replicates, while our algorithm provides complementary information by computing the combined significance of a number of overlapping peaks. Despite comparable running times, we differ from IDR as we do not automatically discard non-overlapping peaks and we can directly apply our method to more than two replicates without relying on multiple pairwise comparisons. We stress that our method should not be considered an alternative to IDR, but rather complementary to it.

A further comparison with a tool designed for more complex analysis (identifying combinatorial patterns of enrichment across different ChIP-seq experiments performed over the same biological sample), jMOSAICS, revealed that, in the specific task of comparing replicates of ChIP-seq experiments performed against the same target, this last tool confirms more peaks, which however do not always enrich for the E-box.

Recently, JAMM (Ibrahim *et al.*, 2015), a tool based on local multivariate Gaussian mixture models for directly finding ERs on ChIP-seq replicates, has been introduced. JAMM confirms that pooling replicates can blur the specific spatial resolution of single-sample peaks and lead to less accurate calls in terms of peak width and intensity.

In summary, our strategy represents a promising trade-off between stringent techniques (IDR) and permissive techniques (jMOSAICS).

Acknowledgements

We thank Mattia Pelizzola and Stefano de Pretis for useful discussions.

Funding

This work was supported by the Fondazione Istituto Italiano di Tecnologia and by AIRC [grant no. IG_13182], and it is part of and supported by the ‘Data-Driven Genomic Computing (GenData 2020)’ PRIN project (2013–2015), funded by the Italian Ministry of the University and Research (MIUR).

Conflict of Interest: none declared.

References

- Bailey, T.L. (2011) DREME: motif discovery in transcription factor ChIP-seq data. *Bioinformatics*, **27**, 1653–1659.
- Bailey, T. *et al.* (2013) Practical guidelines for the comprehensive analysis of ChIP-seq data. *PLoS Comput. Biol.*, **9**, e1003326.
- Benjamini, Y. and Hochberg, Y. (1995) Controlling the false discovery rate: a practical and powerful approach to multiple testing. *J. R. Stat. Soc. Series B*, **289**–300.
- Chen, Y. *et al.* (2012) Systematic evaluation of factors influencing ChIP-seq fidelity. *Nat. Methods*, **9**, 609–614.
- Gupta, S. *et al.* (2007) Quantifying similarity between motifs. *Genome Biol.*, **8**, R24.
- Ibrahim, M.M. *et al.* (2015) JAMM: a peak finder for joint analysis of NGS replicates. *Bioinformatics*, **31**, 48–55.
- Landt, S.G. *et al.* (2012) ChIP-seq guidelines and practices of the ENCODE and modENCODE consortia. *Genome Res.*, **22**, 1813–1831.
- Li, Q. *et al.* (2011) Measuring reproducibility of high-throughput experiments. *Ann. Appl. Stat.*, **5**, 1752–1779.
- Mathelier, A. *et al.* (2014) JASPAR 2014: an extensively expanded and updated open-access database of transcription factor binding profiles. *Nucleic Acids Res.*, **42**, D142–D147.
- Murre, C. *et al.* (1989) A new DNA binding and dimerization motif in immunoglobulin enhancer binding, daughterless, MyoD, and myc proteins. *Cell*, **56**, 777–783.
- Rashid, N.U. *et al.* (2011) ZINBA integrates local covariates with DNA-seq data to identify broad and narrow regions of enrichment, even within amplified genomic regions. *Genome Biol.*, **12**, R67.
- Walhout, A.J. *et al.* (1997) c-Myc/Max heterodimers bind cooperatively to the E-box sequences located in the first intron of the rat ornithine decarboxylase (ODC) gene. *Nucleic Acids Res.*, **25**, 1493–1501.
- Zeng, X. *et al.* (2013) jMOSAICS: joint analysis of multiple ChIP-seq datasets. *Genome Biol.*, **14**, R38.
- Zhang, Y. *et al.* (2008) Model-based analysis of ChIP-Seq (MACS). *Genome Biol.*, **9**, R137.

MuSERA: Multiple Sample Enriched Region Assessment

Vahid Jalili, Matteo Matteucci, Marco J. Morelli and Marco Masseroli

Corresponding author. Marco Masseroli, Dipartimento di Elettronica, Informazione e Bioingegneria, Politecnico di Milano, Piazza Leonardo da Vinci 32, 20133 Milano, Italy. Tel.: +39-02-2399-3553; Fax: +39-02-2399-3411; E-mail: marco.masseroli@polimi.it

Abstract

Enriched region (ER) identification is a fundamental step in several next-generation sequencing (NGS) experiment types. Yet, although NGS experimental protocols recommend producing replicate samples for each evaluated condition and their consistency is usually assessed, typically pipelines for ER identification do not consider available NGS replicates. This may alter genome-wide descriptions of ERs, hinder significance of subsequent analyses on detected ERs and eventually preclude biological discoveries that evidence in replicate could support. MuSERA is a broadly useful stand-alone tool for both interactive and batch analysis of combined evidence from ERs in multiple ChIP-seq or DNase-seq replicates. Besides rigorously combining sample replicates to increase statistical significance of detected ERs, it also provides quantitative evaluations and graphical features to assess the biological relevance of each determined ER set within its genomic context; they include genomic annotation of determined ERs, nearest ER distance distribution, global correlation assessment of ERs and an integrated genome browser. We review MuSERA rationale and implementation, and illustrate how sets of significant ERs are expanded by applying MuSERA on replicates for several types of NGS data, including ChIP-seq of transcription factors or histone marks and DNase-seq hypersensitive sites. We show that MuSERA can determine a new, enhanced set of ERs for each sample by locally combining evidence on replicates, and prove how the easy-to-use interactive graphical displays and quantitative evaluations that MuSERA provides effectively support thorough inspection of obtained results and evaluation of their biological content, facilitating their understanding and biological interpretations. MuSERA is freely available at <http://www.bioinformatics.deib.polimi.it/MuSERA/>.

Key words: next-generation sequencing; ChIP-seq and DNase-seq data analysis; combined evidence in replicates; integrated genome browser; genomic data visualization

Background

Next-generation sequencing (NGS) is a multi-purpose technology, which allows precise determination of DNA or RNA sequences within a sample of interest [1]. In particular, some strategies

allow enriching for regions of cellular DNA characterized by some common property: chromatin immunoprecipitation followed by NGS (ChIP-seq) [2] reveals genome-wide DNA–protein interactions and chromatin modifications, e.g. histone marks, while

Vahid Jalili is a PhD candidate at Dipartimento di Elettronica, Informazione e Bioingegneria, Politecnico di Milano, Milano, Italy. His research on tertiary analysis of next-generation sequencing data is focused on systematic solutions for analytical and computational challenges.

Matteo Matteucci is associate professor of pattern recognition and machine intelligence at Dipartimento di Elettronica, Informazione e Bioingegneria, Politecnico di Milano, Milano, Italy. His research includes pattern recognition, machine learning, classification, robotics, computer vision and signal processing. He has co-authored more than 150 scientific international publications.

Marco Morelli is a researcher at the Center for Genomic Science of IIT@SEMM, Fondazione Istituto Italiano di Tecnologia, Milano, Italy. A physicist by background, his research interests span from bioinformatic analysis of next-generation sequencing data to dynamical models and machine learning algorithm for pattern recognition in big data.

Marco Masseroli is associate professor of bioinformatics and biomedical informatics at Dipartimento di Elettronica, Informazione e Bioingegneria, Politecnico di Milano, Milano, Italy. His research interests include distributed Internet technologies, biomolecular databases, biomedical terminologies and bio-ontologies to effectively retrieve, manage, analyze and semantically integrate genomic information with clinical and high-throughput genomic data. He is author of more than 170 scientific articles.

Submitted: 1 December 2015; **Received (in revised form):** 1 February 2016

© The Author 2016. Published by Oxford University Press. For Permissions, please email: journals.permissions@oup.com

DNase I sequencing (DNase-seq) [3] provides a global view of the open chromatin in a cellular sample through the identification of hypersensitive sites. The analysis of these NGS data returns, in both techniques, a list of enriched regions (ERs), often named 'peaks' and defined through their genomic coordinates; usually these peaks are also associated with a statistical significance score, i.e. a P-value. The availability of NGS data has opened the possibility of a comprehensive characterization of genomic and epigenomic landscapes; yet, extracting such biological information from raw data requires the use of complex computational pipelines, which include the identification of the ERs as a key step.

Although NGS experimental protocols recommend the production of at least two replicates for each sequenced sample, specific methods and tools currently used for ER calling (e.g. MACS [4] or ZINBA [5]) usually consider only a single sample at a time, and use global stringent thresholds to eliminate the noise in the data [6]; then, the ERs extracted from individual replicates are compared, and typically only the ERs identified in multiple replicates are retained (e.g. by simple intersection or using the irreproducibility discovery rate (IDR) method [7]). As we recently demonstrated [8], considering single samples and applying individual stringent thresholds lead to the discovery of the strongest ERs only, and it may discard true, although less intense, ERs, which in turn could be picked up by taking advantage of the increased sensitivity provided by replicates. Neglecting weak ERs may eventually distort the genome-wide picture of the genomic locations of the ERs of interest, hamper the significance of the following analyses on the identified ERs and ultimately prevent biological discoveries that could be supported by considering also the true, but less intense, ERs (i.e. genomic features) present in the NGS data. Alternative methods considering multiple samples exist, but were designed for other purposes (e.g. jMOSAICS [9], which was designed to identify combinatorial patterns of enrichment across multiple ChIP-seq samples); they can be used also to discover ERs across replicates, but at the cost of a higher number of not validated peaks (false positives) [8].

Recently, we proposed a novel method that simultaneously considers multiple ChIP-seq replicates for transcription factors (TFs) and rigorously combines local evidence of ERs; the method provides new sample-specific peak lists taking into account the combined evidence of ERs, called with a threshold less stringent than usual [8]. In the tests performed on public ChIP-seq data sets for the Myc TF in K562 human cells, this method allowed to significantly extend the number of detected ERs, with respect to single-sample analysis with an equivalent significance threshold. The newly discovered ERs were validated by motif analysis and overlap with open chromatin regions. Furthermore, comparison with alternative methods (i.e. IDR and jMOSAICS) showed that the method discovers more validated peaks than the former and less peaks than the latter, but with a better validation.

The authenticity of the ERs discovered by combining evidence depends on a variety of factors, including the quality of replicates and called ERs, as well as the choice of parameter values used to combine the evidence. An assessment of the resulting ERs should always be performed: for example, it could be achieved by visualizing the results in a genome browser, inspecting ER nearest-neighbour distributions, and/or comparing the ERs with known genomic annotations (functional analysis). These last two procedures involve the calculation of the distribution of distances between ERs of replicates, or between ERs and known genomic features (e.g. genes, promoters or other

regulatory regions). Such distributions may show, for example, that ERs in different sets are relatively close to each other, but they are not overlapping, or that they are not at specific distances from known genes; if this is not expected (e.g. as in the case of replicates regarding ChIP-seq experiments of TFs), it may suggest a revision of the parameter values used for peak calling, or for combining ER evidence.

Addressing all the above aspects, here we review MuSERA, a novel, broadly useful, advanced graphical tool that efficiently implements, extends and generalizes the original method presented in [8], and, in addition, integrates commonly used analysis features that allow performing easily further assessments, genomic annotations and functional analyses on the identified ERs. Through its intuitive graphical interface, MuSERA provides several graphic displays that help the user in gaining a deeper insight and biological evaluation of the analysis results. We review the main MuSERA features, describing how they are implemented, and we apply MuSERA to several types of data, from ChIP-seq experiments of TFs or histone marks, both narrow and broad, to DNase-seq experiments. Finally, we review and discuss some examples of the analysis of these data using MuSERA, which show the relevance of the additional ERs identified with MuSERA, as well as the efficacy of the graphical displays of the computational results that MuSERA provides in supporting the biological interpretation of NGS experiments.

Notations

An ER is a unique independent entity, denoted by r_{ji} , belonging to the sample $R_j = \{r_{j1}, \dots, r_{ji}, \dots, r_{jJ}\}$, with $U = \{R_1, \dots, R_j, \dots, R_J\}$ being a set of replicates; the index i , with $1 \leq i \leq I$, identifies the regions within a given replicate, and the index j , with $1 \leq j \leq J$, identifies the replicates. An ER is characterized by its genomic coordinates ($chromosome_{ji}$, $start_{ji}$, end_{ji}) and $P-value_{ji}$ (p_{ji}). The significance of r_{ji} is stratified by the 'stringent' (T^s) and 'weak' (T^w) thresholds, with $T^s < T^w$; accordingly, $R_j^s = \{r_{ji} \mid p_{ji} < T^s\}$, $R_j^w = \{r_{ji} \mid T^s \leq p_{ji} < T^w\}$ and $R_j^b = \{r_{ji} \mid p_{ji} \geq T^w\}$ represent the sets of 'stringent', 'weak' and 'background' ERs, respectively, for the replicate sample j .

Let R_{ji}^* be the set of all ERs intersecting with r_{ji} (including r_{ji}), where only the intersecting ER with the lowest/highest P-value of each sample is considered if multiple intersecting ERs exist in a sample, and let $K = |R_{ji}^*|$, where $1 \leq K \leq J$ by definition. According to the method in [8], the significance of an ER in R_{ji}^* is assessed by computing a 'combined evidence' χ^2 statistics (i.e. the sum, over the K ERs in R_{ji}^* , of $-2 \ln p_{ji}$), which, according to the Fisher's combined probability test [10], follows a χ^2 distribution with $2K$ degrees of freedom; the right-tail cumulative probability of this χ^2 distribution defines the ER combined evidence p_{ji}^{comb} , whose comparison with a 'stringency threshold' γ defines 'confirmed' ($R_j^c = \{r_{ji} \mid p_{ji}^{comb} \leq \gamma\}$) and 'discarded' ($R_j^d = \{r_{ji} \mid p_{ji}^{comb} > \gamma\}$) sets of ERs for each replicate sample j . Subsequently, the method generates an 'output set' (R_j^o) for each replicate sample by applying a multiple testing correction procedure on the confirmed ERs of the sample. Additionally, for each replicate sample j , the method defines the following sets: (i) 'stringent confirmed' $R_j^{sc} = \{r_{ji} \mid p_{ji} < T^s \wedge p_{ji}^{comb} \leq \gamma\} \subseteq R_j^c$, (ii) 'stringent discarded' $R_j^{sd} = \{r_{ji} \mid p_{ji} < T^s \wedge p_{ji}^{comb} > \gamma\} \subseteq R_j^d$, (iii) 'weak confirmed' $R_j^{wc} = \{r_{ji} \mid T^s \leq p_{ji} < T^w \wedge p_{ji}^{comb} \leq \gamma\} \subseteq R_j^c$, (iv) 'weak discarded' $R_j^{wd} = \{r_{ji} \mid T^s \leq p_{ji} < T^w \wedge p_{ji}^{comb} > \gamma\} \subseteq R_j^d$, (v) 'multiple-testing confirmed' R_j^{mtc} (with $R_j^{mtc} = R_j^o$) and (vi) 'multiple-testing discarded' R_j^{mtd} (with $R_j^{mtc} + R_j^{mtd} = R_j^o$). In addition to the method from [8], MuSERA provides also a single 'unified output set' (R_j^u) representing the confirmed ERs present in all the R_j^o sets of the

combined replicate samples. This R^{uo} set is built by merging all ERs in all the R_j^o sets (one for each replicate sample), so that, for each group of overlapping ERs in the R_j^o sets, only a single ER is present in R^{uo} , having as left-end and as right-end the left-most left-end and the right-most right-end of the overlapping ERs, respectively. A significance score is assigned to each ER in R^{uo} , calculated by rigorously combining the significance of the overlapping ERs in the R_j^o sets using the Fisher's method [10].

MuSERA features

MuSERA combines replicates to increase the statistical significance of ERs. It assigns ERs to different sets and provides an integrated genome browser for their visualization. Furthermore, for the evaluation of the replicate-combined results, it offers several additional features, including 'genomic annotation and functional analysis of enriched regions', 'nearest enriched region distance distribution' and 'global correlation assessment of enriched regions', for in-depth investigation of each of the ER sets. MuSERA bins distances based on a user-modifiable window size, shows results on tables and plots (supporting user-friendly zoom and pan) and allows operations to be applied on user-selected chromosomes. These and other MuSERA features, including 'interactive and batch execution' as well as 'input/output standard data formats', are reviewed in the following sections, where we show the relevance and utility of MuSERA for biological investigation. MuSERA is a .NET application implemented in C# that runs primarily on Microsoft Windows® and may be run also on other operating systems using an Oracle Virtual Box virtual machine freely provided for non-commercial use.

Combining replicates

To combine ER evidence present in sample replicates, MuSERA extends the method described in [8], and implements it in an efficient multi-threaded environment. Each ER is categorized as 'stringent', 'weak' or 'background' with respect to the significance of the ER according to user-defined stringent (T^s) and weak (T^w) thresholds, with only 'stringent' and 'weak' ERs being considered for replicate evidence combination. The algorithm combines the P-values of intersecting ERs using the Fisher's method [10], if and only if the number of such intersecting ERs is above or equal to a user-defined lower bound (C); accordingly, it assigns the property of 'confirmed' or 'discarded' to each of the intersecting ERs if the combined evidence p_{ji}^{comb} is below or is not below, respectively, the user-defined combined stringency threshold γ (see the 'Notations' section). Besides, overlapping ERs located in a number of samples below the required value of the parameter C are 'discarded'. Each replicate sample contributes to the evidence combination with single evidence only; hence, if a sample has multiple ERs overlapping with a single ER of another sample, only the most/least stringent (according to user definition) overlapping ER of the former replicate is considered for the evaluation of the ER of the latter replicate.

Genomic annotation and functional analysis of ERs

An ER can overlap known genomic loci, like promoters or other regulatory elements of genes. Besides, a gene might be regulated by a TF bound to a DNA regulatory element far from its

promoter (e.g. regulatory elements called 'enhancers' [11] can be located far from transcription start, like for the Sonic hedgehog (Shh) gene in mouse [12]), even interspersed with other non-regulated genes [13, 14]. MuSERA can efficiently assign an ER to the closest up-/down-stream genomic feature [e.g. gene transcription start site, promoter region, Coding DNA Sequence (CDS) or enhancer], thanks to its optimized implementation using an adaptive binning of data (see 'Implementation' section and Figures 1B and 2). Furthermore, MuSERA estimates the 'ER-to-feature overlap score', by determining the number of ERs intersecting with genomic annotations (e.g. known genes, 3'/5' untranslated regions, CDSs, intergenic regions (IGR), introns, promoter regions), or with any experimentally verified binding sites uploaded in MuSERA by the user as annotations in General Transfer Format (GTF). Additionally, it estimates the 'ER-to-feature distance distribution' between the ERs and the closest up-/down-stream features per functional group. All these options allow better biological evaluation of the distribution of the ERs in the genomic context.

Nearest ER distance distribution

MuSERA can compute the ER nearest neighbour distance distribution (NND). In each analysis session consisting of at least two samples, the ERs of each sample are grouped into different sets before ('stringent' or 'weak' set) and after ('stringent confirmed', 'weak confirmed', 'stringent discarded', and 'output' set) the multiple-sample analysis. To estimate the NND, after the user chooses the desired sample(s) and set(s) to be considered, for each ER, MuSERA determines the distance to the nearest ER; an option is available to treat all selected samples and sets either as a single entity or as distinct entities. In the case of single entity, the closest neighbour of an ER could be an ER belonging to any set of any sample of the analysis session. In the case of distinct entities, the closest neighbour of an ER is determined within the same set and sample of the ER.

Global correlation assessment of ERs

The similarity between replicates is frequently assessed either before peak calling, using genome-wide read densities, or after peak calling, using the identified ERs. Pearson's product-moment correlation coefficient (PCC) [15] is a threshold-independent and scale-invariant method [16] commonly used to compute a global correlation assessment between replicates. PCC is also used after the peak calling when binned signal intensities are provided, either in a separate 'wiggle' file per sample or as numerical vectors per identified ER (e.g. data set chipseq_mES of [17]). Similarly, the Jaccard Similarity Coefficient (JSC) is a statistical method for correlation/diversity assessment of samples, consisting on the ratio between the cardinalities of the intersection and the union of two sets; it can be used both before peak calling (e.g. [18] increases genes detection power of RNA-seq data using JSC for global similarity filtering) or as a post peak calling correlation assessment procedure (e.g. [19, 20]).

MuSERA determines both region-level and base-pair-level correlations between all pairs of sets using JSC (see Figures 2 and 3). They are respectively computed as the ratio between the number of overlapping regions (region-level correlation), or genomic bases (base-pair-level correlation), and the total number of regions, or genomic bases, in the considered sets. Base-pair-level correlation is more stringent and is to be preferred when the position of the ERs is known with more certainty, or when

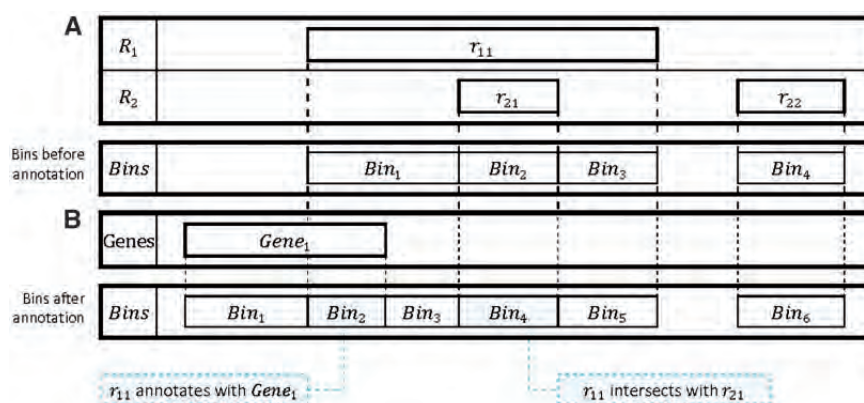


Figure 1. Binned data. (A) A set of bins is created with respect to the ERs of the replicates. (B) The bins are then modified with respect to known binding sites/genomic annotations. Each bin contains all available information for the segment of genome it represents; for instance, in (B), Bin_2 corresponds to r_{11} intersecting with $Gene_1$ at the genome position determined by the Bin_2 coordinates. Bins are orderly stored by their genomic position, which enables a binary search for a specific bin. An ER is possibly represented by more than one bin, i.e. by all bins that start/end within the ER coordinates (e.g. in (B), the ER r_{11} is represented by bins Bin_2 , Bin_3 , Bin_4 and Bin_5); therefore, comprehensive information about an ER is provided by the union of all bins spanning it.

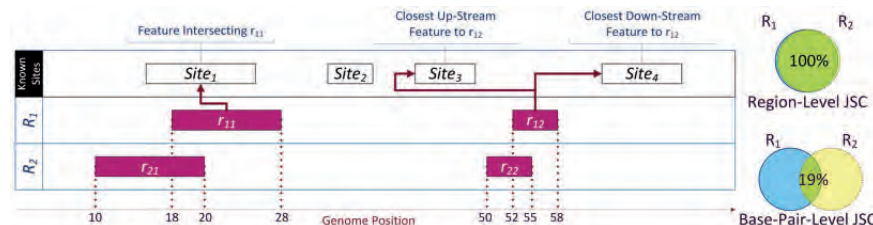


Figure 2. Genomic annotation and correlation assessment. For each ER, MuSERA computes the distance between the ER and the closest known genomic feature (site). If an ER overlaps a feature (e.g. r_{11} and site₁), their distance is 0; otherwise two distances are computed between the ER and the closest up-stream and down-stream features, respectively. MuSERA determines the correlation between samples in terms of the Jaccard Similarity Coefficient (JSC), both at region level and base-pair level. The right-hand side of the figure highlights the possibility of considerable difference between the two levels.

the experimental protocols have a low level of noise. Region-level correlation is instead more permissive, as it scores the overlap of entire regions rather than quantifying the magnitude of this overlap; this correlation measure is then to be preferred in the presence of heterogeneous or noisy data sets.

Interactive and batch execution

MuSERA implements two execution procedures: 'Interactive' and 'Batch' processing. The 'Interactive mode' is provided through a graphical user interface (GUI) with a wide range of review graphical features; it is intended for processing a limited number of samples, where results need to be reviewed through multiple user iterations for parameter tuning (see Figure 4 for cross-functional flowchart). The 'Batch mode' is suitable for processing a high number of samples with a given set of parameters; it reads 'jobs' defined in a simple way through an Extensible Markup Language (XML) file and it has a limited set of review features. The XML file is compliant with the World Wide Web Consortium Document Object Model (DOM) level 1 core and DOM level 2 core recommendations, and its schema has been defined so to ease the work of the end user in the definition of 'jobs'.

Input/output standard data formats

MuSERA processes ERs and allows further investigation of results using genome annotations as references. ERs can be read from tab-delimited files consisting of the ER genomic interval

attributes (chromosome, start, end and P-value) as essential fields; common standard tab-delimited formats such as Browser Extensible Data (BED), ENCODE narrowPeak and ENCODE broadPeak are of such kind. Genome annotations like Reference Sequence (RefSeq) or GENCODE genes can be parsed and loaded from files in standard formats such as the GTF.

MuSERA exports each of the resulting ER sets in a separate BED file. Additionally, an XML file is created for each R_j^o , R_j^f and R_j^d set, containing extensive explanatory information for each included ER, such as (i) ER signature (i.e. chromosome, start, end, name, P-value), (ii) initial categorization (i.e. stringent or weak), (iii) computed combined P-value (X^2) and corresponding right-tail probability (p^{comb}) and (iv) signatures of the ERs it is combined with, including the sample name they belong to. Chromosome-wide basic statistics of each input sample (e.g. widest/narrowest peak, lowest/highest P-value and average/median/standard-deviation of P-values) are provided in a separate text file. When running in 'Batch Mode', MuSERA also exports a text file for each analysis session, providing comprehensive information about the parameters and the overall analysis results for any future reference.

Implementation

Overview

MuSERA is a .NET application written in Model-View-ViewModel (MVVM) pattern [21], with a GUI developed in Windows Presentation Foundation (WPF) graphical system and

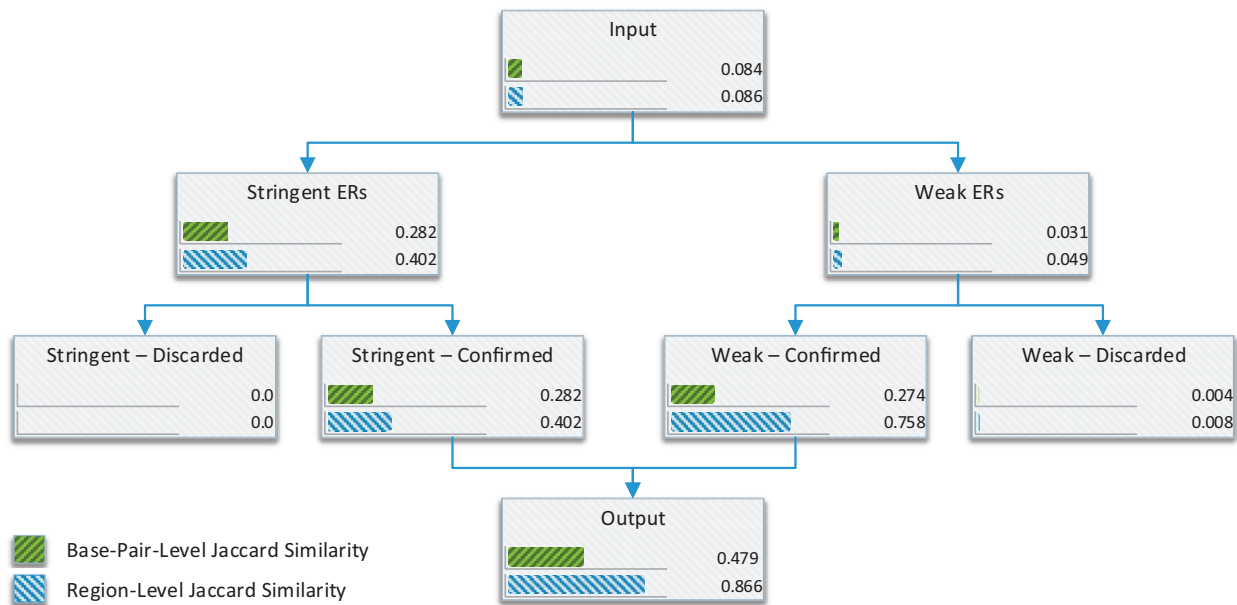


Figure 3. Correlation assessment hierarchy. During processing of two replicate samples, MuSERA estimates the Jaccard Similarity Coefficient between the two samples and for each of their computed ER sets. Values are shown for the ENCODE samples wgEncodeSydhTfbsK562Cmyclfna30StdAlnRep1 and wgEncodeSydhTfbsK562Cmyclfna30StdAlnRep2 (processed with analysis parameters: BioRep, $T^s = 10^{-8}$, $T^w = 10^{-4}$, $\gamma = 10^{-8}$, $C = 1$), which overall show a rather low correlation (Input). In these samples the peaks are called using MACS2.0 [4] with 0.001 P-value threshold; hence, such low correlation is expected because of low signal-to-noise ratio. Initial classification of the ERs in each replicate (i.e. Stringent ERs versus Weak ERs) confirms that in the replicates stronger evidence correlates better than weaker one. Combining the samples, each of the two initial categories is divided into the Confirmed and Discarded sub-categories; ERs in the Confirmed sub-categories result to be considerably more correlated compared with their corresponding ERs in the Discarded sub-categories.

business logic written in C# programming language. Being the GUI implemented in WPF, leveraging on DirectX and on the Graphical Processing Unit in a Multithreaded Apartment (MTA) model, MuSERA delivers a high-end smooth, interactive and user-friendly graphics. The MVVM pattern and MTA model enable separation of business logic and GUI process load, which avoids any possible lag on either side. As far as code metrics are concerned (calculated by Microsoft Visual Studio), MuSERA consists of roughly 6500 lines of code with a maintainability index of 82, cyclomatic complexity of 2.000 and 9 maximum depth of inheritance [22].

MuSERA source code is freely available under open-source GPLv3 license at <http://musera.codeplex.com/>; its implementation for MS-Windows systems and an Oracle Virtual Box virtual machine for its evaluation on other systems (e.g. Linux, Mac) are freely available for downloading for non-commercial use from <http://www.bioinformatics.deib.polimi.it/MuSERA/>, where the MuSERA user manual (Supplementary file 1) is also available.

Interactive and batch execution

The 'Interactive mode' is implemented using the Multi-Threaded Apartment model, while the 'Batch mode' uses the Single-Threaded Apartment model. The two modes use common thread-safe components that enable concurrent execution of modes with no intervention, and the possibility to set the process priority of the 'Batch mode'.

The 'Batch mode' executes a series of 'jobs' collected in an 'at-job' that is defined in an XML file compliant with the DOM specifications. An 'at-job' consists of three parts: (i) properties (e.g. 'Height', 'Width', 'Font size') for all generated plots, (ii) path of the file where the batch log writes and (iii) a collection of

'jobs'. A 'job' is entitled as 'Session' and has three sets of parameters:

- Load and Save parameters, which define the full path of input files and the folder where to save the results; additionally, they enable/disable saving different ER sets to separate files.
- Analysis parameters, which set analysis properties such as T^s and T^w .
- BED parser parameters, which set properties such as P-value column number in input BED files to correctly load them.

A sample portion of an 'at-job' XML file is shown in Supplementary file 2. The 'at-job' is executed by a 'managed code' with least possible footprints, all being memory resources freed-up at 'job' execution termination. Hence, the 'Batch mode' memory requirement is limited to the amount needed for a single 'job' execution.

Determination of intersecting ERs

Cross-replicate, co-localized ERs shall be combined for overall significance determination of evidence; for each ER i of each sample j (i.e. r_{ji}), MuSERA combines the ERs in R_{ji} , i.e. the set of ERs in the replicates that intersect with r_{ji} (including r_{ji}), using the Fisher's method [10]. The set R_{ji} can be determined using various efficient methods, such as algorithms based on ordered lists, i.e. by scanning all lists in parallel and linearly grouping ERs. However, the performance of such algorithms degrades when the intersection size is considerably smaller than the input size, or when input sizes vary significantly between the ER sets [23].

Algorithms based on variants of self-balancing binary search trees, such as interval trees [24] (i.e. an augmentation of red-black trees [25]) or segment trees [26], are asymptotically

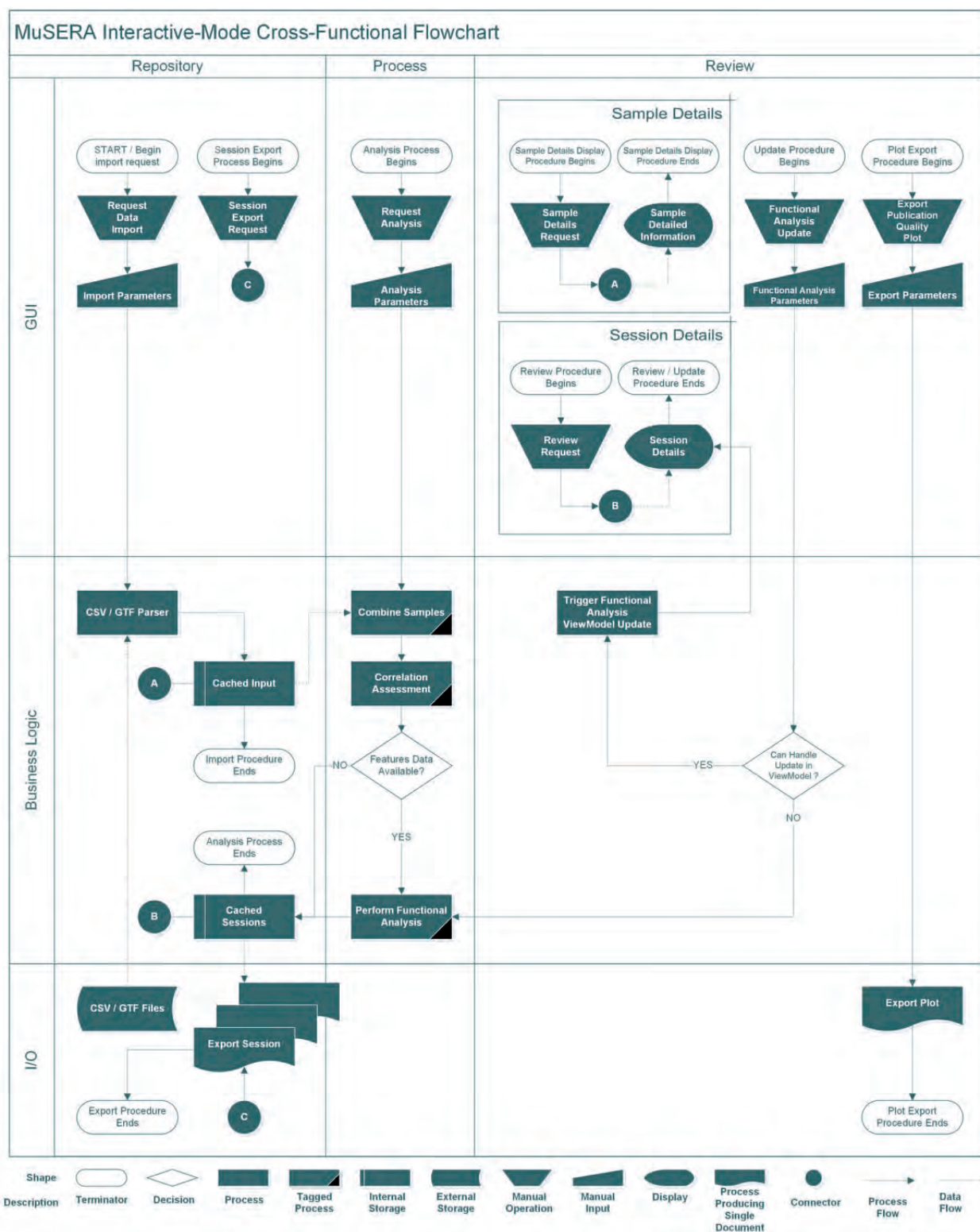


Figure 4. Cross-functional flowchart for the MuSERA Interactive mode. The flowchart shows a simplified flow of the major Interactive mode uses. In the Process part of the Business Logic section of the flowchart, the processes (rectangles) tagged with a black triangle in their bottom-right corner are time-consuming concurrent processes that allow executing other paths while MuSERA is busy computing them.

optimal data structures that store intervals and efficiently support queries for intervals overlapping a given interval/point. An ER is an interval on the genomic domain with respect to its 'chromosome', 'start' and 'end' attributes; this makes interval trees an appropriate data structure for the determination of R_{ji} sets. Additionally, interval trees do not require the input to be sorted, which saves the time for sorting a possibly unsorted input.

MuSERA creates distinct interval trees, one for each chromosome of each replicate. The query time of an interval tree is order of $O(k \log_2 n)$ for reporting k intervals when the tree holds n items. Therefore, R_{ji} determination has $O(J k \log_2 n)$ complexity, as it requires processing J distinct interval trees, each representing the same chromosome for one of the J replicates. Additionally, MuSERA processes chromosomes independently, and hence it is parallelized by distributing individual chromosome processes on available threads.

Genomic annotation of ERs

Once replicates are combined, MuSERA automatically annotates ERs with user-provided genomic features (e.g. genes, promoters, CDSs, binding sites of other TFs), independently for each of the ER sets (e.g. $R_j^s, R_j^w, R_j^c, R_j^d$). MuSERA is an interactive tool, where the user can tune a few parameters to achieve better results; hence, response time to update each annotation parameter should be reasonably fast. MuSERA can linearly group ERs and known binding sites/genomic annotations that overlap; however, this would require re-running the algorithm in case of any user-defined parameter is changed. To avoid this, the genomic annotation algorithm of MuSERA pre-processes data by defining genome-wide dynamic bins with coordinates determined by the ERs of the considered set (Figure 1A) and the known binding sites/genomic annotations (Figure 1B), the bins being stored and sorted according to their 'start' coordinate.

A bin spans a segment on the genome determined by two consecutive start/end coordinates of ERs or genomic annotations (i.e. start-start, start-end, end-start or end-end; see Bin₁, Bin₂, Bin₃ and Bin₅, respectively, in Figure 1B), and it includes all available information for that segment of DNA; hence, it enables constant access for the biological interpretation of the segment. This aspect avoids re-running the annotation process in case of changing any user-defined annotation parameter, such as the filter option (e.g. considering only TF binding sites or CDSs as known binding sites/genomic annotations). Additionally, given

an ER, the corresponding DNA segments (i.e. bins) are determined in logarithmic time, because this requires a binary search on sorted elements (bins), and the element annotations are determined in constant time; therefore, an ER annotation is optimally computed in $O(\log_2 n)$, where n is the number of defined bins.

Integrated genome browser

MuSERA implements also a flexible and highly interactive set of plotting features based on the Dynamic Data Display [27] package, allowing real-time interactive zoom and pan on genome-scale samples. Having combined samples, MuSERA automatically creates bins independently for each of the determined sets (e.g. $R_j^s, R_j^w, R_j^c, R_j^d$), as already shown in Figure 1, and displays in tabular format all the ERs of the sets with their corresponding information (e.g. 'chromosome', 'start', 'end', 'P-value', X^2). By double-clicking on any of the listed ERs, MuSERA plots it together with all the ERs (in different colours according to the set they belong, i.e. 'stringent confirmed', 'stringent discarded', 'weak confirmed' or 'weak discarded') and annotations, if any, within a window of user-defined size (e.g. see Figure 5); then, this can be easily scrolled, panned and zoomed to interactively explore the location on the DNA also of all the other ERs and annotations.

Use case results and practical guidance

In this section, we first illustrate how sets of significant ERs are expanded by applying MuSERA on replicates, for several types of NGS data, such as ERs from ChIP-seq of TFs and broad and narrow histone marks, and DNase-seq hypersensitive sites. We show that MuSERA is able to correctly determine a new set of ERs by locally combining their evidence on replicates, and we prove how the integrated graphical features that MuSERA provides well support thorough inspection of the obtained results and evaluation of their biological content.

Used data sets

We applied MuSERA on publicly available NGS data sets from the ENCODE repository, which always provides at least two biological replicates for each experiment [28]; we considered data sets regarding K562 (acute myelogenous leukaemia) human cells. To test MuSERA against a variety of different types of data and peak shapes, we decided to consider nine different data

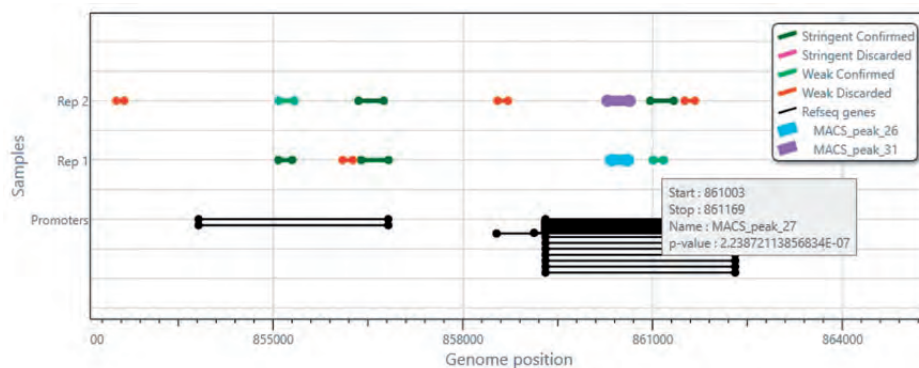


Figure 5. An example view of the integrated genome browser. For a selected ER (e.g. the ER represented by the light blue thick interval on Rep1 line, named MACS_peak_26), the ER(s) it is combined with (e.g. the ER represented by the purple thick interval on Rep2 line, named MACS_peak_31) and all surrounding ERs (coloured according to the set they belong) and available annotations are plotted; hovering the cursor on an ER, a tooltip is opened to show the corresponding information (e.g. start, stop, name, P-value).

sets: two ChIP-seq data sets of the TF CTCF (CTCF1, with three replicates, and CTCF2, with two replicates), one ChIP-seq data set of the TF JunD (JunD, two replicates), one ChIP-seq data set of the RNA Polymerase II molecule, responsible for gene transcription (Pol2, two replicates), one ChIP-seq data set of the histone mark H3K4me3, marking active promoters (H3K4me3, two replicates), which usually generates narrow, TF-like peaks, one DNase-seq data set, corresponding to open chromatin regions (DNaseI, two replicates) and three data sets of histone marks, which are deposited over large genomic regions (H3K9me3 and H3K27me3, two replicates each, marking the body of repressed genes; H3K36me3, two replicates, marking actively transcribed gene bodies). The details of the samples are given in Table 1.

Results of combining ER evidence on replicates and their validation

MuSERA has been run with parameters $T^s = 10^{-8}$, $T^w = 10^{-4}$, $C = 1$ in the 'biological replicate' mode, meaning that the validation of an ER overlapping with ERs in the other replicate samples is sufficient to validate all overlapping ERs. In particular, with the choice $C = 1$ we decided to automatically confirm each stringent peak, regardless of its overlap with peaks in the other replicates, as we found this was the best strategy for the TF Myc [8].

As we can see from Figure 6 (panels A–I), showing the sizes of the different ER sets determined, by combining ER evidence on replicates MuSERA allows the 'rescue' of a large number of peaks ($R^{w,c}$ set, dark green) below the chosen significance threshold T^s in a single sample. The number of 'rescued' (i.e. weak, confirmed) peaks in the output set R^o ranges from 12% to more than the double of the size of the original single-sample stringent set R^s (panel J): the presence of a biological replicate allows a consistent expansion of the set of 'good' peaks by locally lowering the sensitivity threshold. The highest efficiency is found for the Pol2, DNaseI, H3K27me3 and H3K36me3 samples, where the sample output set R^o has more than double (up

to more than triple for H3K27me3) of the peaks in the stringent set R^s .

In all determined ER sets of the CTCF samples, we validated our results by looking for the presence of the CTCF motif (coded as a Position Weight Matrix in the JASPAR CORE Vertebrata database entry MA0139.1 [29]) recognized on the genome in the sequences spanned by the peaks in the different sets. We found the motif enriched in all the CTCF R^s and R^o sets, as expected, but also in all the $R^{w,c}$ and in two of five $R^{w,d}$ sets, even if the P-values of the enrichments are higher (i.e. less significant) in the $R^{w,d}$ set case. This result fully validates the 'rescue' process proposed by MuSERA, and also suggests that our peak call has been rather stringent. The details of the validation results are shown in Table 2.

Use case result evaluation with MuSERA graphical features

Through its graphical interface, MuSERA allows a quick inspection of the analysis results and a thorough evaluation of their biological content. Figure 7 shows the MuSERA 'Overview' panel providing a general overview of the ER sets determined for the CTCF1_1 sample, and including a global view of the parameter values used. All ERs of each set are listed in a table view, together with all their quantitative values and computed statistics; with just a double click, the ERs can be easily displayed in the genomic context along the DNA, thanks to the MuSERA integrated genome browser (Figure 5).

Furthermore, several other quantitative features that MuSERA automatically computes can be straightforwardly displayed; some of them are shown in Figure 8: the stratification of the ER sets over the different chromosomes (panel A), the distribution of the combined significance (X^2) of the ERs in each set (panel B, Output Set of the CTCF1_1 sample) and the distribution of the distance of the ERs in each set from the closest genomic feature chosen (panel C, Output Set of the CTCF1_1 sample; the

Table 1. ENCODE alignment files used and their quantitative features

Sample name	Short name	Aligned reads	R^s	R^w
wgEncodeOpenChromChipK562CtcfAlnRep1	CTCF1_1	6 051 439	53 339	22 290
wgEncodeOpenChromChipK562CtcfAlnRep2	CTCF1_2	6 211 475	57 104	26 177
wgEncodeOpenChromChipK562CtcfAlnRep3	CTCF1_3	11 988 569	66 262	36 278
wgEncodeSydhTfbsK562CtcfIggrabAlnRep1	CTCF2_1	26 957 114	58 089	45 727
wgEncodeSydhTfbsK562CtcfIggrabAlnRep2	CTCF2_2	26 437 775	52 386	34 130
wgEncodeSydhTfbsK562JundbIggrabAlnRep1	JunD_1	16 175 565	48 152	67 154
wgEncodeSydhTfbsK562JundbIggrabAlnRep2	JunD_2	28 086 672	66 936	59 105
wgEncodeSydhTfbsK562Pol2IggmusAlnRep1	Pol2_1	17 762 352	18 392	53 489
wgEncodeSydhTfbsK562Pol2IggmusAlnRep2	Pol2_2	19 293 573	21 810	61 160
wgEncodeBroadHistoneK562H3k4me3StdAlnRep1	H3K4me3_1	9 512 593	28 595	28 271
wgEncodeBroadHistoneK562H3k4me3StdAlnRep2	H3K4me3_2	15 640 462	35 285	34 526
wgEncodeOpenChromDnaseK562AlnRep1	DNaseI_1	9 993 542	41 184	133 829
wgEncodeOpenChromDnaseK562AlnRep2	DNaseI_2	29 472 357	56 800	146 113
wgEncodeBroadHistoneK562H3k9me3StdAlnRep1	H3K9me3_1	15 816 227	2428	3324
wgEncodeBroadHistoneK562H3k9me3StdAlnRep2	H3K9me3_2	33 939 687	1978	8555
wgEncodeBroadHistoneK562H3k27me3StdAlnRep1	H3K27me3_1	12 210 065	1969	6916
wgEncodeBroadHistoneK562H3k27me3StdAlnRep2	H3K27me3_2	12 119 288	21 554	25 603
wgEncodeBroadHistoneK562H3k36me3StdAlnRep1	H3K36me3_1	14 803 144	12 606	10 365
wgEncodeBroadHistoneK562H3k36me3StdAlnRep2	H3K36me3_2	10 393 298	4435	10 189

Peaks were called with the software package MACS2.0 [4] using the parameters '-auto-bimodal -p 0.01 -g hs' (thus setting a P-value threshold of 10^{-2}). R^s : stringent ER set (ERs with P-value $< T^s$). R^w : weak ER set (ERs with $T^s \leq \text{P-value} < T^w$). $T^s = 10^{-8}$, $T^w = 10^{-4}$. Peaks for the histone marks H3K4me3, H3K9me3, H3K27me3 and H3K36me3 were called with the 'broad' option.

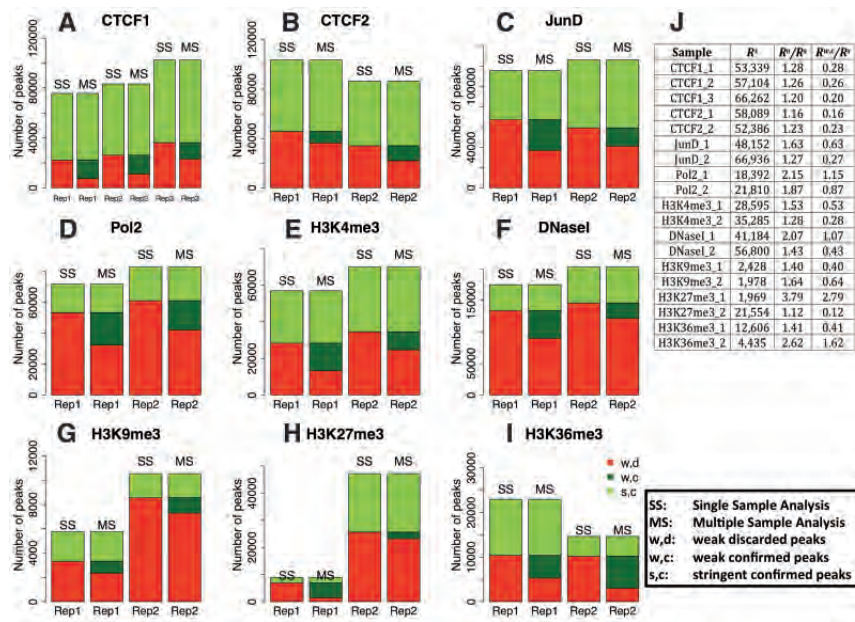


Figure 6. ER sets for the considered data sets. (A–I) ER sets in the testing data sets considered (biological replicates). SS: single sample analysis; MS: multiple sample analysis. In each panel, the SS stacked bars represent R^s (light green/gray) and R^w (green/gray) sets in the replicates, while the MS bars show the same peaks, confirmed or discarded according to the MuSERA output: $R^{s,c}$ (light green/gray), $R^{w,c}$ (dark green/gray) and $R^{w,d}$ (red/black) sets. Note that setting the parameter $C = 1$, the $R^{s,d}$ set is always empty. (J) General statistics on the cardinality of the ER sets. See Table 2 for the validation results of the CTCF peaks.

Table 2. Validation results of the peaks for the CTCF samples

Sample	R^s	R^o	$R^{w,c}$	$R^{w,d}$
CTCF1_1	7.4 e-3575	3.4 e-3786	5.2 e-338	–
CTCF1_2	1.6 e-3586	2.2 e-3846	4.1 e-147	–
CTCF1_3	8.6 e-4115	3.8 e-4321	6.9 e-199	2.4 e-128
CTCF2_1	2.6 e-3610	5.2 e-3676	2.7 e-116	9.4 e-67
CTCF2_2	7.5 e-3414	1.0 e-3517	2.8 e-267	–

P-values for the enrichment of the CTCF binding motif (JASPAR CORE Vertebrata database entry MA0139.1), as estimated with the DREME package [30] in the 150bp around the peak midpoint. R^s : stringent ER set. R^o : output ER set. $R^{w,c}$: weak, confirmed ER set. $R^{w,d}$: weak, discarded ER set. ‘–’: no enriched motif.

chosen genomic feature is the set of promoters in the human genome hg19).

The ‘genomic annotation and functional analysis of enriched regions’ and the ‘nearest enriched region distance distribution’ that MuSERA supports can provide better understanding and improved biological interpretation of the obtained results. For example, looking at the peak-to-peak distance across the different samples (Figure 9, panel A: narrow ERs; panel B: broad ERs), which MuSERA automatically quantifies to build the ‘nearest enriched region distance distributions’, we can see that this quantity is higher in weak confirmed peaks than in stringent confirmed peaks for the considered samples of the CTCF TF and H3K4me3 and H3K9me3 histone marks; whereas, this distance is roughly the same for the DNase I Hypersensitive Site (DHS), RNA Polymerase II and for some of the broad histone mark samples considered. This probably depends on the fact that the stringent confirmed CTCF peaks correspond to high-affinity binding sites of the TF, while the CTCF weak peaks could be generated by transient interactions with the DNA, which are not stabilized by a specific target, and therefore are scattered across the genome. A similar argument holds for the H3K4me3 and H3K9me3 histone marks, although in this case the strength

of the signal is just an indication of the fraction of cells bearing the modification, and it is more difficult to identify the mechanism responsible for this difference; a good guess is that it could be related to the local balance of the enzymes transferring and removing the methyl groups to the histone proteins. On the other hand, DHS ERs are found throughout the genome and do not have preferred genomic locations where the signal is stronger; therefore, in this case the peak-to-peak distance distribution is similar for strong and weak peaks. The mixed behaviour of sample H3K27me3 may depend on the large differences in the number of peaks across the two replicates: replicate 1 almost quadruplicates its number of ERs in the R^o set, thanks to the high number of ERs in replicate 2, and probably the few stringent confirmed peaks were more scattered around the genome than the many weak confirmed peaks. A similar case, although with lesser intensity, may hold true for replicate 2 in sample H3K36me3.

The case of RNA Polymerase II is rather surprising: RNA Polymerase II is the molecule transcribing the genome, and it is mostly localized on genes and promoters, although recent studies indicate that most of the genome has the potentiality of being transcribed [31]. To gain a better insight on this aspect, we took advantage of the ‘genomic annotation and functional analysis’ available in MuSERA to inspect the genomic location of the RNA Polymerase II peaks. Using the ‘ER-to-feature overlap score’ that MuSERA automatically calculates when promoters, intragenic regions or IGR are selected as genomic features, respectively, we found that the fraction of RNA Polymerase II peaks located on these regions is unchanged in the stringent and weak ER sets and across the replicates (Figure 10). Thus, the features that MuSERA computes and graphically shows enabled us to conclude that RNA Polymerase II binds with a wide range of intensities to both genes and intergenic elements.

Finally, the ‘global correlation assessment’ provided by MuSERA, through the evaluation of the JSC for each type of ER set determined, confirms that the obtained output sets are

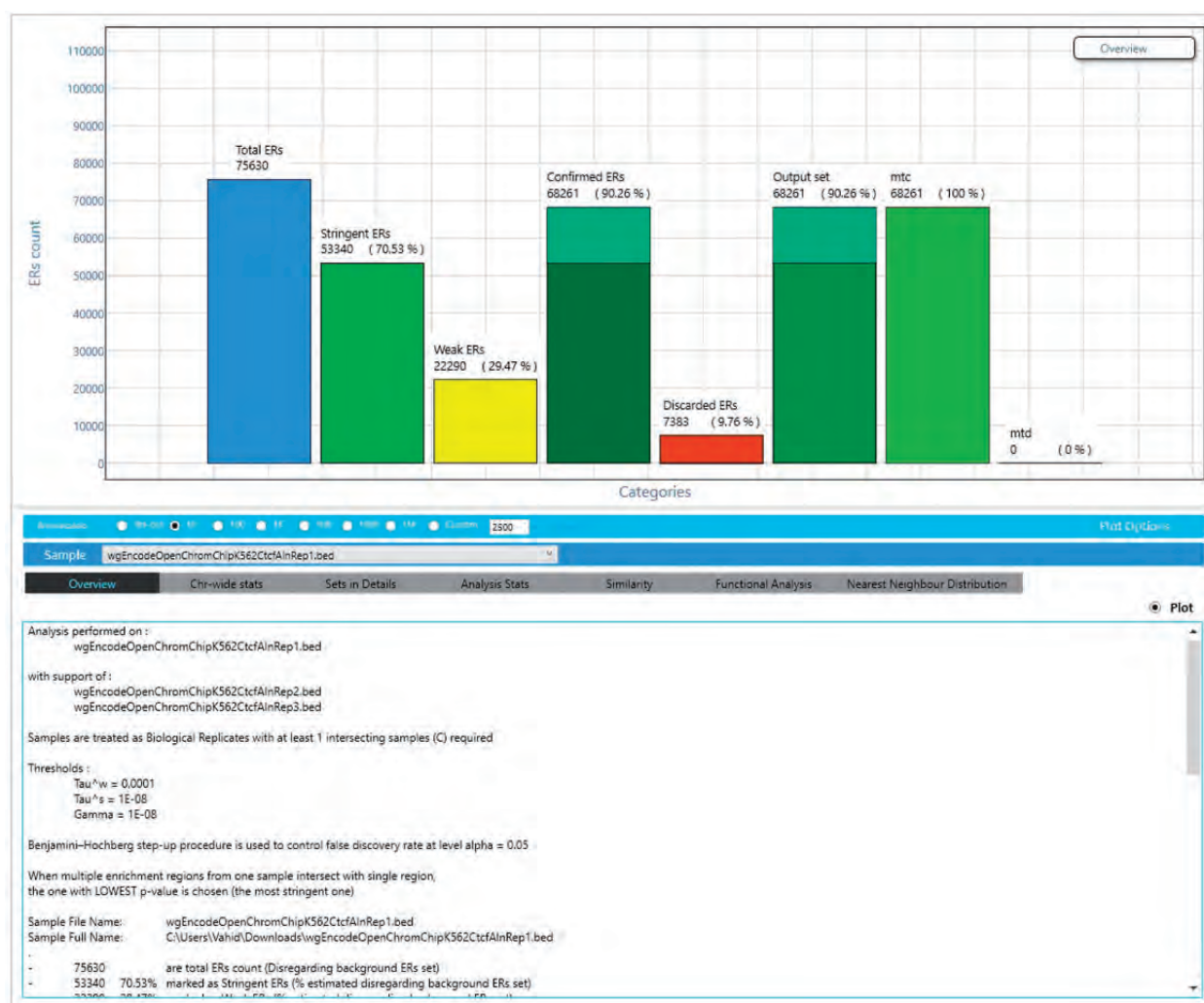


Figure 7. Overview panel of the MuSERA graphical interface. After an analysis is performed, MuSERA shows the statistics of the analysed sets in the 'Overview' panel. Data shown regard the CTCF1_1 sample.

much more congruent than the stringent sets, which would have been used as outputs in the absence of MuSERA (Figure 11, left panel); besides, even for weak ERs, which may include a higher fraction of spurious binding sites, the JSC value increases considerably for the weak confirmed sets, confirming the validity of the 'rescue' process that MuSERA performs (Figure 11, right panel). The evidence in the replicates of a data set is therefore combined in sets of ERs that are more coherent between themselves than the outputs of single-sample analyses.

Performance

We benchmarked the MuSERA performance for a variety of operations, including loading data, combining replicates and pre-processing of analysis results for further assessments through e.g. genomic annotation, similarity search or integrated genome browser. Tests were performed on a standard laptop computer running Microsoft Windows® 10, with Intel® Core™ i3 (2.10 GHz) CPU and 6 GB of RAM. The benchmark was performed on multiple ENCODE ChIP-seq and DNase-seq data sets regarding K562 human cells, including two to three replicates each, where the

overall number of ERs in the replicate samples of each data set spanned few thousands to millions of ERs. Additionally, a data set of human genome hg19 promoters (counting 82 960 promoter regions) was imported for genomic annotation performance benchmarking.

In general, MuSERA performance is in the scale of seconds, spanning few tens to hundreds of seconds depending on operation and number of ERs on replicates, from few tens of thousands to millions of ERs (Figure 12). The process of parsing and loading ERs from input sample files runs in a handful of seconds for most samples. The algorithm of combining replicates is highly optimized and runs in few tens of seconds for two to three replicates with a few hundreds of thousands of ERs each. The correlation between replicates is assessed once replicates are combined; the algorithm runs instantaneously (hence it is not explicitly included in Figure 12). Some operations (e.g. nearest neighbour search, genomic annotation and genome browsing) depend on a data structure that is automatically populated once an analysis session is selected. Such process is executed in background to minimize its effect on other MuSERA independent operations and maximize user experience (i.e. the user can benefit the other independent features of MuSERA while the

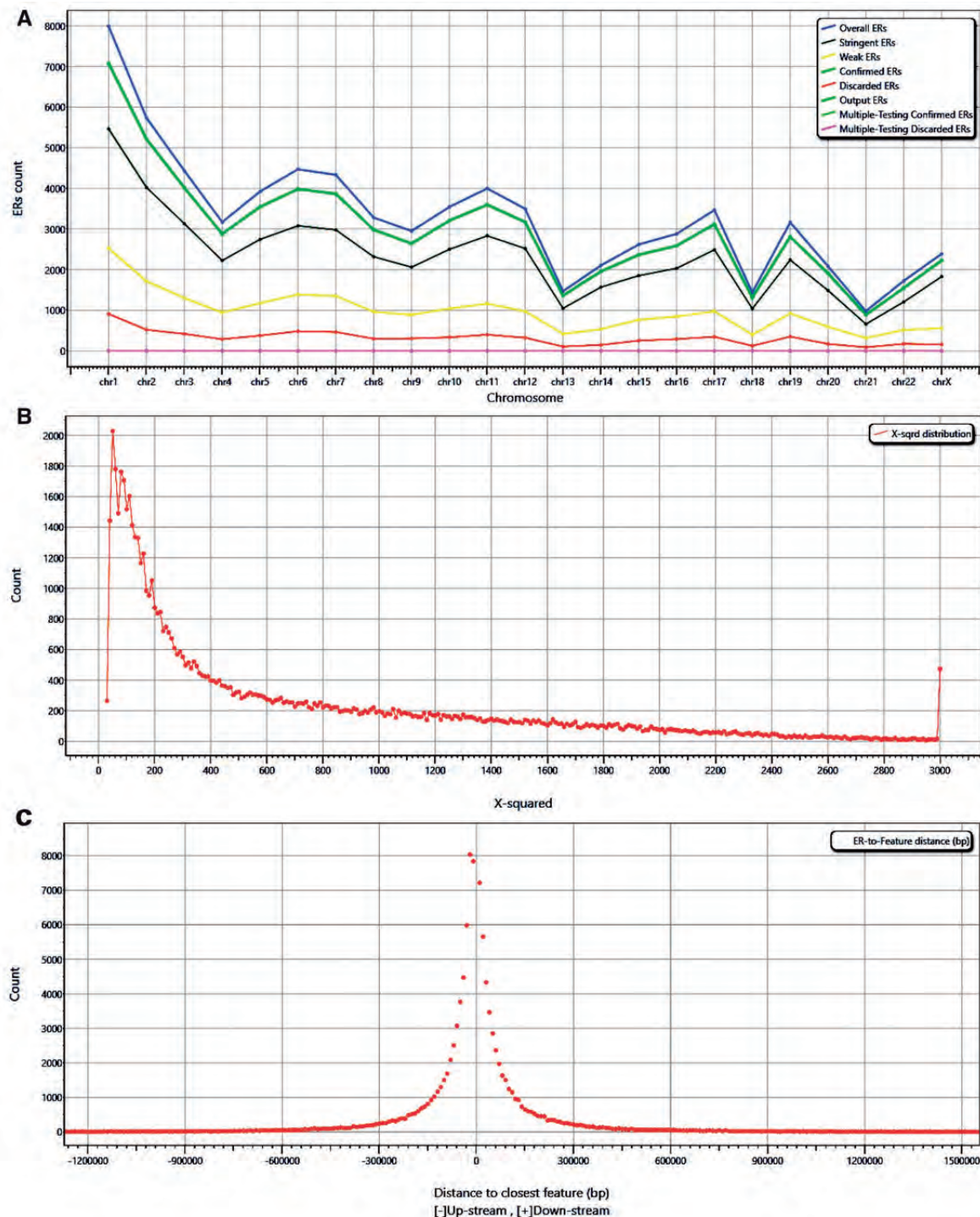


Figure 8. Some graphical analyses performed by MuSERA. The different panels of MuSERA plot the features computed in the analysis. For example, we show here (A) elements in the sets, stratified by chromosomes; (B) distribution of the combined significance (X^2) of the output set (R^0) ERs for the CTCF1.1 sample; (C) distance of ERs in the output set (R^0) of the CTCF1.1 sample from human promoters: clearly, the CTCF TF prefers to bind the DNA close to the regulatory regions of a gene.

required data structure is being populated in background); the process completes in few tens of seconds, depending on the number of considered ERs and genomic features. Once the data structure is populated, the operations, such as genome browsing, are instantaneous.

Discussion and conclusions

We reviewed MuSERA, an effective, efficient and easy-to-use graphical tool of broad utility to combine evidence across ChIP-seq or DNase-seq replicates, and to evaluate them and their

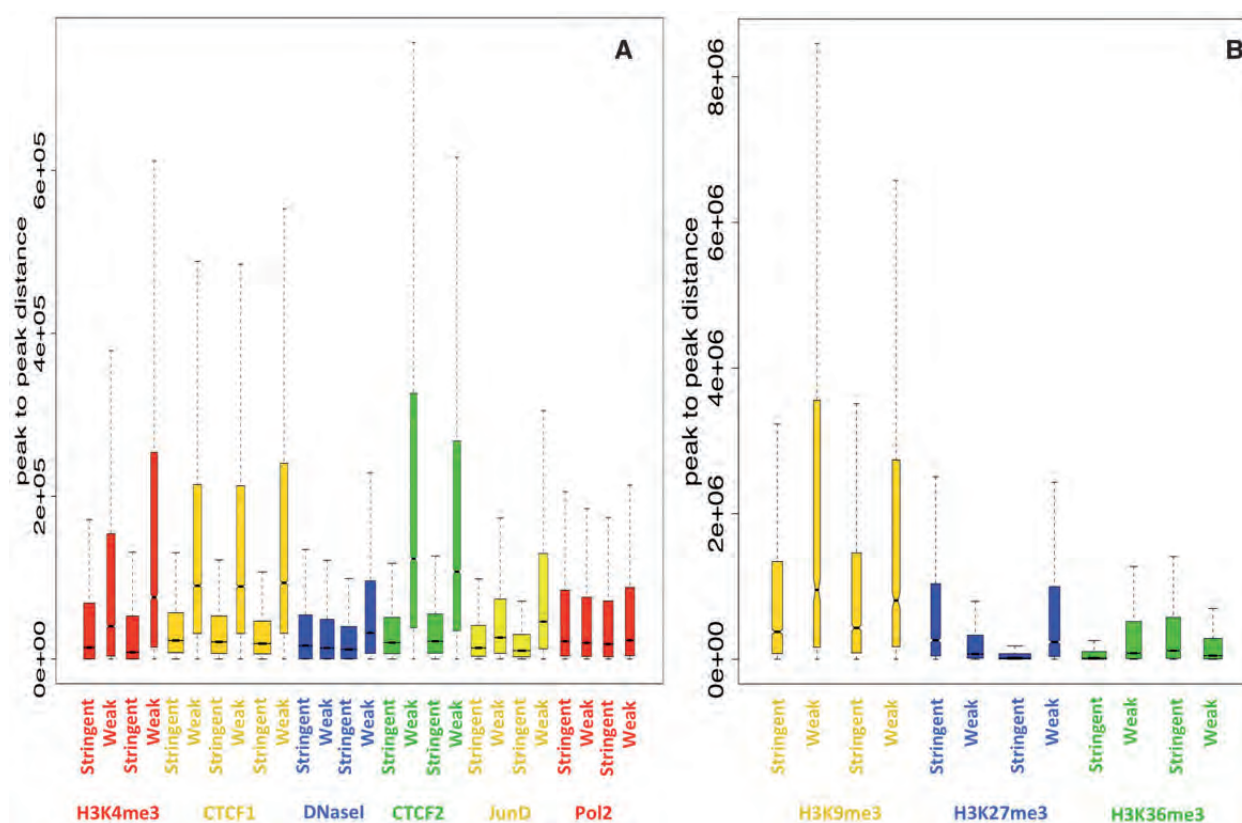


Figure 9. Peak-to-peak distance. The boxplots represent the peak-to-peak distance for the stringent confirmed and weak confirmed ER sets for all the samples considered. Samples displaying narrow, Gaussian-like peaks (A) are shown separated from samples having ERs with a broader shape (B). While this distance is on average greater in the weak confirmed ER sets for the considered TFs (CTCF and JunD) and the H3K4me3 and H3K9me3 histone marks, it stays roughly constant in the two sets for DHS and RNA Polymerase II, and in the remaining histone marks.

biological relevance in the genomic context. MuSERA allows the annotation of samples with user-defined genomic features and the visualization of results in an integrated genome browser. Furthermore, it provides a rich set of quantitative evaluations and interactive graphical displays, which greatly help the understanding and biological interpretations of results.

Common tools used to analyse NGS data are usually designed for scientists with training in bioinformatics or other quantitative disciplines, as they usually involve command-line interfaces and heavily rely on extensive coding abilities. This naturally poses a barrier against biologists who generated the data, and would like to directly perform simple analyses on them. Only few tools make use of a GUI to reach out to larger audiences, the Galaxy project being the most prominent example [32, 33]. However, this large, all-purpose tool can become rather complex to use despite the presence of a GUI, and usually requires powerful computing facilities to run analysis applications on NGS data files, which are typically large. MuSERA, on the other hand, is a dedicated tool efficiently performing integrated analysis of replicated NGS data sets involving ERs, which can be directly used on any personal computer and mastered in a short time. Some tools, like Nebula [34], provide a more focused GUI centred on the processing of ChIP-seq data, and yet, they do not consider the presence of replicates. This same and relevant limitation generally applies to the available tools commonly used for NGS data evaluation, including GenometricCorr [35], which is focused on the detection of genome-wide correlations between pairs of samples; it includes

four different methods to compute these correlations ('relative distance', 'absolute distance', 'projection' and 'jaccard'), together with appropriate null models and statistical tests to evaluate the significance of the correlations. We note that the 'ER-to-feature overlap score' implemented in MuSERA can be thought of a specific case of the 'absolute distance' method implemented in GenometricCorr, where all distances >0 (i.e. considering only non-overlapping regions in the pair of samples considered) are discarded. Some other tools, like PAPST [36], focus on co-localization of different types of ERs, but do not consider the significance of the ERs in their analysis. Instead, MuSERA uniquely combines a rigorous approach for jointly evaluating ERs in replicates [8] with an intuitive GUI and an array of useful downstream analyses, both computational and graphical. Moreover, it leverages on high-end data structures to minimize the runtime of common analysis procedures, and executes time-consuming operations in the background, resulting in high user-friendly interaction with minimal lag. Additionally, while batch processing on common tools requires scripting and/or coding knowledge, MuSERA facilitates batch execution specification by providing a simple XML structure to define batch jobs.

We applied MuSERA to ChIP-seq data sets of TFs and histone marks, and to a DNase-seq data set, and we found that the efficiency of the 'rescue' of weak ERs varies between 12% and 279%, thus potentially making a big impact on the final list of sample-specific confirmed ERs. Variation of MuSERA efficacy depends on many factors, including the quality of replicates and the

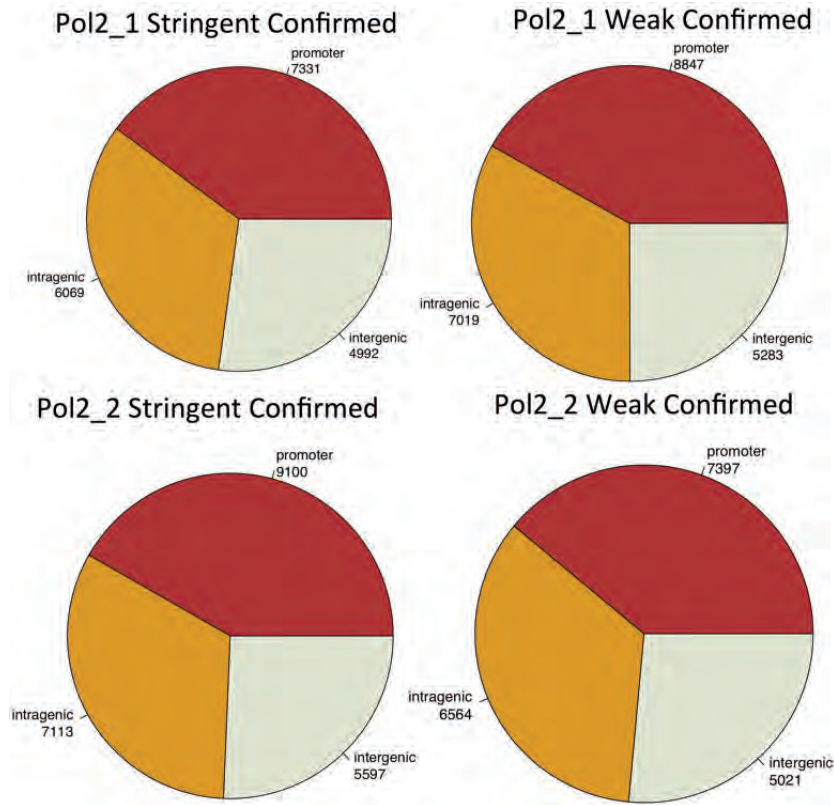


Figure 10. Distribution of RNA Polymerase II ERs in the genome. RNA Polymerase II (Pol2) ERs fall mostly around genes (promoter, intragenic), but a considerable fraction is located in intergenic regions. This behaviour is highly conserved across the two replicates considered and across stringent confirmed and weak confirmed ERs.

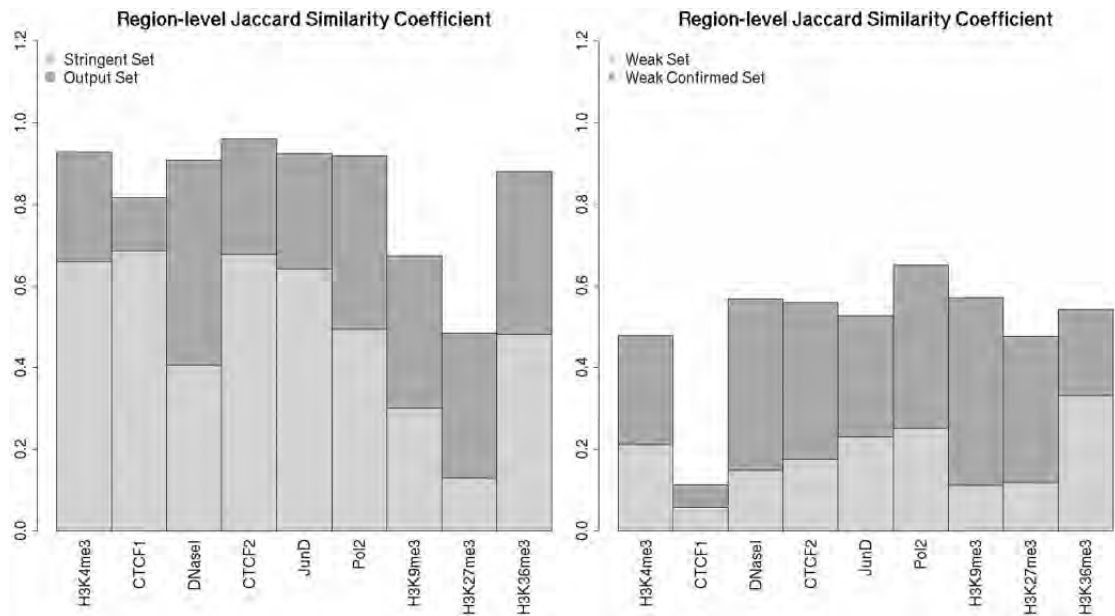


Figure 11. Region-level Jaccard Similarity Coefficient (JSC). The JSC measures the similarity between two or more sets, and it is automatically computed by MuSERA for any type of ER set determined. The figure shows that, for each of the data sets considered, the similarity between the output sets is much higher than the similarity between the stringent sets (left panel). For the sets of weak ERs (right panel), which usually contain a higher fraction of binding sites, the JSC value is rather low, but it considerably increases for the weak confirmed sets, supporting the validity of the evidence-combining process that MuSERA performs. Finally, we note that the CTCF1 data set, which has three replicates, has a lower JSC value owing to the evaluation of an additional sample in the overall ER overlaps.

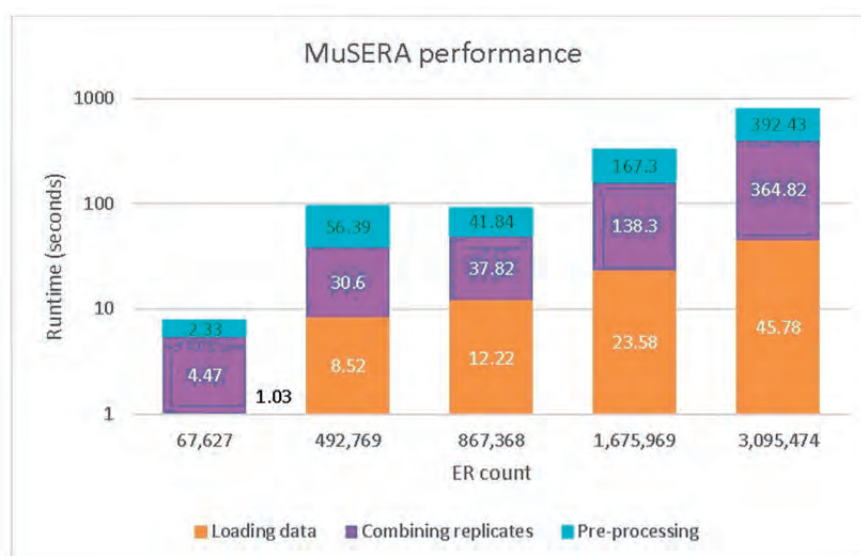


Figure 12. Benchmarking of MuSERA main operations. Operation runtime, on a logarithmic scale, for increasing number of ERs in combined replicates of ENCODE ChIP-seq or DNase-seq data sets (two to three replicates for each data set). The data sets considered were downloaded from ENCODE; for the ER counts in the figure they were, from left to right: wgEncodeSydhHistoneK562bH3k4me3bUcdAlnRep1/2, wgEncodeUwDnaseK562Znf4g7d3AlnRep1/2, wgEncodeUwDnaseK562Znf2c10c5AlnRep1/2, wgEncodeOpenChromDnaseK562NabutAlnRep1/2 and wgEncodeOpenChromDnaseK562G1phaseAlnRep1/2/3.

biological characteristics of the ERs. For example, for the TF CTCF we observed the lowest rate of rescue of weak peaks among all the replicates: this TF makes contact with the DNA through 11 distinct zinc-finger domains [37], and therefore binds in an extremely strong way. In this case, most of these interactions correspond to a clear signal in the ChIP-seq experiment, and the corresponding ERs are inevitably classified as stringent. Therefore, for this particular TF, the replicates are more coherent than usual, and most of the weak interactions are classified as noise. On the opposite, DNaseI hypersensitive sites have been shown to display a continuum of intensities, which does not saturate even at high sequencing depths [38]; therefore, for these experiments, the border between weak and stringent ERs is somewhat arbitrary and many of the ERs classified as weak in a single sample correspond to true open chromatin regions, consistently observed across replicates. In this case, MuSERA is particularly successful in expanding the set of the confirmed ERs.

The integrated genome browser and the several graphical features that MuSERA offers for genomic annotation and functional analysis of ERs, nearest ER distance distribution and global correlation assessment of ERs proved useful for the evaluation and biological interpretation of the obtained ERs within the genomic context, and could be the starting point of deeper functional analyses based on more refined measures, as currently implemented in other tools [9, 35, 36]. Moreover, the method that MuSERA implements to obtain the ERs was proved, with respect to other approaches, to optimally address the specific task of combining evidence over replicates [8]. Its output, designed to allow quick pipelining to downstream analyses, provides both sample-specific BED files of the different ER sets determined, and a single BED file unifying the significant confirmed ERs present in the combined replicate samples; all these files can be directly analysed with common tools like BEDTools [39], BEDOPS [40] or Bioconductor [41]. In addition, the overview XML files generated give all the details about the performed combination of multiple evidence across replicates, and allow tracking down the individual overlapping events among ERs. All

this makes MuSERA a tool likely to be of broad utility that represents a significant advance over previously published software.

Supplementary data

Supplementary data are available online at <http://bib.oxfordjournals.org/>.

Key Points

- Replicates in next-generation sequencing experiments are recommended, but their full potential, especially in experiments involving the identification of enriched regions (ER), is often neglected.
- MuSERA is a tool that allows combining local evidence in replicates to improve ER calling, and provides quantitative evaluations and graphical features to assess the biological relevance of each determined ER set within its genomic context; they include genomic annotation of determined ERs, nearest ER distance distribution and global correlation assessment of ERs.
- MuSERA comes with an intuitive graphical user interface, making it immediate to use, which provides an integrated genome browser and an array of graphical displays that greatly support understanding and biological interpretations of the results.
- By applying MuSERA to different data types, including ChIP-seq of transcription factors or histone marks and DNase-seq hypersensitive sites, we always found enhanced sets of ERs and proved its effective support in the inspection of obtained results and evaluation of their biological content.
- MuSERA represents a significant advance over previously published software, as we discuss and comparatively demonstrate.

Funding

This work was supported by the Fondazione Istituto Italiano di Tecnologia and by AIRC [IG_13182] and the Italian Ministry of the University and Research (MIUR) ['Data-Driven Genomic Computing (GenData 2020)' PRIN project (2013-2015)].

References

- van Dijk E, Auger H, Jaszczyszyn Y, et al. Ten years of next-generation sequencing technology. *Trends Genet* 2014;**30**(9):418–26.
- Park P. ChIP-seq: advantages and challenges of a maturing technology. *Nat Rev Gen* 2009;**10**:669–80.
- Cockerill P. Structure and function of active chromatin and Dnase I hypersensitive sites. *FEBS J* 2011;**278**:2182–210.
- Zhang Y, Liu T, Meyer CA, et al. Model-based analysis of ChIP-Seq (MACS). *Genome Biol* 2008;**9**(9):R137.
- Rashid NU, Giresi PG, Ibrahim JG, et al. ZINBA integrates local covariates with DNA-seq data to identify broad and narrow regions of enrichment, even within amplified genomic regions. *Genome Biol* 2011;**12**(7):R67.
- Chen Y, Negre N, Li Q, et al. Systematic evaluation of factors influencing ChIP-seq fidelity. *Nat Methods* 2012;**9**(6):609–14.
- Li Q, Brown JB, Huang H, et al. Measuring reproducibility of high-throughput experiments. *Ann Appl Stat* 2011;**5**(3):1752–79.
- Jalili V, Matteucci M, Masseroli M, et al. Using combined evidence from replicates to evaluate ChIP-seq peaks. *Bioinformatics* 2015;**31**(17):2761–9.
- Zeng X, Sanalkumar R, Bresnick EH, et al. jMOSAICS: joint analysis of multiple ChIP-seq datasets. *Genome Biol* 2013;**14**(4):R38.
- Fisher RA. *Statistical Methods for Research Workers*. Guildford, UK: Genesis Publications, Ltd, 1925.
- Bulger M, Groudine M. Enhancers: the abundance and function of regulatory sequences beyond promoters. *Dev Biol* 2010;**339**(2):250–7.
- Lettice LA, Heaney SJ, Purdie LA, et al. A long-range Shh enhancer regulates expression in the developing limb and fin and is associated with preaxial polydactyly. *Hum Mol Genet* 2003;**12**(14):1725–35.
- Glassford WJ, Rebeiz M. Assessing constraints on the path of regulatory sequence evolution. *Philos Trans R Soc Lond B Biol Sci* 2013;**368**(1632):20130026.
- MacQuarrie KL, Fong AP, Morse RH, et al. Genome-wide transcription factor binding: beyond direct target regulation. *Trends Genet* 2011;**27**(4):141–8.
- Pearson K. Note on regression and inheritance in the case of two parents. *Proc R Soc Lond* 1895;**58**(347–352):240–2.
- Bardet AF, He Q, Zeitlinger J, et al. A computational pipeline for comparative ChIP-seq analyses. *Nat Protoc* 2012;**7**(1):45–61.
- Zhao X, Valen E, Parker BJ, et al. Systematic clustering of transcription start site landscapes. *PLoS One* 2011;**6**(8):e23409.
- Rau A, Gallopin M, Celeux G, et al. Data-based filtering for replicated high-throughput transcriptome sequencing experiments. *Bioinformatics* 2013;**29**(17):2146–52.
- Giannopoulou EG, Elemento O. An integrated ChIP-seq analysis platform with customizable workflows. *BMC Bioinformatics* 2011;**12**(1):277.
- Ashoor H, Hérault A, Kamoun A, et al. HMCAN: a method for detecting chromatin modifications in cancer samples using ChIP-seq data. *Bioinformatics* 2013;**29**(23):2979–86.
- Smith J. WPF apps with the Model-View-ViewModel design pattern. *MSDN Magazine* 2009;**24**(2):dd419663.
- Visual Studio Code metrics values. 2015. <https://msdn.microsoft.com/en-us/library/bb385914.aspx> (30 January 2016, date last accessed).
- Hwang FK, Lin S. A simple algorithm for merging two disjoint linearly ordered sets. *SIAM J Comput* 1972;**31**:9.
- Cormen TH, Leiserson CE, Rivest RL, et al. Section 14.3: interval trees. In: *Introduction to algorithms*. 3rd edn. Cambridge, MA: MIT Press and McGraw-Hill, 2009, p. 348354.
- Bayer R. Symmetric binary B-trees: data structure and maintenance algorithms. *Acta Inform* 1972;**1**(4):290–306.
- Bentley JL. *Solutions to Klee's rectangle problems*. Pittsburgh, PA: Carnegie-Mellon University, 1977.
- CodePlex. Dynamic Data Display. 2011. <http://dynamicdata.display.codeplex.com/> (30 January 2016, date last accessed).
- Landt SG, Marinov GK, Kundaje A, et al. ChIP-seq guidelines and practices of the ENCODE and modENCODE consortia. *Genome Res* 2012;**22**(9):1813–31.
- Mathelier A, Zhao X, Zhang AW, et al. JASPAR 2014: an extensively expanded and updated open-access database of transcription factor binding profiles. *Nucleic Acids Res* 2014;**42**:D142–7.
- Bailey TL. DREME: motif discovery in transcription factor ChIP-seq data. *Bioinformatics* 2011;**27**(12):1653–9.
- Djebali S, Davis CA, Merkel A, et al. Landscape of transcription in human cells. *Nature* 2012;**489**(7414):101–8.
- Blankenberg D, Taylor J, Schenk I, et al. A framework for collaborative analysis of ENCODE data: making large-scale analyses biologist-friendly. *Genome Biol* 2007;**17**(6):960–4.
- Boekel J, Chilton JM, Cooke IR, et al. Multi-omic data analysis using Galaxy. *Nat Biotechnol* 2015;**33**:137–9.
- Boeva V, Lermine A, Barette C, et al. Nebula – a web server for advanced ChIP-seq data analysis. *Bioinformatics* 2012;**28**(19):2517–19.
- Favorov A, Mularoni L, Cope LM, et al. Exploring massive, genome scale datasets with the GenometriCorr package. *PLoS Comput Biol* 2012;**8**(5):e1002529.
- Bible PW, Kanno Y, Wei L, et al. PAPST, a user friendly and powerful Java platform for ChIP-seq peak co-localization analysis and beyond. *PLoS One* 2015;**10**(5):e0127285.
- Phillips JE, Corces VG. CTCF: master weaver of the genome. *Cell* 2009;**137**(7):1194–211.
- Neph S, Viestra J, Stergachis AB, et al. An expansive human regulatory lexicon encoded in transcription factor footprints. *Nature* 2012;**489**(7414):83–90.
- Quinlan AR, Hall IM. BEDTools: a flexible suite of utilities for comparing genomic features. *Bioinformatics* 2010;**26**(6):841–2.
- Neph S, Kuehn MS, Reynolds AP, et al. BEDOPS: high-performance genomic feature operations. *Bioinformatics* 2012;**28**(14):1919–20.
- Gentleman RC, Carey VJ, Bates DM, et al. Bioconductor: open software development for computational biology and bioinformatics. *Genome Biol* 2004;**5**:R80.

Indexing Next Generation Sequencing Data

Vahid Jalili^{a,*}, Matteo Matteucci^a, Marco Masseroli^a, Stefano Ceri^a

^a*Dipartimento di Elettronica, Informazione e Bioingegneria (DEIB)
Politecnico di Milano, Piazza Leonardo da Vinci, 32, Milan, Italy*

Abstract

Next-Generation Sequencing (NGS), also known as high-throughput sequencing, has opened the possibility of a comprehensive characterization of the genomic and epigenomic landscapes, giving answers to fundamental questions for biological and clinical research, e.g., how DNA-protein interactions and chromatin structure affect gene activity, how cancer develops, how much complex diseases such as diabetes or cancer depend on personal (epi)genomic traits, opening the road to personalized and precision medicine. In this context, our research has focused on *sense-making*, e.g., discovering how heterogeneous DNA regions concur to determine particular biological processes or phenotypes. Towards such discovery, characteristic operations to be performed on region data regard identifying co-occurrences of regions, from different biological tests and/or of distinct semantic types, possibly within a certain distance from each others and/or from DNA regions with known structural or functional properties. In this paper, we present Di3, a Durable Interval Inverted Index, acting as a multi-resolution single-dimension data structure for interval-based data queries. Di3 is defined at data access layer, independent from data layer, business logic layer, and presentation layer; this design makes Di3 adaptable to any underlying persistence technology based on key-value pairs, spanning from classical B⁺tree to LevelDB and Apache HBase, and makes Di3 suitable for different business logic and presentation layer scenarios. We demonstrate the effectiveness of Di3 as a general purpose genomic region manipulation tool, with a console-level interface, and as a software component used within MuSERA, a tool for comparative analysis of region data replicates from NGS ChIP-seq and DNase-seq tests.

*To whom correspondence should be addressed
Email address: vahid.jalili@polimi.it (Vahid Jalili)

Keywords:

Genomic computing; domain-specific data indexing; region-based operations and calculus; data integration.

1. Introduction

Next-Generation Sequencing (NGS) is a family of technologies for precisely, quickly and cheaply reading the DNA or RNA of biological samples (Shendure and Ji, 2008), (Schuster, 2008), producing huge amounts of data. Large-scale sequencing projects are spreading and very numerous genomic features, produced by processing NGS raw data, are collected by research centers, often organized through world-wide consortia, e.g., ENCODE (ENCODE_Project_Consortium et al., 2012), TCGA (Weinstein et al., 2013), 1000 Genomes Project (1000_Genomes_Project_Consortium et al., 2010), Epigenomics Roadmap (Romanoski et al., 2015), and others.

The availability of NGS data has opened the possibility of a comprehensive characterization of genomic and epigenomic landscapes. Answers to fundamental questions for biological and clinical research are hidden in these data, e.g., how DNA-protein interactions and chromatin structure affect gene activity, how cancer develops, how much complex diseases such as diabetes or cancer depend on personal (epi)genomic traits. Personalized and precision medicine based on genomic information is becoming a reality; the potential for data querying, analysis and sharing may be considered as the biggest and most compelling big data problem of mankind.

NGS technologies (Schuster, 2007), (Caporaso et al., 2012) allow collecting genome-wide genomic and epigenomic features, including DNA mutations or variations (DNA-seq), transcriptome profiles (RNA-seq), DNA methylations (BS-seq), DNA-protein interactions and chromatin characterizations (ChIP-seq and DNase-seq) (Park, 2009), (Cockerill, 2011). The processing of raw data (i.e., NGS reads) produced by these technologies returns lists of regions of cellular DNA, characterized by some common property; such regions, often referred as *peaks* (of NGS reads), are defined through their linear genomic coordinates and they are usually associated with several attribute values, including a statistical significance score, e.g., a p-value (Zhang et al., 2008) (Rashid et al., 2011).

The comparative analysis of heterogeneous genomic features produced by NGS technologies is named *tertiary analysis*, in contrast to *primary analysis*,

focused on the alignment of raw data (short reads) to reference genomes, and *secondary analysis*, focused on feature calling. Tertiary analysis is responsible of *sense-making*, e.g., discovering how heterogeneous regions synergically concur to determine particular biological processes or phenotypes.

Our research is targeting tertiary analysis; we recently proposed a new holistic approach to genomic data modeling and querying¹ that takes advantage of cloud-based computing to manage heterogeneous data produced by NGS technologies. In (Masseroli et al., 2015), we introduced the novel *GenoMetric Query Language* (GMQL), built on an abstract model for genomic data; we sketched out its main operations and demonstrated its usefulness, expressive power and flexibility through multiple different examples of biological interest (including finding ChIP-seq peaks in promoter regions, finding distal bindings in transcription regulatory regions, associating transcriptomics and epigenomics, and finding somatic mutations in exons).

We also developed methods for secondary data analysis, with a focus on data integration. Indeed, NGS experimental protocols recommend the production of at least two replicates for each sequenced sample, in order to remove false positive calls and “rescue” (i.e., call) regions with low significance score which would probably be discarded in a single sample evaluation, but that are supported by a sufficiently strong evidence when combined across multiple replicate samples. To perform such task, we recently proposed a novel method (Jalili et al., 2015), which has been then implemented in MuSERA (Jalili et al., 2016), an efficient tool for locally combining evidence on replicates and interactive graphical evaluations of results in the genomic context.

1.1. Our Contribution

Both GMQL and MuSERA are grounded on the efficient execution of operations for the composition and comparison of (epi)genomic regions, and their associated attributes. Region-based operations to be performed towards these goals include the identification of co-occurrences or accumulations of regions, possibly from different biological tests and/or of distinct semantic types, within the same area of the DNA, sometimes within a certain distance from each others and/or from DNA regions with known structural or functional properties (e.g., describing particular DNA sequence motifs,

¹http://www.bioinformatics.deib.polimi.it/genomic_computing/

genes involved in certain biochemical pathways, or regulatory regions of gene transcription activity). Nowadays, such complex operations are only partially supported by existing tools, e.g., BEDTools (Quinlan and Hall, 2010), BEDOPS (Neph et al., 2012), GROK (Ovaska et al., 2013); these tools typically support only algebraic operations based on the genomic coordinates of the regions within a single data sample or a pair of samples at the time, requiring the use of scripts to perform complex operations on multiple data samples.

To cope efficiently with complex region calculus, we have developed the Durable Interval Inverted Index (Di3), a multi-resolution single-dimension data structure. Di3 is defined at the data access layer, and it is independent from data layer, business logic layer, and presentation layer. This design decision has two significant advantages; firstly, being independent from the data layer, Di3 is adaptable to any key-value pair persistence technology. These may range from Apache Cassandra, LevelDB, Kyoto Cabinet, and Berkeley DB for persisted large scale data (NoSQL databases are surveyed in (Han et al., 2011) and (Tudorica and Bucur, 2011)), to simple in-memory key-value collections implemented by most of modern programming languages (e.g., “Dictionary” in C# and “map” in C++)². Secondly, Di3 design can support different business logic and presentation layer scenarios, which are complemented by user-defined functions (UDF) provided via behavioral design patterns such as *strategy pattern* (Vlissides et al., 1995).

In this paper, we describe two contexts of use of Di3:

- General-purpose, self-contained use of Di3 from the Di3B command-line interface (Di3BCLI) that provides console-level accessibility to Di3B operations (see Figure 1). This is also used for performance evaluation.
- Use of Di3 as a component within the MuSERA tool (Jalili et al., 2016), showing how Di3 adapts to the general requirements of secondary and tertiary data analysis (see Figure 1).

²In this manuscript, we use an implementation of Di3 which relies on classical B⁺tree and runs both in-memory and persistent.

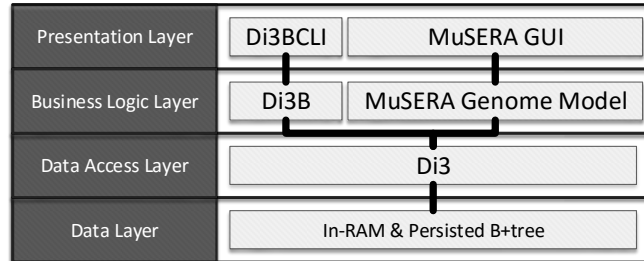


Figure 1: Two applications' design of Di3.

1.2. Outline

This paper is organized as follows. Section 2 presents the state of the art. Section 3 is dedicated to the Di3 method, further sub-structured in its model, data structure, and supported operations. Section 4 reports experiments which assess the effectiveness of Di3 with big data sets and compare to BEDOPS and BEDTools, which are currently used in the state of the art for region-based operations. In Section 5 we demonstrate how Di3 is used within MuSERA, a recently developed tool for multiple sample peak calling.

2. Related Work

Previous work for Di3 can be traced to classical search trees, such as interval trees (Cormen et al., 2009), segment trees (Bentley, 1977), range trees (Bentley, 1979), or Fenwick trees (Fenwick, 1994); these are optimal solutions, each for particular interval-based retrieval, and some are used in common bioinformatics tools as underlying data structure, e.g., UCSC Genome Browser and BEDTools use R-Trees (Guttman, 1984). However, individually such data structures do not provide a comprehensive solution for tertiary analysis challenges. For instance, retrieval queries such as ‘*find all the intervals (i.e., regions) intersecting with a given interval*’ can be determined in $O(\log_2 n)$ using interval trees; while queries such as ‘*find the n -th closest interval*’ require re-mastering the very same data structure, which becomes inefficient. Moreover, some of such data structures are mainly designed as in-memory data structures; and when persisted, they cause a severe overhead in terms of Input/Output operations. Additionally, these data structures are poor candidates for parallelism; for instance, R-trees partition intervals into

hierarchical *bins*, hence non-uniformly distributed intervals (which are common for ChIP-seq, RNA-seq, and exome sequencing) unbalance bin loads, and, consequently, some bins take considerably longer time to be processed than others.

Temporal databases and spatial data structures provide solutions for interval manipulation, from efficient storage and retrieval to manifold operations on top of that. Generally, in the temporal databases the challenge is formalized as object/tuple versioning, e.g., (Snodgrass and Ahn, 1985); in such methods, objects have different versions at various timespans, and can be valid or invalid at a given period, see also (Elmasri et al., 1990). Other works have shown temporal indexing schemes which handle the different notions of time, as well as non-temporal attributes, by using classical B⁺ trees (Goh et al., 1996) or by exploiting the built-in functionalities of relational database systems (Stantic et al., 2010). Such systems hold the concept of time; hence, they render *past (history)*, *now*, and *future*. Consequently, the storage technology is commonly 'append-only', and removing an object is generally defined as invalidating the object at a given timespan. Additionally, temporal operations commonly target recent events; hence, events prior to a certain time point are usually archived to reduce the amount of information to be processed for regular queries (which generally target recent events only).

A common prerequisite in temporal databases design is that the update, insertion, and deletion operations target *now* and history is intact (e.g., John's paid salary at last year does not change), however, few works exist concerning bi-temporal data where updating *past* is possible through additional dimension (Kaufmann et al., 2015). Compared to temporal databases, genomic databases have a fundamental difference: they do not hold any synonym for temporal model concerning *past*, *now*, and *future* concepts, as all events on the genomics domain are of equal importance from the location perspective; manipulation functions can target any position on the genome, and thus storage, indexing, and operations should be targeted at this aim.

Similarly to what we propose here, several works have proposed the embedding of region query processing functions within libraries that can be integrated within programs (Cereda et al., 2011), (Ovaska et al., 2013). In particular, GROK (Ovaska et al., 2013) presents a rather elegant mathematical formalism based on set algebra; its authors propose a genomic region abstraction (that may represents reads, genomic variants, mutations, and so on), and then define a set of region operations, delivered as the *Genomic Re-*

gion Operation Kit (GROK) library. In comparison, GROK supports lower-level abstractions than Di3 and some low-level operations (e.g., flipping regions) that Di3 does not directly support, but they must be embedded into C++ programming language code. Furthermore, high-level declarative operations, such as **COVER** and **MAP** (see Section 3), can be encoded in GROK, but they must be invoked from line editors or C++ programs.

BEDTools (Quinlan and Hall, 2010) and BEDOPS (Neph et al., 2012) are customarily used by biologists for processing region data in BED format; they can be used from within software environments for bioinformatics (e.g., *BioPerl*, *BioPython*, *R* and *Bioconductor*), and support algebraic operations based on the genomic coordinates of regions, but only within a single or a pair of data samples at the time, requiring the use of scripts to evaluate multiple data samples. A thorough comparison of Di3 with BEDTools and BEDOPS is presented in Section 4.

Alternative approaches exist to efficiently process intervals in the Apache Hadoop ecosystem (Buck et al., 2011), (Eldawy and Mokbel, 2013) by implementing algorithms which partition the operands in order to speed up their evaluation. In (Chawda et al., 2014) the authors propose an algorithm based on data binning; recently, (Afrati et al., 2015) further analyze binning-based algorithms in order to assess their computation bounds. For Apache Spark (Zaharia et al., 2012), similar problems have been addressed by projects such as GeoSpark (Yu et al., 2015) and (Sriharsha, 2015). Besides cloud-based systems, scientific databases can also be used to support genomic computing, including *Vertica*³ (used by the Broad Institute and NY Genome Center), and *SciDB*⁴ (further enhanced by *Paradigm4*⁵ a company whose products include genomics adds-on to SciDB and access to NGS data from TCGA and 1000 Genomes Project).

Several organizations are considering genomics at a global level. For instance, *Global Alliance for Genomics and Health* (GA4GH)⁶ is a large consortium of over 200 research institutions with the goal of supporting voluntary and secure sharing of genomic and clinical data; their work on data interoperability is producing a conversion technology for the sharing of data on DNA sequences and genomic variations (GA4GH_Data_Working_Group,

³<https://www.vertica.com/>

⁴<http://www.scidb.org/>

⁵<http://www.paradigm4.com/>

⁶<http://genomicsandhealth.org/>

2015). Also *Google* recently provided an API to store, process, explore, and share DNA sequence reads, alignments and variant calls, using Google’s cloud infrastructure⁷.

3. Di3 Design

Di3 is a general-purpose index structure which provides fast access to intervals; therefore, it applies to several domains, including genomics (any genomic region is a linear interval defined by the genomic coordinates of the region ends). Di3 main strengths are its ability to adapt to domain needs, thanks to the native support for user-defined functions (UDFs), and its portability to several implementation technologies, thanks to its high-level, layered design that abstracts from implementation details. Di3 models homogeneous and heterogeneous intervals on a domain; related events are collected in sets, which collectively constitute a *sample*. For instance, a genomic data sample may contain regions (i.e., intervals) of DNA-protein interactions occurring on a genome under an experimental setup; each interval can be associated with values, e.g., a significance score.

In general, let $\mathbb{S} = \{S_1, \dots, S_j, \dots, S_J\}$ denote the available samples, where each sample is a set of intervals $S_j = \{I_1^j, \dots, I_i^j, \dots, I_{|S_j|}^j\}$; each interval $I = [\underline{I}, \bar{I})$ is included within its lower (left) and upper (right) bounds (ends). Di3 organizes intervals by means of *snapshots* (see Figure 2); each snapshot corresponds to a point on the domain, and is associated with all the intervals overlapping that point. More precisely, each snapshot B_b is a key-value pair, where the key (e_b) is the coordinate of the snapshot on the domain, and the value (λ_b) is a set of pointers to descriptive metadata of all the intervals overlapping the coordinate e_b .

3.1. Di3 Operations

The coordinate-oriented model of Di3 facilitates region calculus. Data retrieval is defined in three levels: *Physical*, *Logical*, and *Semantic*. The former bridges the Di3 data model to the data layer, using some key-value pair persistence technology. Operations at the physical layer include *Create*, *Read*, *Update*, *Deleted* (CRUD), and *Enumerate*. These operations create and manipulate the snapshots and organize them in a key-value pair storage, by translating input intervals to snapshots and retrieving intervals from

⁷<https://cloud.google.com/genomics/>

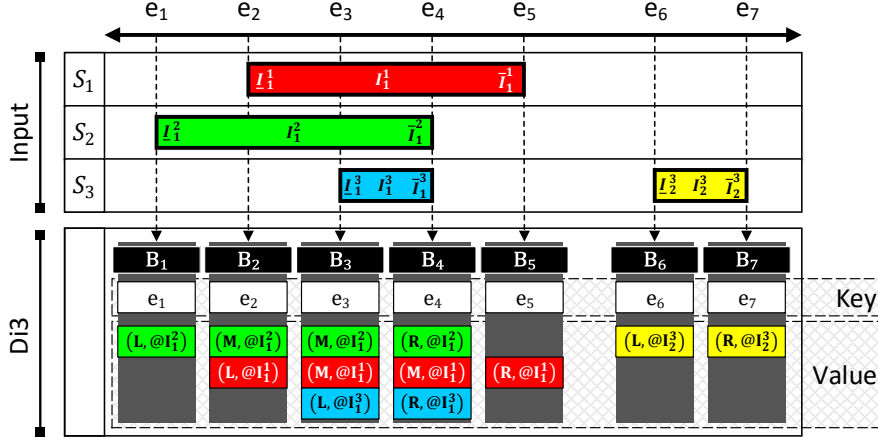


Figure 2: Di3 data structure. S_j : sample j ; I_i : interval i ; B_b : snapshot b .

snapshots. They are internal to Di3 and accordingly do not incorporate UDFs.

Logical level functions leverage on physical level operations, and provide the essential elements for region calculus. These functions cover classical region calculus that benefits from the information of a single snapshot, e.g., 'given a point on the domain, find intervals overlapping with it' (similar to queries on segment trees), or that leverage on information provided by a set of consecutive snapshots, e.g., 'given an interval, find all intervals overlapping with it' (similar to queries on interval trees). Logical level functions leverage on snapshots to optimally retrieve co-occurrences of intervals, or co-occurrence histograms and distributions; some of them define the Di3 public application programming interface (API), whereas other functions can be user-defined.

Upon physical level operations and logical level functions, Di3 builds semantic level functions. The goal of these functions is to facilitate both high-level reasoning on data that include coordinate-attribute criteria, and UDFs creation for extensibility to application requirements. These functions are: *similarity search*, which finds samples that best match the criteria defined in a query; *co-occurrence patterns*, which searches for density-based co-occurrence patterns; *dependency detection*, which determines the positions on the domain where query regions co-occur; and *deviation detection*,

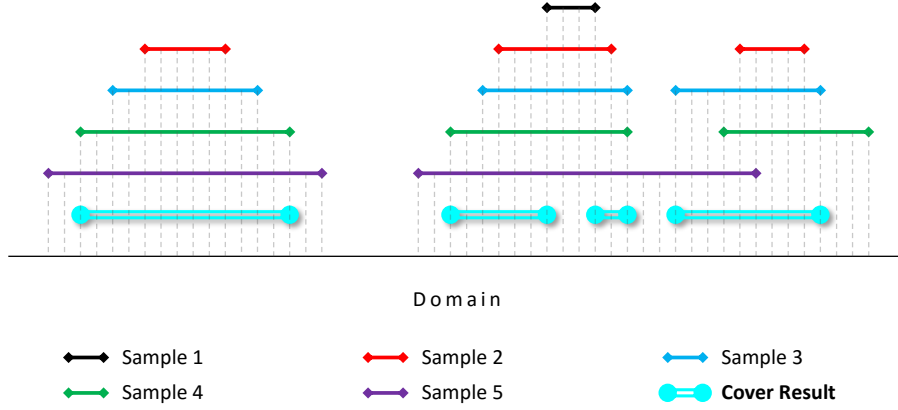


Figure 3: An example of `COVER` function with `minAcc=2` and `maxAcc=4` on 5 samples.

which finds positions on the domain where the regions of a given set do not commonly co-occur, based on the information stored in the Di3 model.

In this paper we focus on the logical layer, being this the *de-facto* Di3 API, whereas the physical layer operations are strictly related to the specific persistence technology used for the Di3 implementation, and the semantic layer functions are application dependent (some of them are used by MuSERA and are described in Section 5). Concerning the logical layer, we have the following operations natively supported by Di3:

Cover. The `COVER` function applies to snapshots and computes a single sample from intervals constituting the snapshots, by taking into account interval intersections and a UDF. Each resulting interval I is the contiguous intersection of at least `minAcc` (minimum accumulation) and at most `maxAcc` (maximum accumulation) of intervals. See Figure 3 as an example of the `COVER` function. For each resulting interval I , the `COVER` function determines contributing intervals, which then are passed to the UDF for further evaluation. A UDF may assign to a resulting region I any user-defined value type, spanning from 'list of all contributing intervals' or 'cardinality of contributing intervals', to analytical evaluations such as 'custom aggregation of significance values'. In `COVER`, the use of UDFs is thus supported, being `count` the default aggregation function.

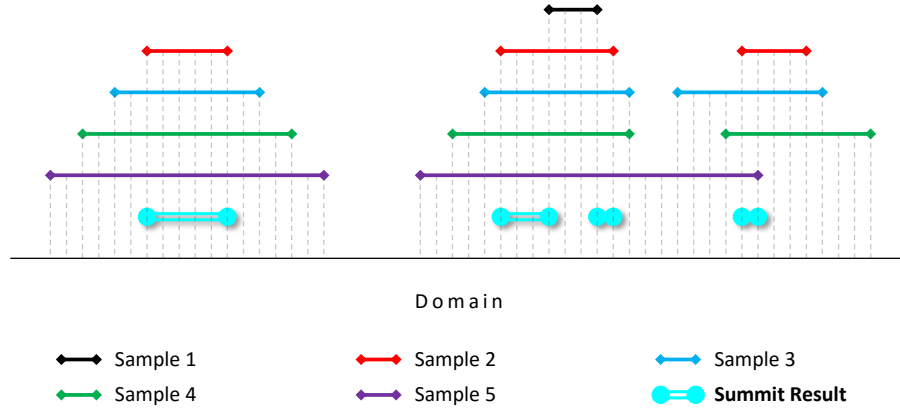


Figure 4: An example of SUMMIT function with `minAcc=2` and `maxAcc=4` on 5 samples.

Summit. The **SUMMIT** function is a variation of the **COVER** function; similarly it takes `minAcc`, `maxAcc` and a UDF, and reports the local intersections of **COVER**. In other words, **SUMMIT** returns regions that start from a position where the number of intersecting regions is between `minAcc` and `maxAcc` and does not increase afterwards, and stop at a position where either the number of intersecting regions decreases, or it violates the `maxAcc` parameter (see Figure 4). Similar to **COVER**, **SUMMIT** determines contributing intervals and pass them to the UDF, which can aggregate any property of intervals and return any user-defined aggregated value type.

Map. Given a reference interval I , the **MAP** function determines all the intervals indexed in Di3 that overlap with I (similar to classical interval-tree operation). In addition to a reference, the function takes a UDF and passes the determined intervals to the UDF (see Figure 5). The output of the UDF is then reported back as an attribute of I . For instance, a UDF may take all the intervals overlapping I and return their cardinality.

Accumulation histogram/distribution. The functions **Accumulation histogram** and **Accumulation distribution** compute the genome-wide accumulation of intervals, and respectively report a histogram or distribution of the calculated information (see Figure 6).

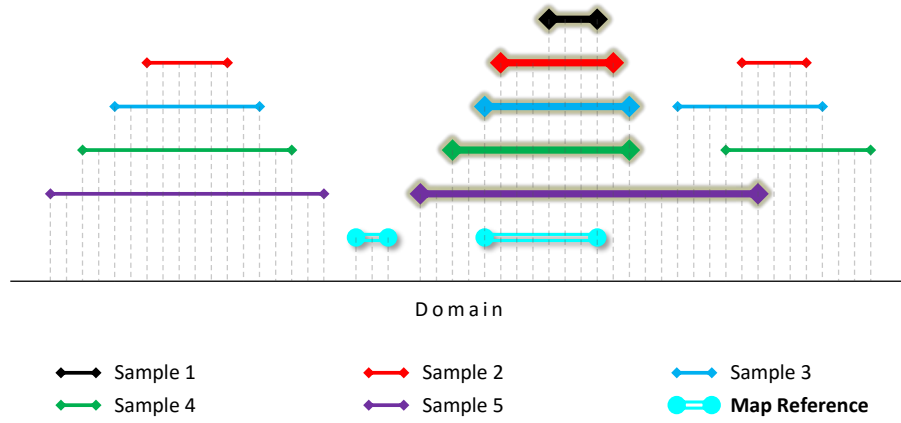


Figure 5: An example of MAP function on 5 samples.

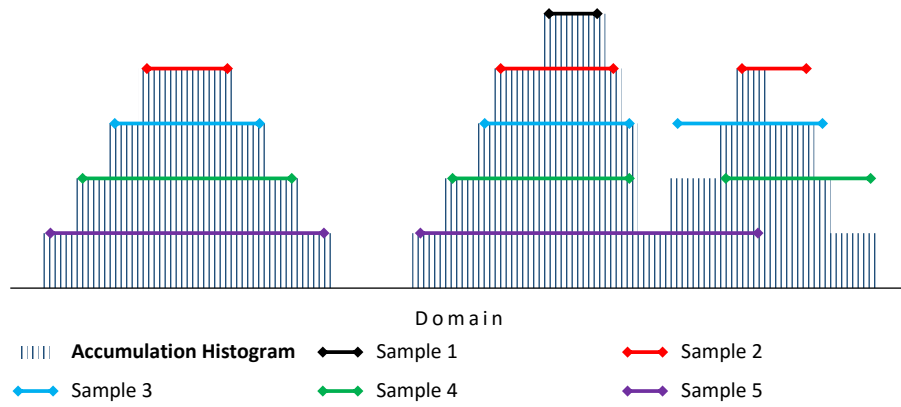


Figure 6: An example of *Accumulation Histogram* function on 5 samples.

Nearest neighbor. Given a reference interval I , the **Nearest neighbor** function determines the nearest neighbor indexed interval, that is either an overlapping interval, or the closest up-stream or down-stream non-overlapping interval.

3.2. Di3 Internals

The design of Di3 is motivated by giving the best possible support to logic operations; in particular, Di3 supports **MAP** and **Nearest neighbour** operations in *logarithmic time* in the number of regions, while it supports **COVER**, **SUMMIT** and **Accumulation histogram** operations in *linear time* (indeed, second resolution indexing supports sub-linear time search, as it will be later explained). These aspects motivate the superior performance of Di3 over the state-of-the-art, as discussed in Section 4.

Di3 adopts a *single-dimension* paradigm, and targets the interval's coordinates; however, it also adopts a *double-resolution* paradigm. The *first resolution* organizes intervals based on their coordinates through snapshots and provides logarithmic access on the coordinates; the *second resolution* builds groups of snapshots based on their coordinates and computes suitable aggregation functions for each group, using all the attributes (and not just the coordinates). These groups act as secondary key for improved access based on specific attributes. In the following sections we describe each resolution in details.

3.2.1. First Resolution

Snapshots summarize information about the coordinates where a *variation* occurs in intervals, i.e., when an interval starts or ends; a snapshot at e_b exists *iff* at least one interval introduced such variation. Hence, a finite set of snapshots can be used to index a finite set of intervals on both discrete and contiguous domains. Note that multiple intervals with the same starts or ends are possible.

Each snapshot holds a list of the IDs of all the intervals overlapping with its position; having a pointer to each interval overlapping a point on the domain is advantageous mainly for queries that target specific or relative positions. For instance, 'given an interval, find all intervals overlapping with it' requires $O(\log_2 n)$ to find the snapshot which has a pointer to all intersecting intervals; or 'given an interval, find all its neighbors at 200 base-pair (unit of the genomic domain) distance' requires $O(\log_2 n)$ to find the snapshot at

overlapping position, plus few additional siblings of the determined snapshot, and then to union the intervals represented by the snapshots (which asymptotically is still $O(\log_2 n)$)⁸.

This organization has been shown already in Figure 2; the upper part of the figure describes 4 input intervals; the lower part of the figure describes the Di3 index, where each snapshot B_b includes a key-value pair, the key is e_i , and the value is the list of intervals which have an intersection with B_b . Precisely, B_b can be at the left (L), at the right (R), or in the middle (M) of an interval; the type of positioning (i.e., L, R or M) is included in each entry of the interval list, which also contains a pointer to descriptive metadata for the interval. For instance, the third snapshot, B_3 , intersects with three intervals, being in the middle of the S_1 and S_2 intervals and at the left of the S_3 interval.

The first resolution index is created using *batch indexing*, a method which includes input intervals one after the other; such method is preferred over *bulk indexing* or *range indexing*, which require the sorting of intervals prior to index creation. These methods are related to the physical layer being implemented through a B⁺tree and they are related to classical methods for B⁺tree data insertion (Graefe, 2003), (Ghanem et al., 2004). To perform batch indexing, we contrasted two methods, respectively called *single pass indexing* and *double pass indexing*. The former one considers each interval in input and precisely updates the data structure in a single pass; the latter one at the first pass orders the snapshots of new intervals with respect to all existing intervals, and at the second pass updates all the snapshots with the information about the list of the intersecting intervals.

The two methods are compared in Figure 7, the comparison clearly shows that no method dominates over the other one. In general, single pass indexing is superior with a small number of new intervals, while double-pass indexing is superior for a large number of new intervals. Based on such analysis, the initial loading of the index in Di3 is performed by using the double-pass indexing, while index update is performed by single pass indexing.

⁸Computational complexities are based on a B⁺tree data structure for the implementation of the physical layer operations.

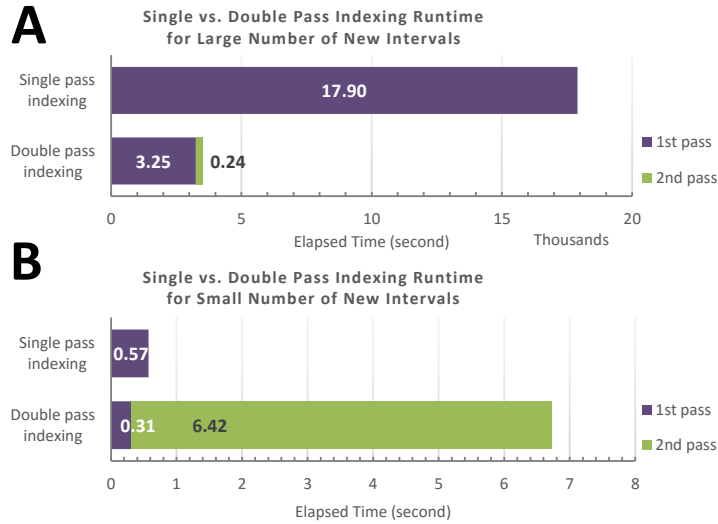


Figure 7: A comparison between single-pass and double-pass indexing in two scenarios: (A) large number vs. (B) small number of new intervals.

3.2.2. Second Resolution

The second resolution is built from the first resolution as of Figure 7, by grouping a set of snapshots and aggregating the relative information. The elements of the second resolution of Di3 are collections of consecutive snapshots grouped together, called *blocks*. Each block is defined as a key-value pair; the *key* is defined by a *grouping* function, while the *value* is defined by an *aggregation* function. More precisely:

In *Grouping*, a user-defined function groups a set of adjacent snapshots; depending on the semantic of the UDF, such groups reveal various aspects. For instance, in the genomic domain with intervals representing a biological activity, a group of snapshots is a region on a genome where at least one biological activity is observed. The grouping function defines the block *key*, as an interval on first resolution with start and stop being respectively the minimum and maximum coordinates of the snapshots in the block.

With *Aggregation*, the information of the snapshots within each block are then aggregated, thereby providing summary statistics on the specific attribute(s). Storing custom aggregations for each block reduces snapshot

access demands when a particular aggregation is commonly used. The aggregation function defines the block *value*, which is generic in type and size. For instance, 'count' of intervals represented by grouped snapshots, or standard deviation and median of interval sizes.

Di3 already provides built-in functions for default grouping and aggregation; the default grouping function groups consecutive snapshots that point to consecutive overlapping intervals, and the built-in aggregation function stores the maximum accumulation of each group. These functions improve operations such as **COVER**, **SUMMIT** and **Accumulation histogram**, discussed in Section 3.1, which depend on accumulation indexes. Indeed in a **COVER** query, when the max accumulation of a block is lower than the min accumulation **accMin** of the query, or the min accumulation of a block is higher than the max accumulation **accMax** of the query, the entire group of intervals does not have an interval that satisfies the query and can be fully skipped. Moreover, disjoint groups can be processed in parallel with no need for synchronization mechanisms.

4. Experiments

We start with an evaluation of Di3 enacted from a user interface, and then we present a comparison with BEDTools and BEDOPS, the most popular tools for genomic region calculus.

4.1. Di3 Evaluation

We customized Di3 to the genomic domain by building a Di3B index at the business logic layer that initializes several independent Di3 instances, one for each DNA chromosome and strand; the Di3BCLI client at presentation layer (see Figure 1) provides user interaction through a set of commands. These include: primitives for initializing the indexes, primitives for the operations of Section 3.1, and primitives for setting indexing modes and parallelism.

In general, the Di3BCLI commands (listed in Table 1) have standard command argument structure, where the number of arguments varies between different commands. Having executed a command, its runtime is reported on console, and also saved in a user-defined log file. The *Index* and *BatchIndex* primitives take a sample or a collection of samples as argument and, based on the indexing mode, index the intervals respectively in single-pass or double-pass mode. Under double-pass indexing mode, the command *2Pass*

Table 1: Di3BCLI commands.

Command	Description
<i>Index</i>	Takes a filename and indexes all its regions.
<i>BatchIndex</i>	Takes a set of files specified using wild-card characters.
<i>2Pass</i>	Runs second-pass of indexing in double-pass indexing mode.
<i>2RI</i>	Indexes second resolution.
<i>Cover</i>	Executes COVER function and exports results.
<i>Summit</i>	Executes SUMMIT function and exports results.
<i>Map</i>	Executes MAP function and exports results.
<i>Merge</i>	Executes MERGE function and exports results.
<i>Complement</i>	Executes COMPLEMENT function and exports results.
<i>AccHis</i>	Determines <i>Accumulation histogram</i> and exports results.
<i>AccDis</i>	Determines <i>Accumulation distribution</i> and exports results.
<i>GetIM</i>	Reports current setting for indexing mode.
<i>SetIM</i>	Sets indexing mode to the specified one.
<i>GetDP</i>	Reports current setting for degree of parallelization.
<i>SetDP</i>	Sets degree of parallelization to the specified one.

(which takes no arguments) executes the second-pass of the indexing. The second resolution of Di3 is created/updated by the *2RI* command (that takes no arguments). The *Cover* and *Summit* commands take *minAcc*, *maxAcc*, *aggregate*, and *output* arguments, execute the functions with the parameters and export results to the output file. The *Map* command takes *reference*, *aggregate*, and *output* arguments, executes the function and exports results in the output file. The *Merge*, *Complement*, *AccHis*, and *AccDis* commands take *output* argument, execute the function and report results to the output file. The *GetIM* reports current setting for indexing mode, and *SetIM* takes a *mode* argument which is either *emphsingle* or *emphmulti*, and sets indexing mode accordingly. Finally, *GetDP* reports current setting for degree of parallelization, and *SetDP* takes two numbers as *chr-degree* and *Di3-degree* of parallelization and updates the execution environment accordingly.

We benchmarked Di3 by exploiting two levels of parallelism: chromosome level parallelism (i.e., executing operations on multiple chromosomes concurrently) and binning within chromosomes (i.e., each chromosome is further divided into multiple sections or *bins*, and multiple threads process the resulting sections). In all the experiments of this section, we used an Amazon

Table 2: Datasets used for Di3 benchmarking.

Label	Sample count	Region count	Dataset size (GB)
<i>A1</i>	500	28,392,674	1.35
<i>A2</i>	1,000	59,980,303	3.17
<i>A3</i>	1,500	94,997,460	4.87
<i>A4</i>	2,000	143,563,549	6.98

EC2 machine running Microsoft Windows Server 2012 with an Intel® Xeon® E5-2670 v2 CPU, 320 GFLOPS, and 122 GB RAM. We used four datasets of ENCODE narrow peak samples, as described in Table 2; the datasets vary in size, but are similar in interval accumulation distribution. Figure 8 shows the accumulation distribution of datasets A2 and A4; statistics for the other two datasets follow similar distributions.

4.1.1. Benchmark of MAP

First, we assessed the performance of **MAP**, an operation which directly operates upon coordinates. We considered as reference a sample from the ENCODE repository, which includes 196,180 regions (8.9 MB in size); the **MAP** was performed over the four datasets in Table 2. Figure 9 shows excellent scalability with respect to growth in data size. Note that the reference sample is also an ENCODE narrow peak sample; hence, its intervals are mostly co-localized with the indexed intervals. Therefore, a very big percentage of indexed data overlaps with the reference intervals.

4.1.2. Benchmark for Accumulation Operations

Second, we assessed Di3 performance for **COVER**, **SUMMIT**, **Accumulation histogram** and **Accumulation distribution** operations, which use accumulation indexes as parameters. The function **SUMMIT** is a variation of **COVER** and, similarly, the function **Accumulation distribution** is a variation of **Accumulation histogram**; therefore, their performance is at the same scale (see Figure 10). Both the **Accumulation histogram** and **distribution** functions scan all snapshots; hence, their performance can be evaluated as the maximum time required for a full scan of Di3, which is linear to dataset size. Likewise, the functions **COVER** and **SUMMIT** require linear scan of snapshots for regions of specific accumulation of intervals. However, the second resolution index prunes a percentage of linear scan based on **minAcc** and **maxAcc** parameters, and on their overlap with the accumulation distribution of data.

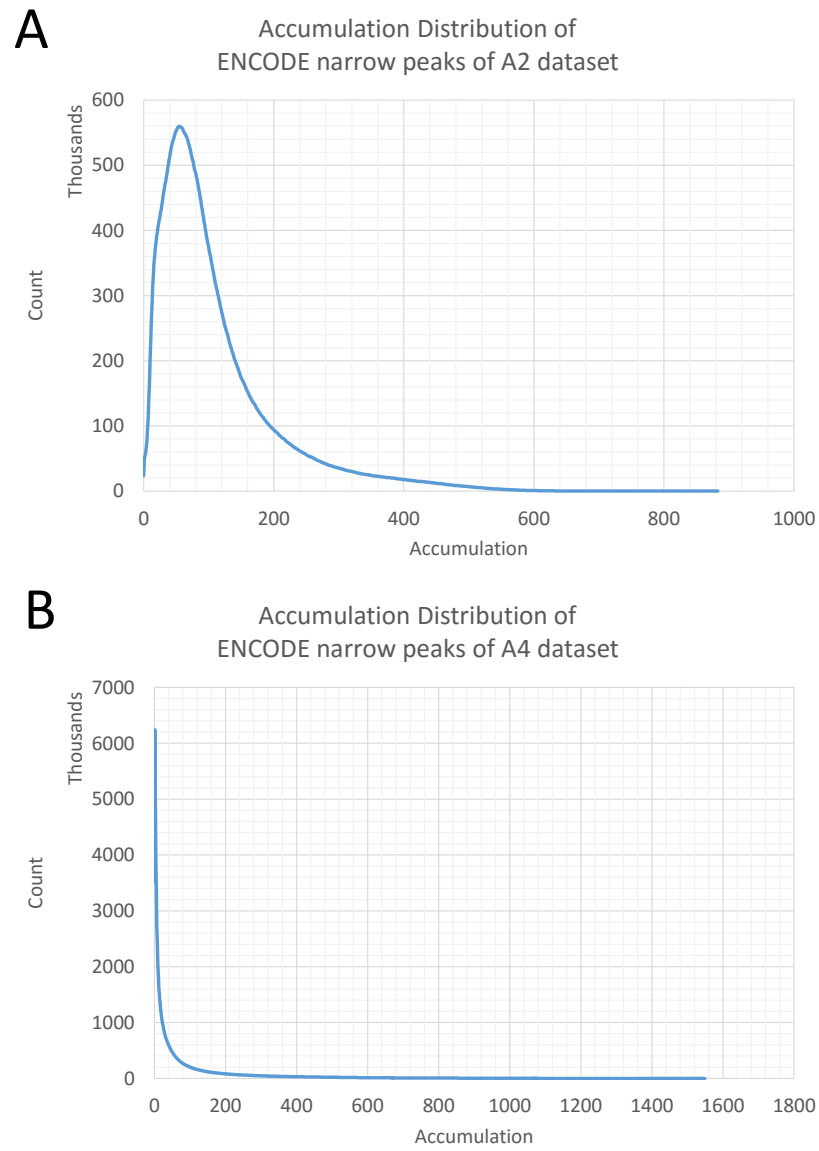


Figure 8: Interval/region accumulation distribution in datasets A2 (Panel A) and A4 (Panel B).

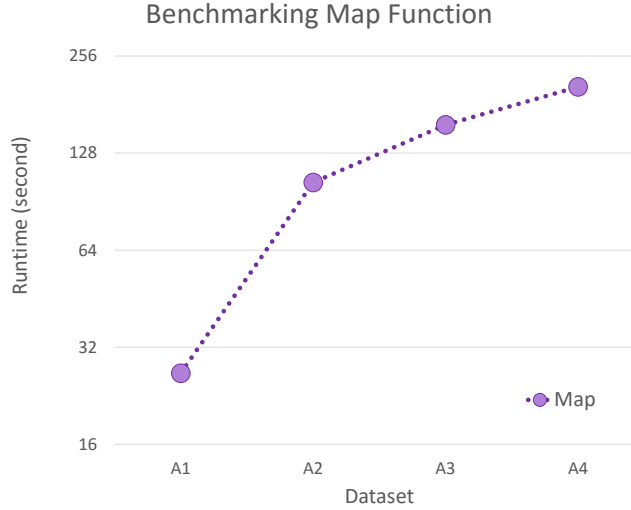


Figure 9: Benchmarking MAP function of Di3.

The less effective pruning is expected with parameters set around the peak of accumulation distribution, therefore the **COVER** and **SUMMIT** functions are expected to be faster than the **Accumulation histogram** and **Accumulation distribution** with such choice of parameters. We executed **COVER** and **SUMMIT** functions with **minAcc** and **maxAcc** at the peak of accumulation distribution (e.g., **minAcc**=80 and **maxAcc**=100 for A2); Figure 10 confirms that even with the peak of the parameter values the functions perform faster than full scan.

4.1.3. Effect of Accumulation Distribution

Data distribution may have a strong effect on the performance of an index. For instance, non-uniformly distributed data may accumulate a big load of information on some keys, while other keys may have lighter load; this is suboptimal because some keys are very expensive to process, while others are cheap. This affects also parallel execution, as some threads are busy for a very long time, while others are set free very early. The first step to avoid such draw backs is at design level, by making correct decisions, based on the nature of the data, for the key and value of the index.

In this section, we evaluate Di3 design and performance for data with different accumulation distribution. For this we simulated 10 datasets, each

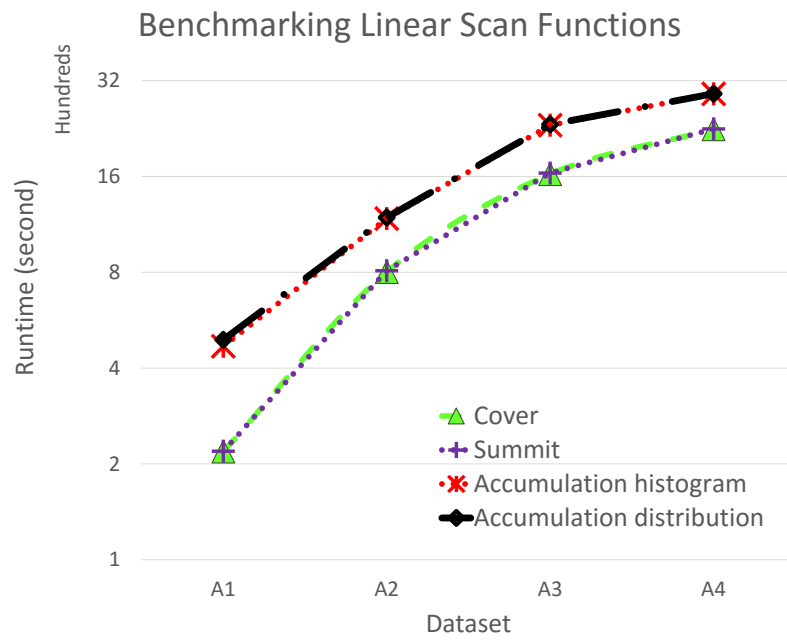


Figure 10: Benchmarking linear scan functions of Di3.

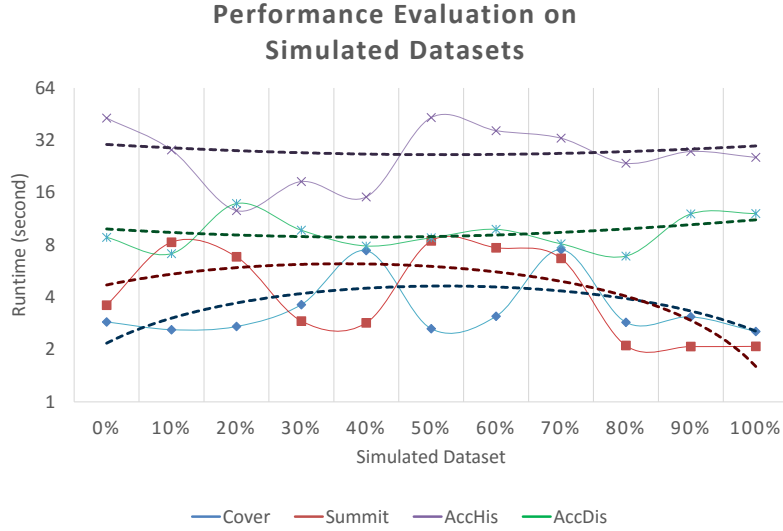


Figure 11: Function performance on simulated data.

containing 500 samples and 200,000 regions in each sample. Datasets differ in the percentage of intersecting intervals (in the extreme case, all samples contribute to intersection; therefore, the accumulation of the intersection region equals to the number of samples). A dataset with no intersection is labeled '0%', while a dataset with all intervals intersecting is labeled '100%'.

We executed Di3 functions on all simulated datasets. Figure 11 shows the results. Minor variations in function runtime throughout the datasets are due to the randomly generated positions and overlapping on intervals in the datasets. The figure highlights that Di3's performance is independent from the accumulation distribution of the input datasets.

4.2. Comparison with State of the Art Tools

In this section we compare the performance of Di3 with the one of two tools commonly used in genomic region processing, namely, BEDTools (Quinlan and Hall, 2010) and BEDOPS (Neph et al., 2012). We ran the tools on a standard laptop (Intel® Core™ i3 2.10 GHz and 8 GB RAM). We ran BEDTools and BEDOPS under Linux, and Di3 under Microsoft Windows® 10 operating system. We prepared Python and shell scripts for the batch execution of BEDTools and BEDOPS, and executed Di3 in-memory. Among

Table 3: Datasets used for Di3 benchmarking versus BEDTools and BEDOPS.

Label	Sample count	Region count	Dataset size (MB)
<i>B1</i>	90	1,407,493	97.4
<i>B2</i>	180	4,649,767	322.0

the possible operations that are available from the Di3B Command Line Interface, BEDTools and BEDOPS implement the **MAP** operator, i.e., given a reference sample, find input intervals overlapping with the reference regions. Therefore, we compared the **MAP** operator of Di3 against the '*bedtools map*' from BEDTools and the '*bedmap*' from BEDOPS. We considered two typical *usage scenarios* in genomic region processing: (i) personal repository, and (ii) on-the-fly processing.

4.2.1. Personal Repository

This is a common scenario for bioinformaticians, where a personal repository of in-house data, as output of the execution of a NGS data processing pipeline, and/or data obtained from publicly available repositories is persistent on their machine. Such data are stored for further processing or to be used for comparative evaluation and cross-referencing. The repository is a collection of properly organized files, indexed and persistent in Di3. We used two datasets of ENCODE narrow peak samples as in Table 3, and a reference sample including 196,180 regions (8.9 MB in size).

The data were pre-processed, i.e., filtered, and regions in samples were sorted for BEDTools and BEDOPS, and indexed for Di3. Hence, benchmark started from pre-processed data, and for comparison considered only execution time. As Figure 12 shows, Di3 performs significantly faster than BEDTools and BEDOPS.

4.2.2. On-the-Fly Processing

Processing data on-the-fly is a bioinformaticians' daily-based scenario, where a relatively small dataset is obtained from the execution of a NGS data processing pipeline and may not be archived for further evaluation. We benchmarked the in-memory version of Di3 versus BEDTools and BEDOPS on **MAP** operation using one reference, including 196,180 regions (8.9 MB in size), and three datasets of ENCODE narrow peak samples, described as in Table 4, as target.

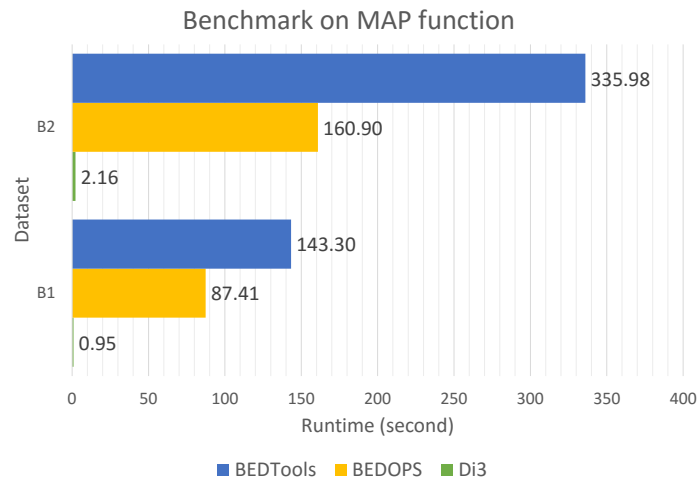


Figure 12: Benchmark: Personal repository scenario

Table 4: Datasets used for Di3 benchmarking versus BEDTools and BEDOPS for on-the-fly processing.

Label	Sample count	Region count	Dataset size (MB)
<i>C1</i>	12	89,623	6.17
<i>C2</i>	22	258,406	17.8
<i>C3</i>	45	456,385	31.5

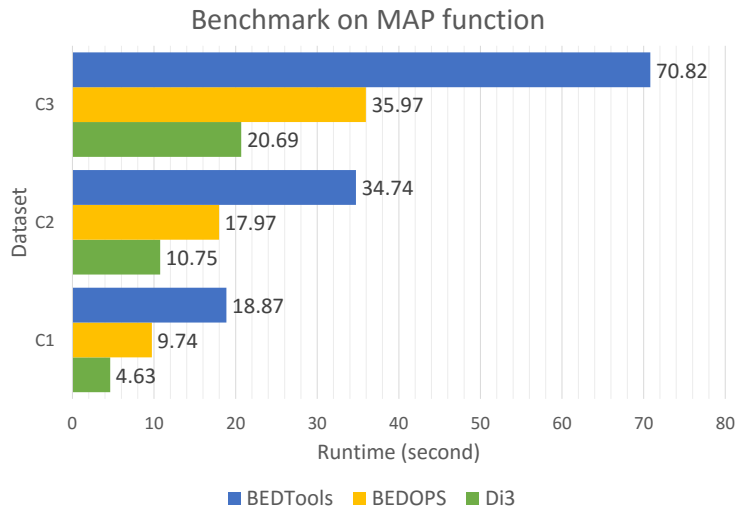


Figure 13: Benchmark: On-the-fly processing scenario

The data were not pre-processed: data were not sorted for BEDTools and BEDOPS, and not indexed for Di3. Hence, the execution time incorporated pre-processing time in all cases. Figure 13 shows that Di3 performs faster than BEDTools and BEDOPS on the three datasets. This highlights that Di3 is also an agile back-end data structure for on-the-fly processing, even by incorporating the indexing time within the processing time.

5. Use of Di3 as a Component within MuSERA

The analysis of NGS ChIP-seq samples outputs a number of enriched regions (ERs) for NGS reads, each indicating a protein-DNA interaction or a specific chromatin modification. ERs (or 'peaks') are called when the enrichment p-value is below a user-defined threshold. ERs with a p-value close to that of the background signal are either a background signal which is slightly enriched due to some biological or technical bias in the experiment, or indeed are biologically important regions with an enrichment less significant than expected. The NGS protocol is subject to noise; to avoid a large number of false positive ERs, commonly used thresholds are often very stringent, yielding many false negatives. However, the guidelines of the ENCODE project recommend repeating a NGS experiment at least twice (Landt et al., 2012) on

replicate samples, where the information contained in replicates is expected to be largely overlapping. We recently proposed a novel method (Jalili et al., 2015) that, by leveraging on available replicates, differentiates between sub-thresholded ERs and truly non-ERs, using a rigorous combination of the significance of individual ERs that overlap in replicate samples.

The results of the method should be further assessed using common procedures, such as visualization on a genome browser, functional analysis, or nearest-neighbor search. To this end, we recently proposed Multiple Sample Enriched Region Assessment (MuSERA) (Jalili et al., 2016), a novel graphical tool that leverages on Di3 (On-the-Fly scenario) to implement (i) comparative analysis of replicate samples using the method originally proposed in (Jalili et al., 2015), (ii) ER functional analysis, (iii) nearest neighbor search, (iv) correlation assessment, and (v) an integrated user-friendly genome browser.

MuSERA builds the business logic layer and presentation layer above Di3 (recall Figure 1). It encapsulates peaks as intervals for Di3, and provides particular comparative analysis driven operations (e.g., methods for combining p-values, and classification of peaks as true sub-thresholded ERs or artifacts) via UDF. Additionally, it uses an in-memory implementation of B⁺tree to implement the physical layer (Figure 1).

5.1. Comparative analysis

MuSERA combines statistical evidence of an ER (i.e., its p-value) with the co-localized evidence from replicates (it uses only the strongest evidence from each replicate if more than one evidence - i.e., ER - from a replicate is co-localized), and confirms/discards ERs based on the comparison between their combined evidence and a user-defined threshold. For this functionality, MuSERA benefits from Di3 MAP function with a UDF for comparative analysis, explained in (Jalili et al., 2015). The procedure is as follows:

- i. MuSERA provides graphical means to select and load replicates.
- ii. MuSERA indexes replicates in Di3 on-the-fly.
- iii. Through a graphical user interface (GUI), MuSERA gets user-defined parameters for a UDF for the comparative analysis.
- iv. MuSERA executes Di3 MAP with reference sample (the replicate against which ERs are to comparatively evaluated) and a UDF that:
 - (a) In each replicate sample, chooses the interval with the lowest p-value from the set of intervals overlapping with a reference interval.

- (b) If the number of overlapping intervals satisfies a user-defined minimum threshold, then combines the p-values of the overlapping intervals and the reference interval using the Fisher's method (Fisher, 1925).
- (c) If the combined stringency satisfies a user-defined threshold, then stores the intervals as confirmed, or as discarded otherwise.

5.2. Genome browser

MuSERA plots a given ER and all the neighbor ERs from replicates, as well as user-uploaded known genomic features (e.g., genes or promoter regions) at a given distance (e.g., within 100k base-pair). This function leverages again on the **MAP** (i.e., 'given an interval, find intervals overlapping with it') and the **Nearest neighbor** search functions of Di3. The determined intervals are then plotted using a dynamic plot that allows zoom in and out, and pan, which together with optimal retrieval from Di3 and plot features provides a high-end genome browser (see Figure 14 Panel A).

5.3. Functional analysis

Assigning an ER to the closest up-/down-stream genomic feature (e.g., gene Transcription Start Site (TSS), or Coding Sequence (CDS)) is a very useful and common task to help the biological interpretation of NGS results. MuSERA implements it optimally using Di3; it leverages on relative ordering of intervals accessible via a consecutive set of snapshots, and on the **Nearest neighbor** search, to calculate the *ER-to-feature conservation rate* (which is determine as the number of ERs intersecting with genomic annotations) and the *ER-to-feature distance distribution* between ERs and the closest up-/down-stream features (see Figure 14 Panel B).

5.4. Nearest neighbor

MuSERA computes the ER nearest neighbour distance distribution with functional annotation, leveraging on **Nearest neighbor** search function of Di3. The distribution facilitates biological assessment of ERs; for instance, it helps evaluating that comparatively confirmed weak ERs are relatively closer to functional annotations than comparatively evaluated true non-ERs (see Figure 14 Panel C).

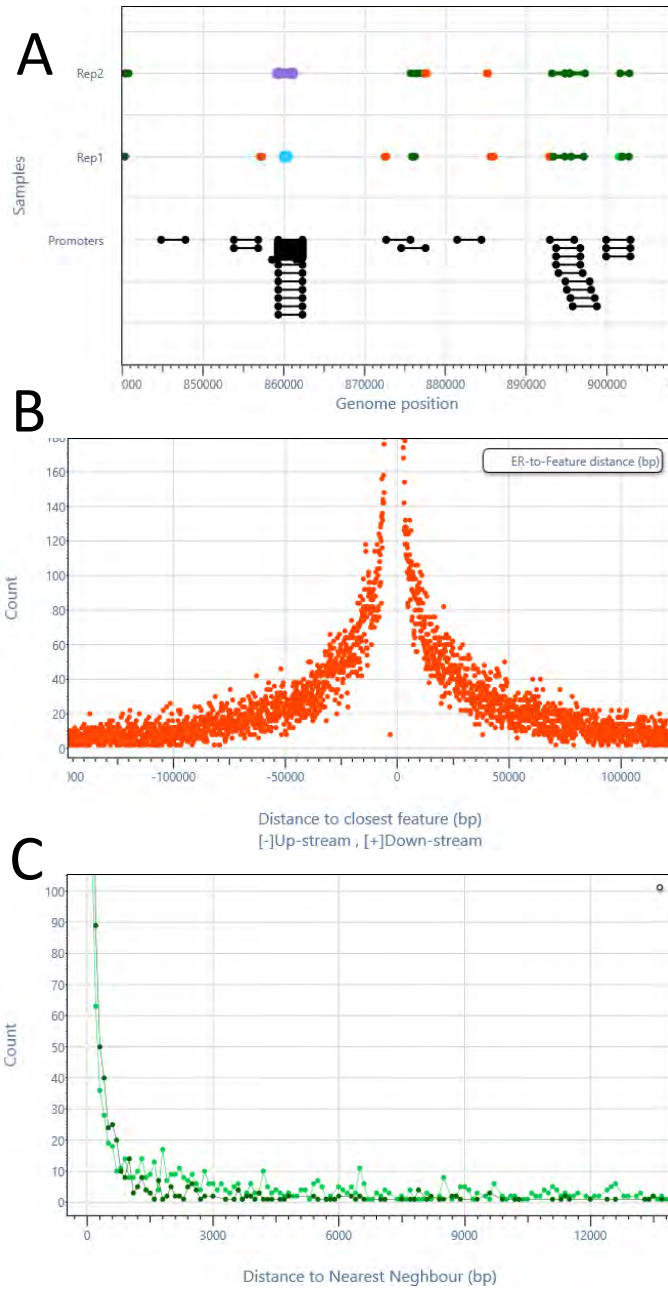


Figure 14: MuSERA at work. (A) ERs on the genome browser together with known genomic feature annotations (i.e., promoter regions); (B) Up-stream and down-stream distances of a given ER to the closest features; (C) Distances of a given ER to the nearest neighbours. Distances are measured in base pairs (bp).

5.5. Correlation assessment

MuSERA determines correlations, both at region-level and base-pair-level, between replicates. They are respectively computed as the ratio between the number of overlapping regions (region-level correlation), or genomic bases (base-pair-level correlation), and the total number of regions, or genomic bases considered, producing the respective Jaccard similarity coefficients. Base-pair-level correlation is more stringent and is to be preferred when the position of the ERs is known with certainty, while region-level correlation is instead more permissive, as it scores the overlap of entire regions rather than quantifying the magnitude of this overlap; this correlation measure is then to be preferred in presence of heterogeneous or noisy data sets. MuSERA estimates both correlations leveraging on the `SUMMIT` function of Di3, that calculates density-based co-occurrences.

6. Conclusion

We proposed Di3 (Durable Interval Inverted Index), a multi-resolution single-dimension data structure for interval-based data queries. We presented Di3's method by illustrating its model, data structure, and supported operations, and then we demonstrated its effectiveness, both as a stand-alone tool and in comparison with the state-of-the-art systems BEDOPS and BEDTools. Finally, we showed how Di3 was used as a software component within MuSERA, a tool for multiple sample peak calling. We proved that Di3 is a flexible, high-performance component, whose use as a self-standing system is much superior to current state-of-the-art.

Previously, we implemented GMQL, the innovative region-based genomic data querying system, by using cloud computing and specifically the Apache Spark and Flink engines within the Apache Hadoop framework; as future work, we plan to support an alternative implementation of the domain-specific operations of GMQL by using Di3, so as to compare cloud-based computing with specialized indexing on very large datasets. We also plan a full integration of the Di3 technology within Galaxy (Giardine et al., 2005), (Goecks et al., 2010)⁹.

⁹Vahid Jalili will join Galaxy in September 2016 as post-doc, with the objective of integrating Di3 within the Galaxy framework.

Acknowledgments

We acknowledge the essential contributions of Fernando Palluzzi, who has advised us about the biological use of Di3, and of Roger Knapp, who provided a B⁺tree implementation. Research is supported by the Data-Driven Genomic Computing (GenData 2020) PRIN project (2013-2016), funded by the Italian Ministry of the University and Research, and by a grant from Amazon Web Services.

Author contributions

Concept: Jalili, Matteucci, Ceri. Software Design: Jalili, Matteucci. Software Implementation: Jalili. Data acquisition: Jalili. Data analysis and interpretation: Jalili, Matteucci. Manuscript drafting: Jalili, Matteucci, Masseroli, Ceri. Critical revision: Ceri.

References

- 1000_Genomes_Project_Consortium, et al., 2010. A map of human genome variation from population-scale sequencing. *Nature* 467 (7319), 1061–1073.
- Afrati, F. N., Dolev, S., Sharma, S., Ullman, J. D., 2015. Bounds for overlapping interval join on MapReduce. In: *EDBT/ICDT Workshops*. pp. 3–6.
- Bentley, J. L., 1977. Solutions to Klee’s rectangle problems. Tech. rep., Technical report, Carnegie-Mellon Univ., Pittsburgh, PA.
- Bentley, J. L., 1979. Decomposable searching problems. *Information Processing Letters* 8 (5), 244–251.
- Buck, J. B., Watkins, N., LeFevre, J., Ioannidou, K., Maltzahn, C., Polyzotis, N., Brandt, S., 2011. SciHadoop: array-based query processing in Hadoop. In: *Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis*. No. 66. ACM, pp. 1–11.
- Caporaso, J. G., Lauber, C. L., Walters, W. A., Berg-Lyons, D., Huntley, J., Fierer, N., Owens, S. M., Betley, J., Fraser, L., Bauer, M., et al., 2012. Ultra-high-throughput microbial community analysis on the illumina hiseq and miseq platforms. *The ISME journal* 6 (8), 1621–1624.

- Cereda, M., Sironi, M., Cavalleri, M., Pozzoli, U., 2011. GeCo++: a C++ library for genomic features computation and annotation in the presence of variants. *Bioinformatics* 27 (9), 1313–1315.
- Chawda, B., Gupta, H., Negi, S., Faruquie, T. A., Subramaniam, L. V., Mohania, M. K., 2014. Processing interval joins on Map-Reduce. In: *EDBT*. pp. 463–474.
- Cockerill, P. N., 2011. Structure and function of active chromatin and DNase I hypersensitive sites. *FEBS Journal* 278 (13), 2182–2210.
- Cormen, T. H., Leiserson, C. E., Rivest, R. L., Stein, C., 2009. Section 14.3: Interval trees, third edition Edition. MIT press Cambridge, Ch. 14, pp. 348–354.
- Eldawy, A., Mokbel, M. F., 2013. A demonstration of spatialhadoop: an efficient mapreduce framework for spatial data. *Proceedings of the VLDB Endowment* 6 (12), 1230–1233.
- Elmasri, R., Wu, G. T., Kim, Y.-J., 1990. The time index: An access structure for temporal data. In: *Proceedings of the 16th International Conference on Very Large Data Bases*. Morgan Kaufmann Publishers Inc., pp. 1–12.
- ENCODE Project Consortium, et al., 2012. An integrated encyclopedia of dna elements in the human genome. *Nature* 489 (7414), 57–74.
- Fenwick, P. M., 1994. A new data structure for cumulative frequency tables. *Software: Practice and Experience* 24 (3), 327–336.
- Fisher, R. A., 1925. *Statistical methods for research workers*. Genesis Publishing Pvt Ltd.
- GA4GH Data Working Group, 2015. GA4GH API. URL <http://ga4gh.org/#/documentation>
- Ghanem, T. M., Shah, R., Mokbel, M. F., Aref, W. G., Vitter, J. S., 2004. Bulk operations for space-partitioning trees. In: *Data Engineering, 2004. Proceedings. 20th International Conference on*. IEEE, pp. 29–40.

- Giardine, B., Riemer, C., Hardison, R. C., Burhans, R., Elnitski, L., Shah, P., Zhang, Y., Blankenberg, D., Albert, I., Taylor, J., et al., 2005. Galaxy: a platform for interactive large-scale genome analysis. *Genome research* 15 (10), 1451–1455.
- Goecks, J., Nekrutenko, A., Taylor, J., et al., 2010. Galaxy: a comprehensive approach for supporting accessible, reproducible, and transparent computational research in the life sciences. *Genome Biol* 11 (8), R86.
- Goh, C. H., Lu, H., Ooi, B.-C., Tan, K.-L., 1996. Indexing temporal data using existing B+-trees. *Data & Knowledge Engineering* 18 (2), 147–165.
- Graefe, G., 2003. Sorting and indexing with partitioned b-trees. In: *CIDR*. Vol. 3. pp. 5–8.
- Guttman, A., 1984. R-trees: a dynamic index structure for spatial searching. Vol. 14. ACM.
- Han, J., Haihong, E., Le, G., Du, J., 2011. Survey on NoSQL database. In: *Pervasive computing and applications (ICPCA)*, 2011 6th international conference on. IEEE, pp. 363–366.
- Jalili, V., Matteucci, M., Masseroli, M., Morelli, M. J., 2015. Using combined evidence from replicates to evaluate ChIP-seq peaks. *Bioinformatics* 31 (17), 2761–2769.
- Jalili, V., Matteucci, M., Morelli, M. J., Masseroli, M., 2016. MuSERA: Multiple sample enriched region assessment. *Briefings in bioinformatics*, submitted.
- Kaufmann, M., Fischer, P. M., May, N., Ge, C., Goel, A. K., Kossmann, D., 2015. Bi-temporal timeline index: A data structure for processing queries on bi-temporal data. In: *Data Engineering (ICDE)*, 2015 IEEE 31st International Conference on. IEEE, pp. 471–482.
- Landt, S. G., Marinov, G. K., Kundaje, A., Kheradpour, P., Pauli, F., Batzoglou, S., Bernstein, B. E., Bickel, P., Brown, J. B., Cayting, P., et al., 2012. ChIP-seq guidelines and practices of the ENCODE and modENCODE consortia. *Genome research* 22 (9), 1813–1831.

- Masseroli, M., Pinoli, P., Venco, F., Kaitoua, A., Jalili, V., Palluzzi, F., Muller, H., Ceri, S., 2015. GenoMetric Query Language: a novel approach to large-scale genomic data management. *Bioinformatics* 31 (12), 1881–1888.
- Neph, S., Kuehn, M. S., Reynolds, A. P., Haugen, E., Thurman, R. E., Johnson, A. K., Rynes, E., Maurano, M. T., Vierstra, J., Thomas, S., et al., 2012. BEDOPS: high-performance genomic feature operations. *Bioinformatics* 28 (14), 1919–1920.
- Ovaska, K., Lyly, L., Sahu, B., Janne, O. A., Hautaniemi, S., 2013. Genomic region operation kit for flexible processing of deep sequencing data. *IEEE/ACM Transactions on Computational Biology and Bioinformatics (TCBB)* 10 (1), 200–206.
- Park, P. J., 2009. ChIP-seq: advantages and challenges of a maturing technology. *Nature Reviews Genetics* 10 (10), 669–680.
- Quinlan, A. R., Hall, I. M., 2010. BEDTools: a flexible suite of utilities for comparing genomic features. *Bioinformatics* 26 (6), 841–842.
- Rashid, N. U., Giresi, P. G., Ibrahim, J. G., Sun, W., Lieb, J. D., 2011. Zinba integrates local covariates with dna-seq data to identify broad and narrow regions of enrichment, even within amplified genomic regions. *Genome Biol* 12 (7), R67.
- Romanoski, C. E., Glass, C. K., Stunnenberg, H. G., Wilson, L., Almouzni, G., 2015. Epigenomics: roadmap for regulation. *Nature* 518 (7539), 314–316.
- Schuster, S. C., 2007. Next-generation sequencing transforms today’s biology. *Nature* 200 (8), 16–18.
- Schuster, S. C., 2008. Next-generation sequencing transforms today’s biology. *Nature methods* 5 (1), 16–18.
- Shendure, J., Ji, H., 2008. Next-generation DNA sequencing. *Nature biotechnology* 26 (10), 1135–1145.
- Snodgrass, R., Ahn, I., 1985. A taxonomy of time databases. In: *ACM Sigmod Record*. Vol. 14. ACM, pp. 236–246.

- Sriharsha, R., 2015. Magellan.
URL <http://spark-packages.org/package/harsha2010/magellan>
- Stantic, B., Topor, R., Terry, J., Sattar, A., 2010. Advanced indexing technique for temporal data. *Computer Science and Information Systems* 7 (4), 679–703.
- Tudorica, B. G., Bucur, C., 2011. A comparison between several NoSQL databases with comments and notes. In: *Roedunet International Conference (RoEduNet)*, 2011 10th. IEEE, pp. 1–5.
- Vlissides, J., Helm, R., Johnson, R., Gamma, E., 1995. *Design patterns: Elements of reusable object-oriented software*. Reading: Addison-Wesley 49 (120), 11.
- Weinstein, J. N., Collisson, E. A., Mills, G. B., Shaw, K. R. M., Ozenberger, B. A., Ellrott, K., Shmulevich, I., Sander, C., Stuart, J. M., Network, C. G. A. R., et al., 2013. The cancer genome atlas pan-cancer analysis project. *Nature genetics* 45 (10), 1113–1120.
- Yu, J., Wu, J., Sarwat, M., 2015. GeoSpark: A cluster computing framework for processing large-scale spatial data.
- Zaharia, M., Chowdhury, M., Das, T., Dave, A., Ma, J., McCauley, M., Franklin, M. J., Shenker, S., Stoica, I., 2012. Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing. In: *Proceedings of the 9th USENIX conference on Networked Systems Design and Implementation*. USENIX Association, pp. 15–28.
- Zhang, Y., Liu, T., Meyer, C. A., Eeckhoute, J., Johnson, D. S., Bernstein, B. E., Nusbaum, C., Myers, R. M., Brown, M., Li, W., et al., 2008. Model-based analysis of chip-seq (macs). *Genome biology* 9 (9), R137.

Author biographies



Vahid Jalili is a Ph.D. candidate at Politecnico di Milano, Dipartimento di Elettronica, Informazione e Bioingegneria, where his research on tertiary analysis of Next Generation Sequencing data is focused on systematic solutions for analytical and computational challenges. His research interest spans topics in computational biology, cognitive science, and artificial intelligence; such as information retrieval and data mining in Genomics, cognitive inhibition and visual perception, and application of artificial intelligence to games.



Matteo Matteucci is Associate Professor at the Dipartimento di Elettronica Informazione e Bioingegneria of Politecnico di Milano. In 1999 he got a Laurea degree in Computer Engineering at Politecnico di Milano, in 2002 he got a Master of Science in Knowledge Discovery and Data Mining at Carnegie Mellon University (Pittsburgh, PA), and in 2003 a PhD in Computer Engineering and Automation at Politecnico di Milano (Milan, Italy). His main research topics are pattern recognition, machine learning, machine perception, robotics, computer vision and signal processing. He has co-authored more than 150 scientific international publications and he has been involved in national and international funded research projects.



Marco Masseroli received the Laurea Degree in Electronic Engineering in 1990 from Politecnico di Milano, Italy, and a PhD in Biomedical Engineering in 1996, from Universidad de Granada, Spain. He is Associate Professor at the Dipartimento di Elettronica, Informazione e Bioingegneria (DEIB) of Politecnico di Milano, Italy, and lecturer of Bioinformatics and BioMedical Informatics. His research activity is on the application of information technology to the medical and biological sciences in several Italian and international research centers. He has also been Visiting Professor at the Departamento de Anatomía Patológica, Facultad de Medicina of the Universidad de Granada - Spain, and Visiting Faculty at the Cognitive Science Branch of the National Library of Medicine, National Institute of Health, Bethesda - US. His research interests are in the area of bioinformatics and biomedical informatics, focused on distributed Internet technologies, biomolecular databases, controlled biomedical terminologies and bio-ontologies to effectively retrieve, manage, analyze, and semantically

integrate genomic information with patient clinical and high-throughout genomic data. He is the author of more than 170 scientific articles, which have appeared in international journals, books and conference proceedings.



Stefano Ceri is Professor at the Dipartimento di Elettronica, Informazione e Bioingegneria (DEIB) of Politecnico di Milano. He was Visiting Professor at the Computer Science Department of Stanford University (1983-1990), Chairman of the Computer Science Section of DEI (1992-2004), Director of Alta Scuola Politecnica (ASP) of Politecnico di Milano and Politecnico di Torino (2010-2013). In 2008 he has been awarded an advanced ERC Grant on Search Computing (2008-2013). He is co-founder (2001) of WebRatio (<http://www.webratio.com/>). His research work has been generally concerned with extending database technology to incorporate new features: distribution, object-orientation, rules, streaming data, crowd-based and genomic computing. He is currently leading the PRIN project GenData 2020, focused on building query and data analysis systems for genomic data as produced by fast DNA sequencing technology. He is the recipient of the ACM-SIGMOD "Edward T. Codd Innovation Award" (2013), and an ACM Fellow and member of the Academia Europaea.