

Detection and functional annotation of unique genomic regions

Dissertation in fulfillment of the requirements for the
degree of *Doctor rerum naturalium*
of the Faculty of Mathematics and Natural Sciences
at Christian Albrechts University of Kiel

Submitted by Beatriz Vieira Mourato

Plön, October 2024

URN: urn:nbn:de:gbv:8:3-2024-01212-2

First examiner: Prof. Dr. Bernhard Haubold
Second examiner: Prof. Dr. Tal Dagan

Date of the oral examination: 10th December 2024

Acknowledgements

First, I thank my supervisor, Bernhard Haubold for giving me the opportunity to work on this great, but challenging, project. Your constant support over the years during the development of this thesis was highly appreciated. I am also grateful for our endless conversations on science, books and many other topics. These interactions were a great support into my daily life and constantly inspired me to learn more about the world we live in.

I also thank Tal Dagan and Julien Dutheil for agreeing to be part of my thesis advisory committee and for their helpful advice over the years. Additionally, I thank them both and Olivia Roth for agreeing to be a member of my thesis committee.

I owe much of who I became, both scientifically and personally, to the scientific community I found in Kiel and Plön. Your guidance and support have been invaluable. Many of you accompanied me through endless summers and dark winters: from random trips, to gastronomic experiments and other last minute adventures. More specifically, I thank Lavisha for introducing me to the Institute and Elio, Amor, Christina and Gisela for numerous cheerful moments together. Life in Plön wouldn't have been the same without your presence.

I thank Angela Carollo for inspiring me in my path as a data scientist and Adrianna Vucetic for the many hours spent together coordinating the PhDNet survey group. Even though we were miles apart, your input was always appreciated.

Finally, but not least, I am grateful to my family and friends who have supported me since the beginning of this journey, without whom I would have never made it this far. I especially thank my mother for always encouraging me and for giving me the courage to apply for a PhD position.

I want to end with a special thank you to all who saw the beginning of this journey, but not the end of it. Your presence will be missed.

Abstract

Every organism has a genome. In some organisms, like mammals, genomes are highly repetitive. Repetitive genomic regions are associated with multiple biological functions as they can influence genome organization and gene regulation. Therefore, considerable importance is placed on identifying, localizing and categorizing them. Perhaps less studied but also important is the identification and categorization of unique genomic regions. Although unique regions have been associated with housekeeping and developmental genes and are thus crucial for survival, there is a distinct lack of bioinformatic tools for specifically finding unique regions in genomic sequences.

This thesis explores existing tools for finding unique genomic regions, expands their application domains, and establishes workflows for applying them in comparative genomics.

In Chapter 2, I start by identifying genomic regions in mammalian genomes that are unique within their genome. The strong association between these regions and developmental genes is established in 18 mammalian species spanning placentals and marsupials. This link between a sequence feature, uniqueness, and functions, development, may make it possible to identify developmentally important genomic regions in non-model organisms.

In Chapters 3 and 4, I expand the concept of unique regions, such that unique regions are now those present in one or more queries and absent from one or more subjects, where the subject genomes are the closest distinct relatives of the query genomes. In Chapter 3 I use these unique regions to design diagnostic PCR markers for 120 bacterial species. In Chapter 4 I use unique regions from human and mouse genomes in an attempt to associate lineage-specific regions with species-specific phenotypes. For these two chapters, I improve existing tools for large-scale “hypothesis-free” searches for unique regions across bacteria and mammalian genomes.

This thesis contributes to the toolkit of comparative genomics by improving existing tools and integrating them into scaleable work flows for the detection of unique regions. Once detected, the unique regions are investigated for their functional content in mammals (Chapters 2 and 4), and for their suitability as diagnostic markers in bacteria (Chapter 3).

Kurzfassung

Jeder Organismus hat ein Genom. Bei einigen Organismen, wie z. B. Säugetieren, sind die Genome stark repetitiv. Repetitive Genomregionen werden mit zahlreichen biologischen Funktionen in Verbindung gebracht, da sie die Genomorganisation und die Genregulation beeinflussen können. Daher wird der Identifizierung, Lokalisierung und Kategorisierung dieser Regionen große Bedeutung beigemessen. Weniger erforscht, aber ebenfalls wichtig, ist die Identifizierung und Kategorisierung einzigartiger genomischer Regionen. Obwohl einzigartige Regionen mit Housekeeping- und Entwicklungsgenen in Verbindung gebracht werden, und somit für das Überleben entscheidend sind, gibt es kaum bioinformatische Werkzeuge für das spezifische Auffinden einzigartiger Regionen in genomischen Sequenzen.

In dieser Arbeit werden bestehende Werkzeuge zum Auffinden einzigartiger genomischer Regionen untersucht, ihre Anwendungsbereiche erweitert und Protokolle für ihre Anwendung in der vergleichenden Genomik erstellt.

In Kapitel 2 beginne ich mit der Identifizierung von genomischen Regionen in Säugetiergenomen, die innerhalb ihres Genoms einzigartig sind. Die enge Verbindung zwischen diesen Regionen und Entwicklungsgenen wird bei 18 Säugetierarten, von den Plazentatieren bis zu den Beuteltieren, nachgewiesen. Diese Verbindung zwischen einem Sequenzmerkmal, der Einzigartigkeit, und Funktionen, Entwicklung, könnte es ermöglichen, entwicklungsrelevante genomische Regionen in Nicht-Modellorganismen zu identifizieren.

In den Kapiteln 3 und 4 erweitere ich das Konzept der einzigartigen Regionen dahingehend, dass einzigartige Regionen nun diejenigen sind, die in einem oder mehreren Query-Genomen vorhanden sind, und in einem oder mehreren Subject-Genomen fehlen, wobei die Subject-Genome die engsten Verwandten der Query-Genome sind. In Kapitel 3 verwende ich diese einzigartigen Regionen, um diagnostische PCR-Marker für 120 Bakterienarten zu entwickeln. In Kapitel 4 verwende ich einzigartige Regionen aus den Genomen von Mensch und Maus, um zu versuchen, stammesspezifische Regionen mit artspezifischen Phänotypen in Verbindung zu bringen. In diesen beiden Kapiteln verbessere ich bestehende Werkzeuge für die groß angelegte "hypothesenfreie" Suche nach einzigartigen Regionen in Bakterien- und Säugetiergenomen.

Diese Arbeit leistet einen Beitrag zum Instrumentarium der vergleichenden Genomik, indem sie bestehende Werkzeuge verbessert und in skalierbare Arbeitsabläufe für die Entdeckung einzigartiger Regionen integriert. Einmal entdeckte einzigartige Regionen wer-

den auf ihren funktionellen Gehalt in Säugetieren (Kapitel 2 und 4), und auf ihre Eignung als diagnostische Marker in Bakterien (Kapitel 3) untersucht.

Contents

1	General introduction	1
1.1	Exact matching	2
1.1.1	Unique regions in a single query	4
1.1.2	Unique regions between a query and a subject	5
1.2	Suffix arrays	6
1.3	The genomics era	8
1.4	Thesis scope	12
2	Highly complex regions in mammalian genomes	15
2.1	Introduction	16
2.1.1	Transposable elements and conserved regions	16
2.1.2	Defining highly complex regions	18
2.1.3	On the quality of genome assemblies	21
2.1.4	Picking mammalian genomes	22
2.1.5	Chapter aims	23
2.2	Methods	23
2.2.1	Data	23
2.2.2	Identification of transposons in highly complex regions	24
2.2.3	Phylogenetic tree	26
2.2.4	Detecting complex regions	26
2.2.5	Annotation and enrichment analysis	27
2.2.6	Hardware/Software	30
2.2.7	Data availability	30
2.3	Results	30
2.3.1	Phylogenetic tree	30
2.3.2	Transposons and complex regions	31
2.3.3	Identification of highly complex regions	33
2.3.4	Time and memory consumption	41
2.4	Discussion	43

2.4.1	Detection of highly-complex regions	45
2.4.2	Enrichment analysis	46
2.5	Summary	47
3	Large-scale marker discovery in prokaryote genomes	49
3.1	Introduction	50
3.1.1	Finding diagnostic markers	50
3.1.2	Curating targets and neighbor	53
3.1.3	Evaluating primer pairs	55
3.1.4	<i>Mapro</i> : markers for prokaryotes	55
3.1.5	Chapter aims	57
3.2	Methods	57
3.2.1	Fur algorithm	57
3.2.2	Simulating data	59
3.2.3	Large-scale diagnostic marker finding	61
3.3	Results	67
3.3.1	Simulated data	67
3.3.2	Designing diagnostic primers in the large	71
3.4	Discussion	83
3.4.1	<i>nesa</i> and <i>resa</i>	84
3.4.2	Large-scale marker discovery	85
3.5	Summary	87
4	Lineage-specific regions in mammalian genomes	89
4.1	Introduction	90
4.1.1	Humans and chimpanzees	90
4.1.2	The house mouse superspecies complex	92
4.1.3	Chapter aims	94
4.2	Methods	96
4.2.1	Data collection	97
4.2.2	Phylogenetic tree	97
4.2.3	Human-specific regions	100
4.2.4	House mouse-specific regions	101
4.2.5	Hardware/Software	102
4.2.6	Data availability	102
4.3	Results	102
4.3.1	Haplotype-specific regions	103

4.3.2	Human-specific regions	105
4.3.3	Common chimpanzee-specific regions	112
4.3.4	<i>Mus musculus</i> specific regions	116
4.3.5	Runtime and resources	118
4.4	Discussion	121
4.4.1	Haplotype-specific insertions	122
4.4.2	Human-specific insertions	122
4.4.3	House mouse subspecies complex	124
4.4.4	lncRNAs and their function	125
4.4.5	<i>Fur</i> and unique satellite regions	126
4.5	Summary	127
4.6	Supplementary material	128
5	General discussion and future directions	133
5.1	Unique regions in mammalian genomes	135
5.1.1	Large-scale survey of mammalian genomes	135
5.1.2	Genome annotation	136
5.1.3	Memory efficient <i>Macle</i>	136
5.2	Markers in prokaryotes	138
5.2.1	Navigating taxonomic databases	139
5.2.2	In-depth quantification of marker regions	140
5.3	Lineage-specific regions in mammalian genomes	140
5.3.1	Further assessments on the masking step of <i>Fur</i>	141
5.3.2	Individually variable regions	141
5.3.3	Characterization of lncRNA	142
5.3.4	Lineage-specific accelerated regions (linARS)	143
5.4	Further improvements on <i>Fur</i>	143
5.5	General conclusion	145
	Bibliography	147
	Data availability	161
	List of Publications	163
	Author contributions	165
	Statement	167

Chapter 1

General introduction

Beatriz Vieira Mourato¹

Affiliations:

¹Max Planck Institute for Evolutionary Biology, RG Bioinformatics, Plön, Germany

In preparation:

Section 1.1 of this chapter will be included in an invited mini review, “Fast Detection of Unique Genomic Regions” for *Computational and Structural Biotechnology Journal*.

Bioinformatics was founded under the realization that similar regions perform similar functions. This observation meant that given a gene with an unknown function, similarity searches can be used to find known similar genes with a known function.

Since the development of the first optimal global alignment algorithm in 1970, and later its generalization, in 1981, to local alignment [1, Ch.2], similarity searches have been crucial in bioinformatics. From *de novo* assemblies, to sequence annotations and comparative genomics, similarity searches are practically everywhere.

Also important, though less well known are dissimilarity searches. Dissimilarity searches allow the identification of unique regions, that can be detected within individual genomes or between pairs of samples of target and non-target genomes [2].

In individual genomes, unique genomic regions have been investigated since the 1980s. A study in 1985, based on restriction enzyme analysis, showed that 1% of the mouse genome comprised by CpG islands was single-copy, i.e. unique in the genome [3]. Later CpG islands were revealed to be associated with housekeeping genes [4, 5] and to influence chromatin structure [6]. Furthermore, promoters associated with long CpG islands were also found to be associated with developmental genes [7]. From this it follows that in mammals, the study of unique regions in a genome can lead to insights about gene functions.

Instead of using *in vitro* techniques like restriction enzyme analysis, modern genomics is based on the *in silico* study of the DNA sequence. An early search for unique regions in humans and mouse, showed that these regions are highly enriched for developmental genes [8].

Dissimilarity searches can also be used to search for functional differences between organisms by directly comparing two or more genomes. These comparisons have been used to gain insight into shared biological functions [9] or to identify species-specific traits [2].

Despite the apparent importance of these regions, there are few computational tools dedicated to their identification. This is because while conceptually, unique regions can be identified by taking the complement of a similarity search, their accurate detection at a genomic scale turns out to be more challenging.

1.1 Exact matching

Optimal aligners that perform global or local alignments, although highly accurate cannot deal with the ever-increasing size of sequence databases due to their high memory and

time requirements. This led to the development of heuristic aligners. While optimal aligners use dynamic programming to ensure the best possible alignment between two sequences, heuristic programs prioritize speed and efficiency over the guarantee of finding the optimal alignment. The most successful heuristic aligner to date, *Blast* [10, 11] was published in 1990 and uses a seed and extension mechanism. In a first stage, heuristic aligners search for seeds, short k -mers that are identical between the query and subject. Then, the seeds are extended into longer alignments using inexact matching [12, p.99f].

The identification of the seeds in heuristic aligners is done through the use of advanced methods of exact matching like keyword trees [10, 13]. Exact matching aims at finding all occurrences of a given pattern in a text where patterns are usually short and the text is long and can be calculated based on two different strategies: k -mers and maximal matches. K -mers used in programs like *Blast*, are exact matches of a fixed length, k . In contrast, maximal matches cannot be extended to the left or to the right without incurring a mismatch. This is the strategy used in fast genome aligners like *Mummer* [14].

Given that fast aligners like *Blast* and *Mummer* are based on exact matches, exact matching can also be used to search for regions that have no homology. The lack of homology can be with respect to a query itself, or between a query and a subject. Figure 1.1 shows this classification of unique regions into query-only and query/subject and tools for finding them. Tools based on k -mers are shown in red, and tools based on maximal matches in blue.

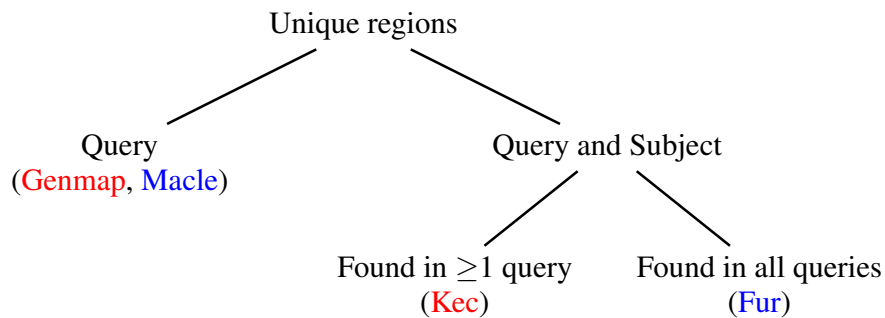


Figure 1.1: Classification of unique regions. Unique regions can be found within a query, or between queries and subjects. Unique regions found between queries and subjects can either be present in at least one query or in multiple queries. Programs shown in red are based on k -mers, while programs in blue are based on maximal matching.

This thesis focuses on the two tools based on maximal matches, *Macle*, which searches for unique regions within a query, and *Fur*, which searches for regions present in all queries that are absent from the subjects.

1.1.1 Unique regions in a single query

When searching for unique regions within a single query, we are in fact searching for non-repetitive genomic regions. These unique regions are transposon and duplication free. It has been known for two decades that transposon-free regions are associated with highly conserved genomic regions involved in regulatory mechanisms including transcription factors like the *Hox* genes that determine the body plan of animals [15–17].

As shown in Figure 1.1, there are currently two tools that can be applied to search for unique genomic regions within a single query: *Genmap* [18], which is k-mer based, and *Macle* [8], which relies on maximal matches.

Genmap calculates the “mappability” of a sequence, where mappability is the inverse of the number of k-mers starting at each position found across the sequence. Given a sequence $S = \text{TAAATAAAATTT}$, the mappability of each k-mer size 3 is shown in Table 1.1, where two unique k-mers, ATT and TTT, can be found at the 3’ end of the sequence.

Table 1.1: K-mer counts and mappability with $k = 3$ along $S = \text{TAAATAAAATTT}$, obtained with *Genmap*

position	1	2	3	4	5	6	7	8	9	10	11	12
sequence	T	A	A	A	T	A	A	A	A	T	T	T
count	2	3	2	1	2	3	3	2	1	1	–	–
mappability	0.5	0.3	0.5	1.0	0.5	0.3	0.3	0.5	1	1	–	–

In contrast, *Macle*, calculates the length of all right-maximal matches in a given sequence. As explained in Section 1.2, right maximal matches can be quickly looked up in suffix arrays.

Homologous genomic regions tend to have longer maximal matches, while unique regions tend to have shorter matches. In fact, matches lengths in unique regions follows a known null distribution for random sequences [19]. Figure 1.2 shows the theoretical null distribution of the genome of *Escherichia coli* K12 compared to the true distribution and the distribution in a shuffled version of the K12 genome. Notice that the curves from the theory and the curves from the shuffled sequence are almost indistinguishable, while the match length distribution for the original sequence is quite different.

I used *Macle* rather than *Genmap* to search for unique regions in query sequences in this thesis, because of the extensive theoretical work on the null distribution of maximal matches lengths underlying *Macle* [8, 19]. Given the length and GC content of the input sequence, this null distribution allows regions to be reliably classified as unique and non-unique. No such classifier is available for the mappability calculated with *Genmap*.

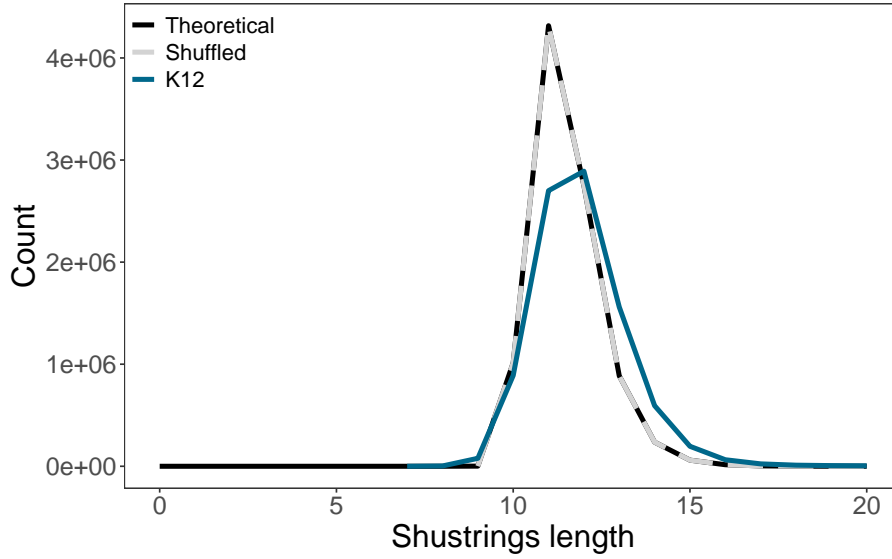


Figure 1.2: Null distribution of matches length for *Escherichia coli* K12 sequence, in blue, the matches length of the shuffled sequence are in a dashed gray line, and the respective theoretical distribution, is shown in black.

1.1.2 Unique regions between a query and a subject

Unique regions can also refer to regions that are present across one or more queries but absent from one or more subjects. In this scenario, the query-specific unique regions can be used to create PCR primers for diagnostic markers or to find lineage-specific phenotypes.

As shown in Figure 1.2, there are currently two tools that can be applied to search for unique genomic regions in a query, that are absent from the subject: *Kec* [20], which is k-mer based and *Fur* [21], which relies on maximal matches.

The program *Kec* performs two operations: exclusion and inclusion. When excluding, *Kec* creates two independent lists of k-mers present in both the query and the subject sequences. Then it cross-references the lists such that k-mers present in both the query and the subject are excluded. At the end, remaining overlapping k-mers are merged. This results in a list of k-mers present in the query that are excluded from the subject. The inclusion function performs the opposite operation. Given a query representative, and a pool of query sequences, *Kec* retains k-mers that are found in both k-mers list. Since k-mers need to appear only once to be included, there are no guarantees that the unique region is present across all query sequences. Hence, while unique regions detected with *Kec* must be present in at least one query, they are not necessarily present in all as shown in Figure 1.1.

Just like *Macle*, *Fur* also uses maximal matches to identify unique regions. However,

while *Macle* looks up within-query matches in a suffix array, *Fur* uses a suffix array to search for maximal matches between queries and subjects. This can be done by indexing the query representative and “streaming” the subject sequence against the suffix array. Unlike *Kec*, which performs the intersection of marker regions across queries and afterwards the exclusion of k-mers also present in the subject, *Fur* starts by identifying regions in a query representative that are unique, in other words absent, from the subject sequences. Then *Fur* retains all unique regions that are shared across all queries. This “All or Nothing” approach, which is quite distinct from *Kec*, is why *Fur* was the preferred tool to search for unique regions between queries and subjects in this Thesis.

1.2 Suffix arrays

One way to search for exact matches in linear time is through the use of suffix trees, a data structure that stores all the suffixes of a given text [22]. A suffix tree is composed of a root and several leaves, where each leaf corresponds to a distinct suffix starting at the leaf labelled in the input string. Suffixes that share the same prefix are located along the same tree branch. Figure 1.3 shows an example of a suffix tree for the sequence $S = \text{TAAATAAAATTT}$. The suffix tree of the example sequence contains 12 leaves, each corresponding to a distinct suffix present in the sequence. The 13th string position is a sentinel character, \$, to ensure that no suffix is a prefix of another suffix, in which case there would be no leaf for it. For example, the suffix $S[11 \dots] = \text{TT}$ is a prefix of the suffix $S[10 \dots] = \text{TTT}$. Here the sentinel forces a mismatch, and hence a leaf, beyond the last position of TT .

The internal nodes of the suffix tree are labelled with circled numbers which denote the “string depth” or match length of a given prefix. The maximal match length of a leaf is given by the match length of its parental node. In Figure 1.3, the first suffix, positioned at the first leaf has a maximal match length of 4.

Suffix trees can be used to search whether a pattern is present in a given text to find the longest repeated substring or to find the longest palindrome in the text [22]. Suffix trees have two advantages over the use of simpler data structures. First, suffix trees require linear time to be constructed proportional to the size of the string. Secondly, one of the most important aspects of suffix trees is that the time it takes to search for a pattern in a text is proportion to the length of the pattern and not the length of the text.

There are, however, some drawbacks to using suffix trees. Although constructing a suffix tree requires time and memory proportional to the size of a text of length m , $O(m)$, the

position to the left of the first entry in the `lcp` array, that entry is uninterpretable, which is marked by -1

Table 1.2: Suffix array, `isa` and `lcp` of a sequence S , where $S = \text{TAAATAAAATTT\$}$.

i	<code>isa</code>	<code>sa</code>	<code>lcp</code>	Suffix
1	11	13	-1	\$
2	3	6	0	AAAATTT\$
3	5	2	3	AAATAAAATTT\$
4	7	7	4	AAATTT\$
5	10	3	2	AATAAAATTT\$
6	2	8	3	AATTT\$
7	4	4	1	ATAAAATTT\$
8	6	9	2	ATTT\$
9	8	12	0	T\$
10	13	5	1	TAAAATTT\$
11	12	1	4	TAAATAAAATTT\$
12	8	11	1	TT\$
13	1	10	2	TTT\$

Suffix arrays that include additional data structures like the `isa` and the `lcp` are denominated enhanced suffix arrays, `esa`.

1.3 The genomics era

The first complete genome to be sequenced was that of the bacteriophage ϕX174 (5,386 bp) in 1978 [24]. Four years later in 1982, whole genome shotgun sequencing was successfully applied for the first time to the genome of bacteriophage λ (48,502 bp) [25]. Still, it was not until 14 years later, in 1996, that whole genome sequencing was used to sequence the genome of a free-living organism, *Haemophilus influenzae* Rd (1.8 Mb) [26]. Afterwards, in just five years, several breakthroughs were achieved, with increasingly larger genomes being sequenced: the first eukaryotic genome (*Sacharomyces cerevisiae* [27]), the first metazoan genome (*Caenorhabditis elegans* [28]), the first plant genome (*Arabidopsis thaliana* [29]), the first metazoan with whole-genome shotgun sequencing (*Drosophila melanogaster* [30]) and finally the first draft of the Human genome [15, 31, 32].

If one were to select a starting date for the genomics era, it would be 2001, when the Human Genome Project announced the first complete draft sequence of the human genome [15, 31]. The ambitious project was expected to revolutionize our understanding of ge-

netics and, with it, contribute to advances in cancer research, drug development and allow the identification of genetic diseases based on inheritance patterns [15, 31–33].

After the draft of the human genome was published, several mammalian genomes such as the mouse [9], dog [34], rat [35] and chimpanzee [2] were also released and compared to the human genome with the aim of gaining insights into both evolutionary processes and gene functionality [36].

This dawn of the genomics era, was not only due to the development of better sequencing technologies but also due to advances in computational biology. The increase in sequencing data created a demand for bioinformatics and other computational biologists that could handle and process the amounts of data that was becoming widely accessible [33].

Since then, the number of sequences publicly available in databases like the NCBI keeps increasing, with the number of bases doubling approximately every 18 months [37]. Figure 1.4 shows the growth of GenBank since the early 1990s, and the increase in Whole Genome Shotgun (WGS) submissions available at the WGS sequencing database which was established in 2002.

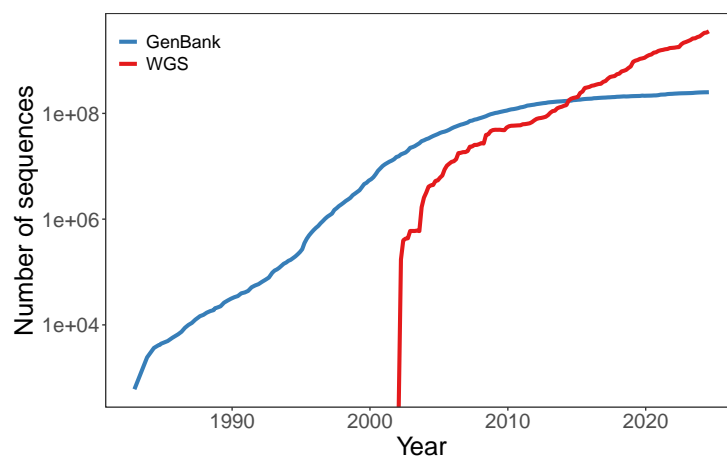


Figure 1.4: Number of bases submitted at GenBank and WGS over the years [37].

Whereas before comparative genomic studies would only compare two genomes at a time, now studies might compare tens if not hundreds of genomes to each other. Thus, it becomes crucial to develop tools that are memory efficient as that is often the limiting resource in computation.

Furthermore, as sequencing technologies improve it's not only the number of available genomes that keeps increasing, but also their quality.

Although often referred to as the first complete human genome, the genome assembly

released in 2001 was in fact incomplete. 8% of the human genome, mostly localized in centromeres, telomeres and other highly repetitive regions remained unknown. Since these repetitive regions are extremely long, the available sequencing technology could not fully span them. This meant that during the assembly of the repeats, repetitive regions would often collapse and be assembled on top of each other.

Recent advances in sequencing, like the Oxford Nanopore’s “ultra-long” reads, mean that now it is possible to obtain reads of more than 100 Kb [38]. Due to this, it was possible to fully resolve the centromere region of the human Y chromosome [39] and to obtain a telomere to telomere assembly of the human X chromosome [40]. With this, the telomere to telomere (T2T) human genome was released on March 2022, 21 years after the first draft of the human genome [40, 41]. This assembly, called T2T-CHM13, is the first gapless sequence of a human genome, adding approximately 3.1 Mb of previously missing regions.

Since then, other telomere-to-telomere projects, like the T2T Primate Project [42, 43] have been launched with the aim of also improving the genomes assemblies of other species. Thus, we can expect that soon telomere-to-telomere genome assemblies will become the *de facto* standard for reference genomes.

The constant increase in the number of available genomes, together with the improvements in sequencing technologies, means that the current bottleneck in genomics is no longer obtaining genome sequences or their quality. Instead, the current bottleneck for large-scale genomics studies is the lack of comprehensive genome annotations associated with the existing genomes.

Genome annotation

After a genome has been sequenced and assembled, it is annotated. This means identifying and describing the functional elements of the genome, like genes, regulatory regions and so on. Genome annotation, however, can be complex especially in eukaryotes due to their large genomes and the presence of introns [44]. While major annotation databases like Ensembl [45] can help overcome this gap by providing and curating the genome annotations, of currently 324 organisms, they cannot keep up with the speed at which new genomes become available.

The difficulty in the annotation of complex genomes lead to the need of quality metrics that can be used to assess both the quality of a genome assembly and its annotation beyond the N50. These metrics can quantify the number of annotated genes, the number of exons

per gene, or their average length, for example [46, 47]. Currently, one of the most popular metrics to measure the completeness of a genome is through BUSCO (Benchmarking Universal Single-Copy Orthologs) [48]. BUSCO works by defining sets of ortholog genes that are expected to be present in a specific phylogenetic clade. The completeness (C) of a genome is then assessed by evaluating how many of the genes contained in the set of expected genes can be identified in the assembly. The assembly is classified according to the number of complete single-copy (S), duplicated (D), fragmented (F) or missing (M) genes out of a total of n expected genes. For example, the T2T assembly of the human genome contains a BUSCO completeness score of C:99.1% [S:98.5%, D:0.7%, F:0.2%, M:0.7%, n :13780].

Although quality measures like N50 and BUSCO scores tend to show a positive association with each other, where assemblies with higher N50 also have high BUSCO scores, this association is not perfect. Assemblies with a low N50 score, and thus seemingly more fragmented, can also yield good BUSCO scores. If this is the case, then the genes themselves are well assembled, even though the assembly may be more fragmented [49]. This suggests one should not rely on a single measure of quality when assessing an assembly.

Genome annotation can be split into two categories: structural and functional annotation [50]. Structural annotation refers to the annotation of structural genomic features, like the location of genes, regulatory regions and so on. On the other hand, functional annotation refers to the characterization and description of gene functions, gene variants and protein isoforms.

Genomes can be structurally annotated through evidence-based methods or prediction methods [46]. Prediction methods rely on *ab initio* approaches to identify promoters, coding, and non-coding regions, or intron-exon junctions in the genome sequence [46, 50]. Generally speaking, however, comprehensive annotation pipelines rely on evidence-based approaches. Here sequence expressed sequence tags (EST), RNA-sequence data, complementary DNA (cDNA), or protein evidence are used together with *ab initio* methods to reach consensus on the best model for each gene [46].

After structurally annotating a genome, ideally a functional annotation should also be performed, as functional annotations can link genes and their respective functions to phenotypes [50].

Good functional annotations rely on experimental data; however, experimental data is often restricted to model organisms. Therefore, an alternative to manual curation of gene functions is through the use of similarity searches, where similar proteins are assumed to

share similar functions. Currently, it is recommended to transfer gene functions across organisms through orthology searches, instead of homology searches. This is because paralogous genes, arising from duplications, are more likely to be associated with different gene functions, than orthologous genes, which are descended from the same ancestral gene. The paralogy and orthology relationship between genes can be inferred through the use of orthology databases like eggNOG [51]. These orthology-based databases are thus essential for functionally annotating genomes, as they deliver better identification of gene functions.

After a gene has been annotated its function can be classified into three ontologies: molecular functions (MF), if the function is associated with biochemical activity, cellular components (CC), the location where the gene protein is active, and biological processes (BP), the biological target of the protein [52, 53]. These three ontologies, also called GO terms, were created by the Gene Ontology (GO) consortium to standardize the vocabulary and framework used to describe gene functions and products across all of life.

Such a standardized framework for describing gene functions is invaluable. It makes it possible to, regardless of the species or model organism, to communicate in an interdisciplinary manner across several fields of research, from clinical studies to comparative genomics and basic molecular biology. Even so, despite the importance of this resource, the fact is that functional annotation relies on existing comprehensive evidence-based annotation pipelines to curate and validate the associations between genes and functions. This remains a limiting factor in attempts to functional annotate many of organisms whose genome is available. For example, as of September 2024, the GO database contained annotations related to biological processes in 536 mammals, however only 49 of these species (9.14%) had more than 100 associated annotations for biological functions. Thus, the current bottleneck in genomics is not so much the lack of sequenced organisms as the capability to extract biological information from the sequenced genomes. In this thesis I concentrate on the functional content of one specific sequence feature, uniqueness.

1.4 Thesis scope

Unique regions are an often ignored set of genomic regions that can give insight into both evolutionary processes and biological functions. The lack of interest in this field is also reflected in the lack of available tools for performing dissimilarity searches and hence finding unique regions.

The aim of this thesis is to explore existing tools for finding unique regions and to further

expand their domains of application. The three central chapters comprising this thesis focus on unique regions in three settings.

Chapter 2: Highly complex regions in mammalian genomes

In the second chapter I search for unique regions in a single query: mammalian genomes. While a previous study showed that unique regions are associated with developmental genes [8], the study was restricted to only two species: humans and mice. Here I expand on this study by investigating whether the association between unique regions and developmental genes is preserved across the mammalian clade, and what are the most important functions associated with these regions.

Chapter 3: Large-scale marker discovery in prokaryote genomes

In the third chapter I search for unique regions between queries and subjects in bacteria. Since current tools for finding unique regions between query and subjects do not scale to large datasets or large genomes, it is necessary to improve these tools. In this chapter I explore how a memory efficient tool for finding unique regions can be constructed, and how this can be used to detect diagnostic marker regions in whole genomes with an “hypothesis-free” approach that is independent of candidate genes.

Chapter 4: Lineage-specific regions in mammalian genomes

In the fourth chapter I once again search for unique regions between queries and subjects but this time in mammalian species. Using the improved tools developed in the previous chapter I search for lineage-specific regions in both apes and mice. Since lineage-specific regions may give insight into species-specific phenotypes, I then briefly expand on the biological functions of these regions.

Chapter 2

Highly complex regions in mammalian genomes

Beatriz Vieira Mourato¹ and Bernhard Haubold¹

Affiliations:

¹Max Planck Institute for Evolutionary Biology, RG Bioinformatics, Plön, Germany

Submitted manuscript:

This manuscript was published on Genes, Genomes, Genetics (G3) in November 2024.

DOI: <https://doi.org/10.1093/g3journal/jkae257>. [54]

2.1 Introduction

2.1.1 Transposable elements and conserved regions

Transposons are mobile DNA elements that occur in both prokaryotes and eukaryotes but tend to be present in higher quantities within the latter. For example, the first draft of the human genome identified that $\sim 45\%$ of the genome can be traced back to transposable elements [15].

Transposons can be classified into two categories according to their replication mechanism: retrotransposable elements (class I) and DNA transposons (class II). Retrotransposable elements, or RNA elements, use a ‘copy-and-paste’ mechanism [55–57] that relies on RNA intermediates. The transposon is transcribed into RNA and then reverse-transcribed into cDNA, which is inserted back into the genome. DNA transposons use instead a ‘cut-and-copy’ mechanism [58] that relies on single or double-stranded DNA intermediates. Therefore, transposons are excised from their current location and integrated somewhere else.

The ‘copy-and-paste’ mechanism employed by retrotransposons means many copies can be produced, making them highly repetitive elements. While 45% of the human genome can be traced back to transposon elements [15], 43% of it is solely due to retrotransposon activity.

The high percentage of transposable elements is not unique to humans. Vertebrate genomes are generally known to contain a high percentage of transposable elements, most of which are retrotransposons [15, 59, 60]. However, the proportion of retrotransposons within genomes is not constant and varies even between species of the same clade [56].

Repetitive elements have played a major role throughout evolutionary history. They are associated with significant changes in genome structure from insertions and deletions to genome rearrangements [59, 61], as well as epigenetic mechanisms from methylation [59, 62] to gene expression regulation [59, 63]. Transposons are also associated with conserved mammalian-specific non-exonic sequences [56], and may have contributed to shaping mammalian development processes during embryonic stages [64]. In the human genome, approximately 4% of genes have transposon-derived sequences in the protein-coding regions while 25% of promoters contain transposon-derived elements [56].

Evolutionarily speaking, transposons confer flexibility in acquiring new genomic features, from affecting gene expression mechanisms to the acquisition of novel gene functions. However, this is a double-edged sword. Because transposons can be inserted almost any-

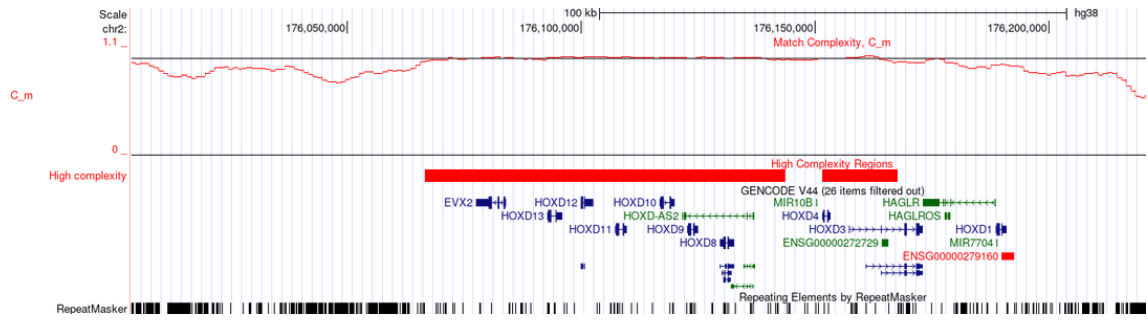


Figure 2.1: *HoxD* cluster on the UCSC Genome Browser of the human genome (GRCh38/hg38) together with the *RepeatMasker* track, in black, showing the location of all repeats, and the high complexity track, in red, showing the repetitiveness of the region. The high complexity intervals span a large portion of the *HoxD* cluster. The interval displayed is Chromosome 2:176,005,710-176,221,466.

where, as they become more frequent in the host organism, the odds that they may disrupt an essential function grows.

Consequently, functional regions sensitive to variation contain fewer recent transposons when compared to the rest of the genome. Thus, transposon-free regions are associated with developmental genes and transcription factors [15–17]. In mammals, the *Hox* clusters are an example of this, as they contain fewer transposon insertions than the rest of the genome [65].

Figure 2.1 shows the current annotation of the human *HoxD* cluster as shown on the UCSC genome browser. The cluster itself is approximately 100 Kb long and is densely packed with transcription factors starting at the 5' end with the even-skipped homeobox 2 gene, *EVX2*, followed by the nine *HoxD* genes. These transcription factors regulate body plan formation in vertebrates [66, Chapter 21]. Apart from the *HoxD* cluster, Figure 2.1 also shows the flanking regions devoid of genes, and, in black, a track of repetitive elements discovered using the program *RepeatMasker*, a well-known program for identifying and masking known interspersed repeats and low-complexity DNA sequences [67]. It is easy to see that the density of repetitive elements is much lower inside the *HoxD* cluster than outside. Complementary to the *RepeatMasker* track is the “High complexity” track in red. The two high-complexity elements span a large portion of the *HoxD* cluster. These elements correspond to the regions where the complexity line (C_m) is at least equal to the threshold (black horizontal line). The complete “High complexity” browser track can be accessed at guanine.evolbio.mpg.de/complexity.

This association suggests that transposon-free regions in a genome are functionally relevant. There are several methods for identifying transposable elements [68]. One of them

is through the use of *RepeatMasker*, as shown above in the UCSC genome browser track of the human genome in Figure 2.1. *RepeatMasker* relies on external databases of known repetitive elements, like *Dfam* [69]. Because *RepeatMasker* uses knowledge from previous studies, it's considered an *a posteriori* approach, where only repeats that have been studied can be detected. While repeats have been extensively cataloged in a few species like humans and mice, the list of known transposons for non-model organisms are not as comprehensive and may introduce bias when searching for function by transposon absence. Recent studies have attempted to close this bias by performing extensive *de novo* annotation for transposable elements across a wide range of mammalian species [70]. While the results from this study were included in the *Dfam* database, the issue remains that not all species are equally represented in the database. Thus, it would be interesting to know if it's possible to search for transposon-free regions based on *a priori* methods. If yes, then it would be possible to extend the search for functionally important regions based on transposon insertions to any organism.

Reversing the statement, one would expect that regions that lack repetitive elements should be functionally relevant. In 2019, Pirogov *et al.* [8] published an *a priori* method to identify globally unique regions in mammalian genomes. They also argued that globally unique regions are transposon-free regions and, hence, functional. Consistent with this argument they observed that the unique regions they identified were highly enriched for developmental genes, concurring with previous studies [16, 17].

2.1.2 Defining highly complex regions

The method devised by Pirogov *et al.* to detect unique regions was implemented in their program *Macle*, which calculates Match Lengths. It starts with a DNA sequence that is decomposed into motifs or match factors, where a factor is a substring that cannot be further extended without including a mismatch. The corresponding match can occur anywhere in the genome.

To calculate match factors, *Macle* relies on an algorithm similar to the Lempel-Ziv decomposition [71], commonly used in “zip” compression programs. However, the Lempel-Ziv algorithm only allows for upstream matches. Because in a biological context, matches can be found anywhere, Pirogov *et al.* implemented a new algorithm (Algorithm 1), where matches can be located both upstream and downstream of a match factor. To do so, they rely on the `isa`, and the `lcp` of a given sequence. The algorithm starts by traversing the sequence, and for each position, obtains the `lcp` at position i and $i + 1$ (lines 6 and 7). The maximal length of both (line 8) is taken as the desired length of the match factor, k .

If no match is found ($\ell_1 = \ell_2 = 0$), k is set to 1 (line 8), and the algorithm skips the match (line 10) and looks for the next match until the entire sequence has been traversed.

Algorithm 1 Match factor decomposition, adapted from Pirogov *et al.*, 2019.

Require: S {input sequence}
Require: isa {inverse suffix array of S }
Require: lcp {longest common prefix array of S }
Ensure: mf {array of match factors (start, length) of S }

- 1: $n \leftarrow |S|$ {length of sequence}
- 2: $\text{lcp}[n + 1] \leftarrow 0$ {prevent out of bounds error}
- 3: $i \leftarrow 1$ {first position in S }
- 4: **while** $i \leq n$ **do**
- 5: $j \leftarrow \text{isa}[i]$ {get position of suffix $S[i\dots]$ in lcp}
- 6: $\ell_1 \leftarrow \text{lcp}[j]$
- 7: $\ell_2 \leftarrow \text{lcp}[j + 1]$
- 8: $k \leftarrow \max(\ell_1, \ell_2, 1)$ {get length of longest match of $S[i\dots]$ or 1, if there is no match}
- 9: mf $\leftarrow \text{append}(\text{mf}, (i, k))$
- 10: $i \leftarrow i + k$
- 11: **end while**

Because DNA sequences have both a forward and a reverse strand, *Macle* concatenates the sequence (S) with its reverse complement before decomposing the forward strand. Considering, for example, a sequence, S , consisting of 20 nucleotides,

$$S = \text{ACCGTATAGTCTTGGTACAT}$$

After applying Algorithm 1, 12 match factors are obtained, here delimited by dots

$$\text{AC.C.GTA.TA.GT.C.T.T.G.GTA.C.AT}$$

The program *lzd*, available at www.github.com/evolbioinf/lzd, can be used to calculate and visualize the decomposition of short sequences.

When a sequence is decomposed, it can be thought of as breaking it down into “words” that appear at least twice. Regions with longer match factors are associated with homologous regions elsewhere in the genome, while shorter match factors that cannot be distinguished from random matches are associated with unique regions.

To decompose a sequence, *Macle* requires, on average ~ 70 MB of memory per Mb and less than 1 s per Mb. For the human genome, this means a memory requirement of roughly

$70 \cdot 3 \cdot 10^9 \cdot 10^{-6} \text{ MB} = 210 \text{ GB}$, and a run time of $3 \cdot 10^9 \cdot 10^{-6} = 50 \text{ min}$. Such resource requirements would make repeated analysis of a given sequence impractical. Fortunately, once decomposed, the match factors can be saved into an index for easy querying. For example, in the case of the human genome, indexing required 195 GB of memory and 3,721 seconds (1 h 02 m) elapsed time, while traversal of the index took just 25.3 GB and 33.9 s elapsed time. Thus, the analysis with *Macle* is split into two stages: first, the sequence is indexed or decomposed and then queried to count the number of observed match factors.

There are two ways to count the number of match factors. Either, by counting the total number of match factors in a sequence within a given region, or by using a sliding window. To quantify match factors in a standardized way, Odenthal-Hesse *et. al* [72] proposed the match complexity, which is given as the ratio between the number of observed match factors (m_o) and the expected number of match factors in a random sequence with the same length and GC content (m_e) (eq. 2.1).

$$C_m = \frac{m_o - 1}{m_e - 1} \quad (2.1)$$

Subtracting 1 from m_o and m_e ensures that C_m has a lower bound of 0 when there is at least one identical copy of the region and an expected upper bound of 1 for regions with repetitiveness similar to what would be expected by chance (eq. 2.2). Regions can have a C_m higher than 1, meaning their match factors are shorter than what would be expected in a random sequence with the same length and GC content.

$$0 \leq C_m \sim 1 \quad (2.2)$$

The null distribution of C_m can be modeled as a normal distribution with mean and variance calculated from the distribution of matches lengths [8] shown in Chapter 1, Figure 1.2. In Figure 2.2A I compare this theoretical distribution to the distribution of C_m values in 10 Kb windows across a random 100 Mb sequence with the GC content of the human genome, 0.4. The theoretical distribution, calculated using *macle2go quantile* [8], closely approximates the simulation and shows a symmetrical distribution tightly centered on the expectation of 1. This is very different from the distribution of C_m values observed in real mammalian genomes. Figure 2.2B shows the C_m values in 10 Kb windows across the human genome, which forms a left-skewed bimodal distribution. Its median complexity is 0.9, but there are also 17 regions with $C_m \sim 0$ (109 Mb) and 22 regions with $C_m \sim 1$

(4 Mb).

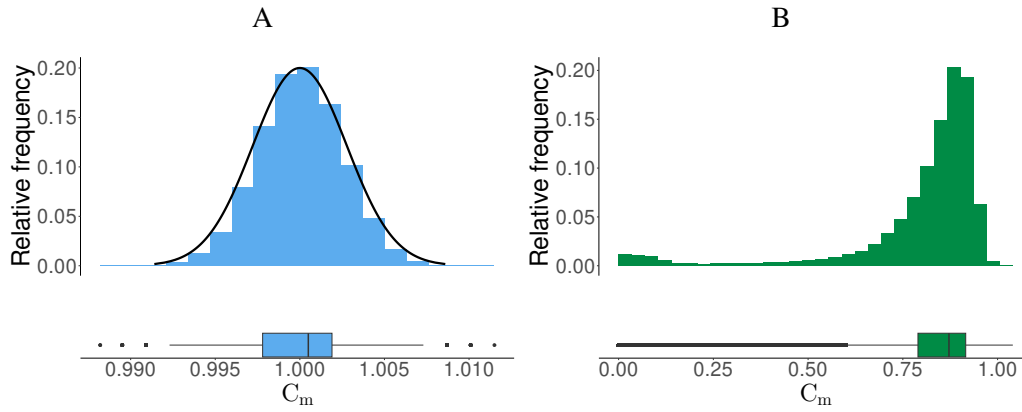


Figure 2.2: Distribution of C_m values using a sliding window of 10 Kb for A) a 100 Mb random sequence, B) the human genome

The threshold, t , beyond which a region is considered to be highly complex, or globally unique, can be determined from the null distribution of C_m (Figure 2.2A). To apply a quantile cut-off of, say, the lower 5%, one would use the null distribution to calculate the expected C_m at the 5% quantile. Any region where $C_m \geq t$ is then classified as highly complex. In Figure 2.1, the *HoxD* cluster intersects a highly complex region, shown in red. The cluster is flanked by regions with a higher number of elements masked by *RepeatMasker* resulting in a lower complexity score.

2.1.3 On the quality of genome assemblies

Until quite recently, non-model organisms had relatively fragmented assemblies, mainly composed of scaffolds and contigs. Although there has been a massive increase in the number of available mammalian genomes, these tend to be of variable quality. In April 2023, it was reported that although there were a total of 657 mammalian available genomes, only 76 had been assembled at the chromosome level [73]. This is because repetitive regions are more challenging to assemble and can collapse into a single copy during the assembly step. Thus, these repetitive regions could be erroneously considered highly complex in a poorly assembled genome. Recent advances in long-read sequencing technologies have made the routine construction of high-quality chromosomal references feasible [74].

Improved assemblies and high-quality reference genomes support advances in medical research. For example, an improved pig genome assembly [75] allowed for a better annotation of immune-related genes, which are essential for immunology studies [76]. Improved

assemblies can also lead to higher precision in gene editing, helping the development of multigene knockouts in pigs [76].

The importance of high-quality reference genomes does not only apply to organisms that are of interest in medicine or agriculture but are also a much welcome addition to comparative genomics [74]. In Chapter 1 I introduced the BUSCO metric which assesses out of a known set of orthologs how many are present in a given assembly. While a high BUSCO score indicates that most expected gene orthologs have been correctly assembled, this gives no indication regarding gene function [77].

2.1.4 Picking mammalian genomes

In the empirical part of their study, Pirogov *et al.* looked at highly complex regions in both the human and mouse genomes, two well-studied species with high-quality genome sequences and annotations. Their original hypothesis that highly complex regions were associated with early-acting, i.e. developmental genes, turned out to be correct, but the underlying assumption that these regions were also equally transposon-free was not tested.

Additionally, the genomes used were released in 2014 and are no longer up-to-date. Since then, the human reference genome has undergone more than 30 smaller correction patches and the mouse genome had a major update of the coordinates system with the release of the GRCm39 reference assembly [78] in 2020. Patches tend to incremental changes. For example, despite undergoing 30 correction patches, the human reference genome has the same N50 at both scaffold and contig level (67.8 Mb and 57.9 Mb, respectively), and the same number of gaps between scaffolds (349). The only difference is that one more scaffold was placed, decreasing the number of unplaced scaffolds from 127 to 126. In contrast, new assembly releases tend to be quite distinct. The latest release of the mouse genome increased the N50 at both scaffold and contig level (from 54.5 Mb to 106.1 Mb, and from 31.8 Mb to 59.5 Mb, respectively), decreased the number of gaps between scaffolds from 191 to 143, and halved the number of contigs.

Newer releases also contain information regarding new completeness scores that might have been developed since. For example, the GRCh38.40 human assembly has an associated completeness BUSCO score of 99.2%, while the GRCm39 mouse assembly has a completeness score of 99.5%.

Finally, Pirogov *et al.* relied on high-quality functional annotation data for both the mouse and human genomes. Unfortunately, many high-quality reference genomes come with only limited functional annotation, if any. As of summer 2023, there were 118 published

reference-level and annotated mammalian genomes in the NCBI database but only 10 of them are functionally annotated.

One way to extend the complexity analysis of Pirogov *et al.* to organisms lacking the extensive annotation data of human and mouse might be to rely on orthologous gene information to map gene functionality across organisms. One such ortholog-based annotation system is implemented in the *eggNOG-mapper* package [79, 80].

2.1.5 Chapter aims

In this chapter, I explain how to identify and annotate highly complex regions in 18 mammalian genomes distributed across nine mammalian orders. I also investigate the assumption that highly complex regions in mammalian genomes are transposon-free.

2.2 Methods

2.2.1 Data

The list of available mammalian genomes was obtained on 1st August 2023 from the NCBI database [81, 82] using their command-line tools *datasets* (v15.12) [83, 84]. Figure 2.3 shows the *datasets* command used where accession 40674 is the taxon ID of Mammalia, and the *released-before* flag reproduces my genome list. This list contains 118 reference genomes that were grouped according to their taxonomy. I selected mammalian orders that were represented by at least two different species. This resulted in a sample covering nine, out of 18, mammalian orders with a total of 78 reference genomes. The nine represented orders are the Artiodactyla (even-toed ungulates), Carnivora, Chiroptera (bats), Dasyuromorphia (carnivore marsupials), Didelphimorphia (marsupials), Eulipotyphla (shrews), Perissodactyla (odd-toed ungulates), Primates and Rodentia. For each order, I selected two genomes, by considering the completeness of their assembly as evaluated by their BUSCO score. The 18 selected organisms and their corresponding accessions can be seen in Table 2.1.

Additionally, for each genome, I also obtained the corresponding annotations in GFF3 format and the protein sequences to annotate the underlying genes through protein-based orthology searches. The BUSCO completeness scores range between 90.4% for the agile gracile mouse opossum and 99.5% for the house mouse.

The obtained human genome has the accession GCF_000001405.40 which corresponds

```

datasets download taxon accession 40674
--reference --annotated
--assembly-level chromosome,complete
--assembly-source refseq
--include genome,gff3,protein
--exclude-atypical
--released-before 08/01/2023

```

Figure 2.3: Datasets command for obtaining the list of mammalian genomes used in this Chapter.

to the GRCh38.p14 release from the Genome Research Consortium [85]. Likewise, the mouse genome with the accession GCF_000001635.27 corresponds to the GRCm39 assembly release [78].

For the Chiroptera, Dasyuromorphia and Eulipotyphla, only two annotated reference genomes were available. For the Carnivora, the dog, which is functionally annotated, and the feline with the highest BUSCO completeness score (C:99.0%) was selected. For the Perissodactyla, both the horse and the plains zebra were selected, (BUSCO C:99.0% and C:98.4%, respectively). Finally, the chimpanzee reference genome (BUSCO C:99.1%) was chosen as the second primate to complement the results of the humans (BUSCO C:99.2%), Table 2.1. Respective accessions and BUSCO completeness scores for each species can be found in Table 2.1.

2.2.2 Identification of transposons in highly complex regions

The identification of repeats in the human genome was performed with *RepeatMasker* (v4.1.5) and *rmblast* (v2.14.0+), using CONS-Dfam (v3.7) as the repeat library. *RepeatMasker* was run with ten parallel threads.

To identify the maximum mutation rate compatible with repeat detection by *Macle* or *RepeatMasker*, I simulated a random sequence of 25 Kb and inserted two identical copies of the MER5A1r transposon, which is 3.7 Kb long (accession DF0290013.2). The two copies were placed 10 Kb away from the sequence ends with a 5 Kb distance between them. The constructed sequence was mutated using the program *mutator* [12] at rates ranging in steps of 0.01 from 0 up to 1. Then, I obtained the proportion of nucleotides masked by *RepeatMasker* for each mutation rate. To calculate the complexity, I performed a sliding window analysis with a window of size 1 Kb. Windows with a complexity of at least 0.98 were extracted and merged. I then considered their complement, the low-complexity regions, and calculated the percentage of non-complex nucleotides that

Table 2.1: Genome accessions, corresponding mammalian order for each species, total genome size (Gb) and corresponding BUSCO completeness score (%) of the 18 mammals surveyed in this chapter.

Accession	Order	Species	Common name	Size (Gb)	BUSCO (%)
GCF_002263795.2	Artiodactyla	<i>Bos taurus</i>	Bovine	2.6	98.8
GCF_000003025.6		<i>Sus scrofa</i>	Pig	2.4	98.2
GCF_011100685.1	Carnivora	<i>Canis lupus familiaris</i>	Domestic dog	2.4	98.1
GCF_016509475.1		<i>Prionailurus bengalensis</i>	Leopard cat	2.4	99.0
GCF_004126475.2	Chiroptera	<i>Phyllostomus discolor</i>	Pale spear-nosed bat	2.1	98.0
GCF_022682495.1		<i>Desmodus rotundus</i>	Common vampire bat	2.0	98.1
GCF_016432865.1	Dasyuromorphia	<i>Antechinus flavipes</i>	Yellow-footed antechinus	3.2	95.4
GCF_902635505.1		<i>Sarcophilus harrisii</i>	Tasmanian devil	3.2	95.5
GCF_000002295.2	Didelphimorphia	<i>Monodelphis domestica</i>	Gray short-tailed opossum	3.5	95.1
GCF_016433145.1		<i>Gracilinanus agilis</i>	Agile gracile mouse opossum	3.4	90.4
GCF_024139225.1	Eulipotyphla	<i>Suncus etruscus</i>	White-toothed pygmy shrew	2.5	94.9
GCF_027595985.1		<i>Sorex araneus</i>	European shrew	2.4	96.8
GCF_002863925.1	Perissodactyla	<i>Equus caballus</i>	Horse	2.4	99.0
GCF_021613505.1		<i>Equus quagga</i>	Plains zebra	2.3	98.4
GCF_000001405.40	Primates	<i>Homo sapiens</i>	Human	3.1	99.2
GCF_028858775.1		<i>Pan troglodytes</i>	Chimpanzee	3.1	99.1
GCF_000001635.27	Rodentia	<i>Mus musculus</i>	House mouse	2.7	99.5
GCF_015227675.2		<i>Rattus norvegicus</i>	Norwegian rat	2.6	99.3

intersected a masked region.

2.2.3 Phylogenetic tree

The phylogeny of the 18 mammalian species was reconstructed based on their mitochondrial sequences (Table 2.2). First, the sequences were aligned using *MAFFT* (v7.520) [86, 87]. Then, the best substitution rate model was estimated with *ModelFinder* [88] to be “GTR+F+I+G4”. Finally, I constructed the tree with *iqtree2* [89] using 1000 ultra-fast bootstraps [90]. The resulting tree was rooted using the minimal ancestor deviation method (MAD) [91] with *mad* (v2.2) and visualized with *ggtree* [92].

Table 2.2: Mitochondrial accessions used in the phylogenetic analysis.

Species	Common name	Accession	Length (Kb)
<i>Bos taurus</i>	Bovine	NC_006853.1	16.3
<i>Sus scrofa</i>	Pig	NC_012095.1	16.8
<i>Canis lupus familiaris</i>	Domestic dog	NC_002008.4	16.7
<i>Prionailurus bengalensis</i>	Leopard cat	NC_028301.1	16.7
<i>Phyllostomus discolor</i>	Pale spear-nosed bat	NC_065690.1	16.7
<i>Desmodus rotundus</i>	Common vampire bat	NC_022423.1	16.7
<i>Antechinus flavipes</i>	Yellow-footed antechinus	NC_056379.1	17.4
<i>Sarcophilus harrisii</i>	Tasmanian devil	NC_018788.1	17.1
<i>Monodelphis domestica</i>	Gray short-tailed opossum	NC_006299.1	17.1
<i>Gracilinanus agilis</i>	Agile gracile mouse opossum	NC_054268.1	16.3
<i>Suncus etruscus</i> ^a	white-toothed pigmy shrew	CM044019.1	17.0
<i>Sorex araneus</i>	European shrew	NC_027963.1	16.9
<i>Equus caballus</i>	Horse	NC_001640.1	16.7
<i>Equus quagga</i>	Plains zebra	NC_044858.1	16.6
<i>Homo sapiens</i>	Human	NC_012920.1	16.6
<i>Pan troglodytes</i>	Chimpanzee	NC_001643.1	16.6
<i>Mus musculus</i>	House mouse	NC_005089.1	16.3
<i>Rattus norvegicus</i>	Norwegian rat	NC_001665.2	16.3

^a Non-reference level

2.2.4 Detecting complex regions

I started the complexity analysis by identifying highly complex regions in the 18 mammalian genomes. Using *Macle* (v0.1), I calculated an index file for each genome and the local complexity with a sliding window analysis. To test the effect of the size of the sliding window, I varied the window size from 1 Kb to 50 Kb.

The cut-off point for defining highly complex regions was obtained as a function of genome length and GC content with *macle2go quantile* [8]. Genome length and GC content were extracted from the Macle index. I chose a C_m cut-off point corresponding to the top 95% most complex regions of the null distribution. Regions with C_m greater or equal to this threshold were extracted and overlapping regions were merged.

2.2.5 Annotation and enrichment analysis

To perform an enrichment analysis on the obtained highly complex regions, it is necessary to map gene names to their function. Unfortunately, such maps are only available for 6 of the selected genomes.

Currently, there are methods to overcome this limitation [93–96]. Some rely on inferring gene function through general homology, that is by comparing genes in different species that have the same origin. This can be done, for example, through homology searches. However, while easy to perform, homology methods do not take into account that ancestral functions are more likely to be retained between orthologous genes than between paralogs [97]. Thus, a better approach is to transfer gene functionality through orthology-based methods.

Here I present a pipeline to annotate and transfer gene functionality through orthology by relying on *eggNOG-mapper* (v2.1.12) [51, 79], an orthology-based method. To make the analysis comparable across all 18 species, I assigned gene function through orthology to all 18 species regardless of whether this information was already available or not. For this I used *eggNOG-mapper* together with the *eggNOG* database [80]. An example of an *eggNOG-mapper* command can be seen in Figure 2.4.

```
emapper.py --cpu 50 -o mouse --output_dir Results/
--override -m diamond --sensmode ultra-sensitive
--dmnd_ignore_warnings -i mouse_proteome.faa
--evaluate 0.001 --score 60 --pident 40 --query_cover 20
--subject_cover 20 --itype proteins
--tax_scope inner_narrowest --target_orthologs all
--go_evidence non-electronic --dbmem
--tax_scope eukaryota --pfam_realign none
--report_orthologs --decorate_gff yes
```

Figure 2.4: Example of an *eggNOG-mapper* command

The mapping was performed between the *eggNOG* protein database and the corresponding proteome for each species with *diamond* (v 2.0.11) [98]. The analysis was run with

50 threads, in the ultra-sensitive mode. The default query parameters for *E*-value, score, percentage identity (pident), query and subject coverage were kept. The annotation was restricted to eukaryotic genes, where the narrowest clade was favored. Finally, orthology was searched allowing for a many-to-many comparisons. Only non-electronically curated GO terms were considered.

The output file from *eggNOG-mapper* contains a list of all the GO terms associated with each protein. From that, I extracted the list of protein-GO terms association, and mapped each GO ID with the corresponding description, and each protein ID with the gene ID. With this information, I constructed an abridged gene2go file containing all the information required for performing an enrichment analysis, Figure 2.5.

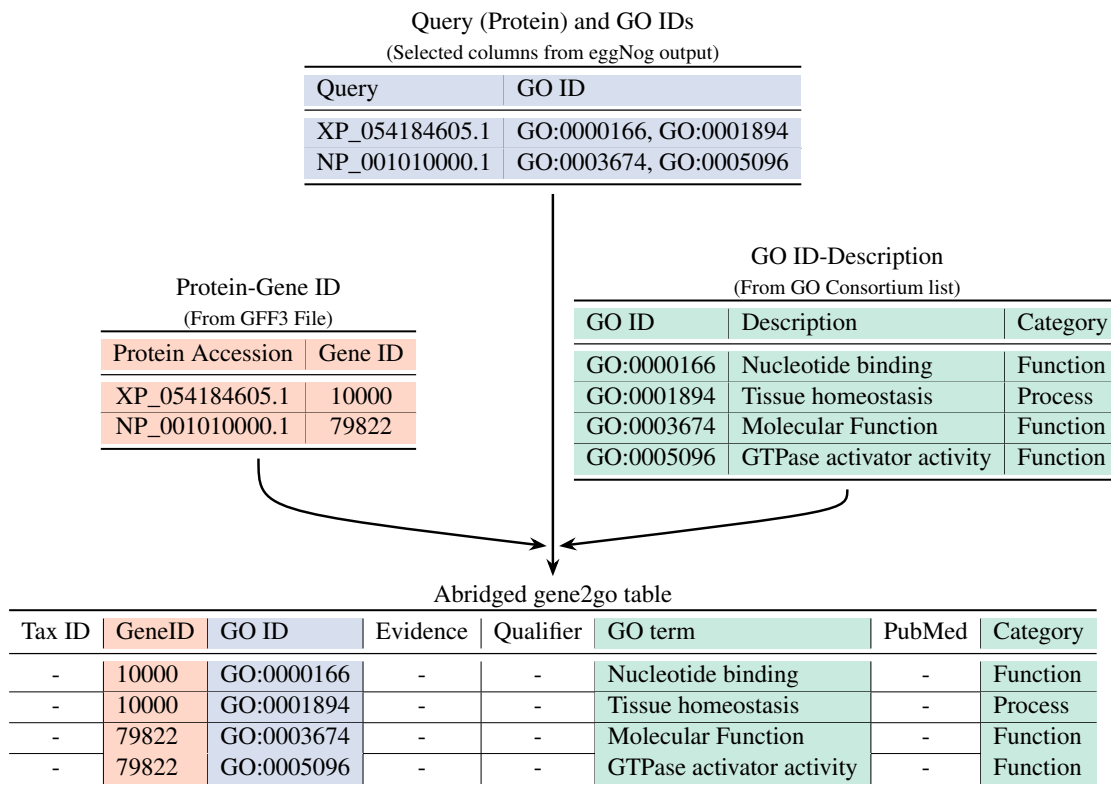


Figure 2.5: Schematic representation of the three tables joined to form an abridged gene2go table.

The list of GO IDs, and associated descriptions was obtained from the GO Ontology website, <http://purl.obolibrary.org/obo/go/go-basic.obo>, accessed 12th June 2024. The list was converted into a table format such that every row is composed of three columns containing the GO ID, its description and corresponding category.

Using such gene2go files the annotation and enrichment analysis of the highly complex

regions was performed with the four programs of the *Gin* package [99]: *annotate* (v1.0-1), *shuffle* (v1.0-1), *ego* (v1.0-1) and *sumEgo* (v1.0-1).

The annotation was performed by intersecting the corresponding GFF3 file with the highly complex intervals in the genome. From the *annotate* program, I obtained the identifiers of all genes whose promoter (interval of 4 Kb centered on the transcription start site) intersects the complex regions. Then, each gene identifier was associated with one or more GO categories using *ego* and the constructed abridged gene2go file. The null hypothesis of no enrichment was tested with a Monte Carlo approach for each GO category occupied by at least ten genes. The high-complexity regions were shuffled 10^5 times across the genome with *shuffle*. Each set of shuffled regions was annotated with *annotate* and passed to *ego* to calculate the frequency of finding at least as many genes in a given GO category as observed originally, the desired *P*-values of that category.

To speed up the Monte Carlo test, the 10^5 iterations were split into 20 jobs of 5×10^3 iterations each, and sent to the background. Once the 20 jobs were all finished (*wait*), their results were summarized with *sumEgo*. The workflow from starting the annotation until obtaining the list of described GO terms is exemplified by the command in Figure 2.6, where *annotate* gives the list of observed genes, and the loop shuffles, annotates and compares the expected frequency of obtaining genes from each GO term in randomly picked regions. The file *genomic.gff* contains the genome annotation, *iv.txt* is the list of intervals of highly complex regions, and *obsId.txt* is the list of observed annotated promoters in the highly complex regions.

```

annotate -c genomic.gff iv.txt > obsId.txt

for a in $(seq 20)
do
    nohup shuffle -n 5000 genomic.gff iv.txt |
    annotate -c genomic.gff |
    ego -o Abridged.gene2go obsId.txt > ego_${a}.txt &
done
wait

```

Figure 2.6: Example command of the three *Gin* programs for annotating, shuffling and performing an enrichment analysis on unique regions.

The *P*-values returned by *sumEgo* were corrected for multiple testing using the Bonferroni-correction of multiplying them by the number of tests, which is the number of GO categories with at least ten occupants. Categories with corrected $P \leq 10^{-2}$ were deemed enriched. As an alternative to the Monte-Carlo test, *ego* now also implements a paramet-

ric test based on the assumption that the null distribution of genes per category is normal. While this may or may not be true, the parametric test is much faster than the Monte Carlo test.

Additionally, I also performed a more extensive enrichment analysis with 10^6 iterations across all 18 species for the highly complex regions detected with a 10 Kb window. As described above, only regions with a corrected $P \leq 10^{-2}$ were deemed enriched.

2.2.6 Hardware/Software

Timed runs of *RepeatMasker* and *Macle* were performed on a server running Ubuntu 22.04.04. The server contains 16 Intel CPU cores (Xeon 2.10GHz) with 64 threads and 264 GB of RAM. The runtime and memory required for running each process was measured with the utility program *time*. This was called using `/usr/bin/time` to distinguish it from the *bash* command *time*.

2.2.7 Data availability

Both the Macle indexes and the gene2go files of the 18 genomes constructed during the analysis of this chapter are available at the edmond repository <https://doi.org/10.17617/3.4IKQAG>. Additionally, the repository *Auger*, <https://github.com/EvolBioInf/auger>, contains supplementary information for reproducing the abridged gene2go files.

2.3 Results

2.3.1 Phylogenetic tree

Figure 2.7 shows the phylogeny of the 18 mammals studied in this chapter. The early split between marsupials (Didelphimorphia and Dasyuromorphia) from the remaining placental species can be clearly seen. Beyond the grouping of the mammalian orders, the Laurasiatheria, which is composed by Chiroptera, Artiodactyla, Perissodactyla, Carnivora and Eulipotyphla can be seen.

According to the literature [100, 101], it was also expected to see the Euarchontoglires superorder formed by the Rodentia and Primates. However, in Figure 2.7 these mammalian orders are not monophyletic. While this disagrees with the established phylogeny of mammals [102], it does agree with previous studies based on mitochondrial DNA [103].

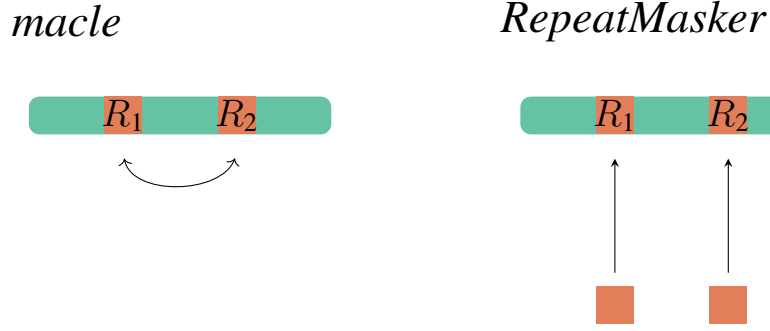


Figure 2.8: Schematic representation of the differences between *Macle* and *RepeatMasker*. *Macle* compares two copies between themselves, while *RepeatMasker* compares each copy with the database record.

lion years ago [104, 105]. Assuming a mutation rate of $0.5 \times 10^{-9} \text{bp}^{-1} \text{year}^{-1}$ [106], this amounts to a distance between the *Hox* clusters of $2 \cdot 500 \cdot 10^6 \cdot \frac{10^{-9}}{2} = 0.5$ mutations per site. This is just at the detection limit of *RepeatMasker*, though it wouldn't flag the *Hox* clusters as they are not transposable elements. The point is, however, that 0.5 mutations/site is more than twice *Macle*'s detection limit of 0.2 mutations/site and hence *Macle* classifies old paralogs like the *Hox* clusters as unique.

I also checked whether this detection limit of 0.2 mutations/site holds true with real data. I started by masking the human genome with *RepeatMasker* and extracting highly complex regions for a sliding window of 1 Kb from *Macle*. *RepeatMasker* masked 1.52 Gb (49.1%) of the human genome. Of those, 92.0% are classified as interspersed repeats (SINEs, 25.5%, LINEs, 41.3% and LTRs, 17.7%). The remaining regions are classified as satellites (5.2%), simple repeats (3.9%), low complexity regions (0.4%), or small RNAs (0.1%). In turn, *Macle* calculates that 15.2%, 468,63 Mb, of the human genome is highly complex. So at most one-third of the repeat-free regions detected by *RepeatMasker* are classified as unique by *Macle* with a 1 Kb window. One explanation for this is that *RepeatMasker* detects known repetitive elements, while *Macle* is sensitive to repeats of any kind.

I now investigate to what extent regions selected by *Macle* are fully transposon-free. 13.5% of the highly complex regions do not intersect any masked region. Of those that do, they overlap masked regions, on average, by only 148.5 bp which is 9.4% of the complex regions (Figure 2.10A). The average divergence rate of the intersected masked regions is 24.7%, more than the 0.2 estimated to be the limit of *Macle*, so the simulations in Figure 2.9 are consistent with the observations. There are, however, some masked regions with a lower divergence level, but these tend to have an even smaller overlap of, on average,

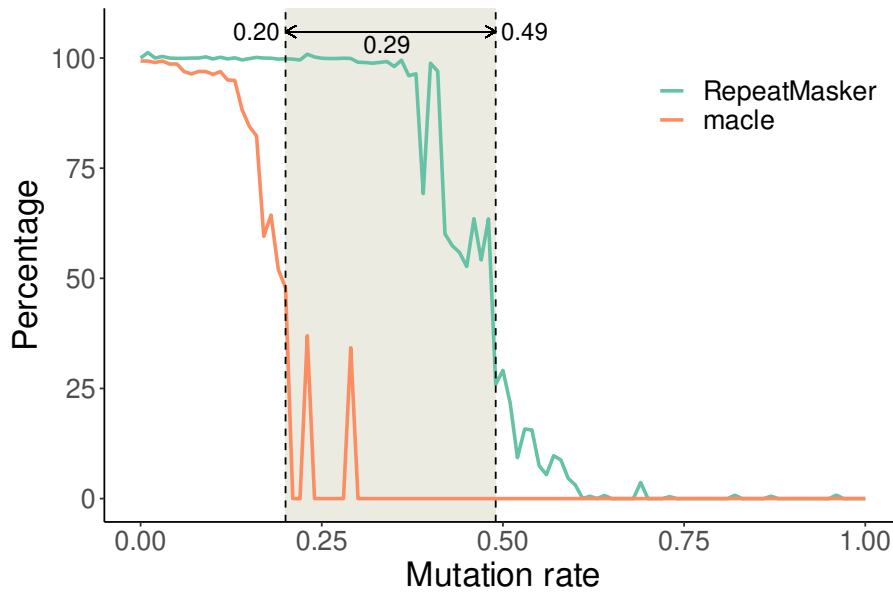


Figure 2.9: Detecting the two identical repeats. Percentage of nucleotides masked by *Repeat Masker*, and percentage of non-complex nucleotides calculated by *Macle*, for a given mutation rate. Vertical lines denote the mutation rate where the percentage first fell below 50%.

55.3 bp (Figure 2.10B).

There is a single complex region, however, that fully encompasses a masked region, sharing a total of 7.2 Kb. This long overlap occurs on the X chromosome, with a human endogenous virus (HERV-Fc2) that has diverged by 16.9% from the consensus sequence. Retroviruses consist of three genes, *gag*, *pol* and *env* that are flanked by two LTRs. In this particular case, the intersection occurs within one of the four enzymes encoded by the *pol* polyprotein, the integrase *int* [107]. Integrases are responsible for integrating the DNA produced by the reverse transcriptase into the host's genome [108].

2.3.3 Identification of highly complex regions

I start by making two general observations concerning *Macle* and window size. First, the size of the applied sliding window constrains the length of the smallest detectable complex region. Generally speaking, highly-complex regions are longer than the applied window size. However, for a shorter window of 1 Kb it was possible to detect smaller regions in half of the genome accessions. As a rule of thumb, only complex regions that are at least as long as the applied window size can be detected. Therefore, if searching for complex regions that are at least 1 Kb long, the applied window size should be smaller or of equal size. In Figure 2.11A, it can be seen that the proportion of the genome for

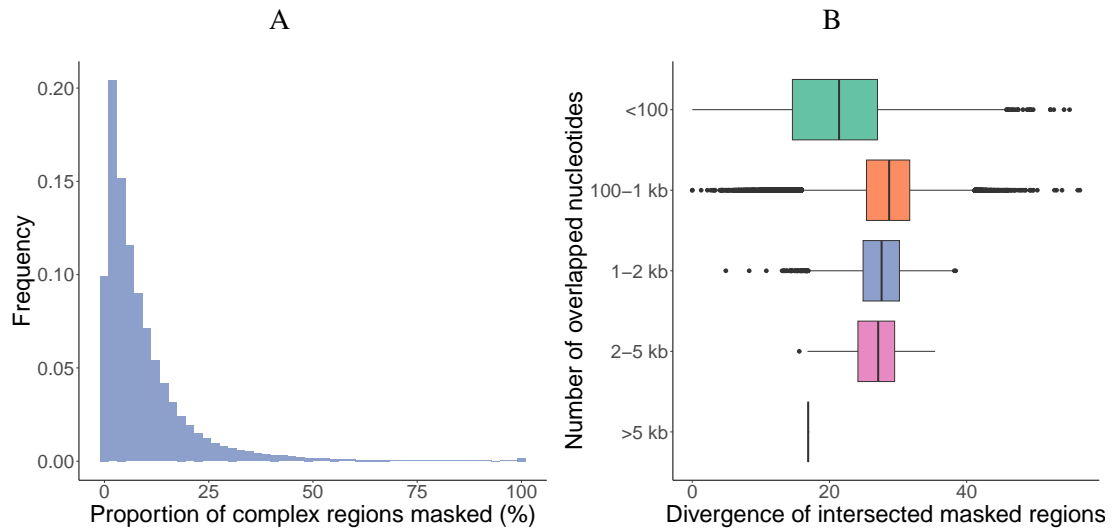


Figure 2.10: Analysis of the intersection between highly complex regions picked by *Macle* and repeats masked by *RepeatMasker*. A) Proportion of highly complex regions covered by masked repeats. B) Distribution of the divergence rate of masked overlaps, according to the total length of the overlaps.

a given species that is considered to be highly complex varies according to the window length. For most of the organisms, for a shorter window of 1 Kb, the proportion of the genome that is considered to be highly complex varies between 23.4% for the bovine and 34.4% for the leopard cat. However, for the shrews (Eulipotyphla), primates and marsupials (Didelphimorphia) this proportion is much lower varying between 10.8% (european shrew) and 16.1% (white-toothed pygmy shrew).

Secondly, as windows get larger, the proportion of the genome that is classified as highly complex decreases. For a 10 Kb window, the proportion of highly complex regions in the genome varies between less than 0.04% for the european shrew and 2.9% for the tasmanian devil. Taken to extremes, it is also possible that no complex region is detected at all. For example, for a window of 50 Kb, the european shrew had no highly complex region, and on average, only 0.09% of the genome was considered complex across all species.

As the proportion of the genome that is considered to be highly-complex decreases, so does the number of promoters that intersect those regions (Figure 2.11B). For a window of 1 Kb more than 50.9% of the promoters for a given species are located within highly-complex regions. For seven species, the Dasyuromorphia, the leopard cat, the common vampire bat, the pig and the Perissodactyla, more than 75% of all promoters are located within these regions. The tasmanian devil (Dasyuromorphia) contains the highest pro-

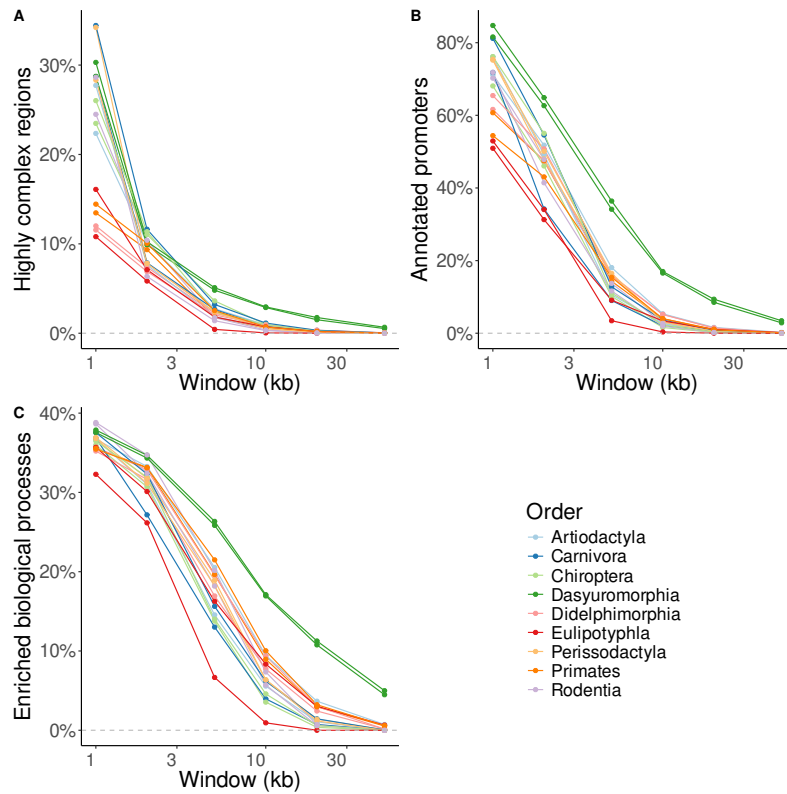


Figure 2.11: Analysis of identified highly complex regions per genome per window size. A) Total length; B) Annotated promoters; C) Enriched biological processes.

portion of promoters that intersect highly complex regions, 84.7%. In contrast, for the white-toothed pygmy shrew only 50.9% of the promoters intersect highly complex regions. The proportion of annotated promoters decreases with increasing window length. With a window length of 5 Kb, less than 36.4% of the promoters were annotated. For a window of 10 Kb only 4.7% of all promoters are annotated on average, ranging from 0.4% for the european shrew to 17.0% for the tasmanian devil.

Likewise, the number of enriched GO terms decreases linearly with the increase in window size. As shown in Figure 2.11C, for a 1 Kb window the number of enriched GO terms varies between 32.3% of all GO terms, for the european shrew, and 38.8% for the mouse house. For a window of 5 Kb, the proportion of enriched GO terms found is less than 25%, with the exception of the Dasyuromorphia with 25.8% (yellow-footed antechinus) and 26.3% (tasmanian devil).

More informative than the total number of enriched processes, is their enrichment ratio. Figure 2.12 shows the average enrichment ratio per organism per window. For most organisms the global enrichment ratio tends to increase with the increase in window length between 1 Kb to 10 Kb. Afterward, the global enrichment level for both marsupial orders

plateaus, while for the other orders the enrichment ratio starts to decrease. The Eulipotyphla and the Primates are an exception, where the enrichment level keeps increasing even for larger window sizes.

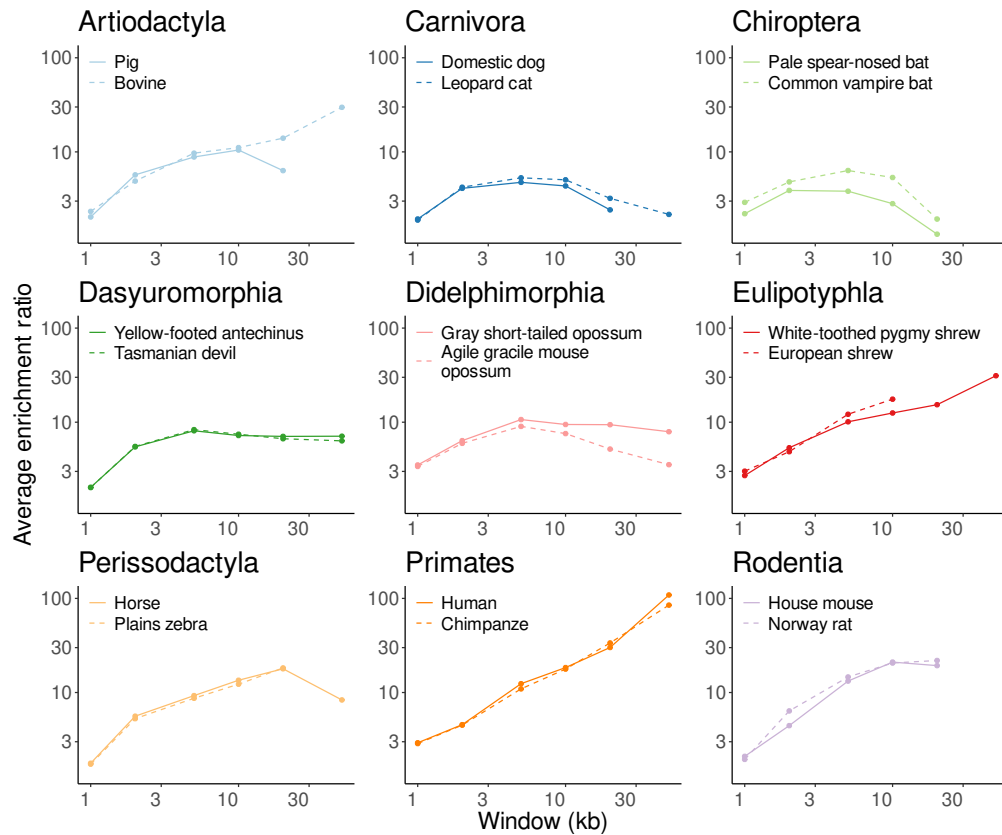


Figure 2.12: Obtained enrichment ratio per genome per window size. Quantification of identified highly complex regions per genome per window size.

I summarize the results from the more extensive enrichment analysis for a 10 Kb window by looking at processes whose enrichment had been detected across all 18 species, resulting in a list of 124 terms. I calculated the mean enrichment ratio across all 18 species for all of the 124 enriched terms and show the top 20 most globally enriched terms in Figure 2.13. The figure shows, ordered by average enrichment ratio, the obtained enrichment ratio per term per species. The enrichment ratio ranges between 3.7 and 89.1. It can be seen that these processes are highly associated with developmental genes, with several of them related to embryonic development. The top seven most globally enriched biological processes are all associated with pattern specification and organ development during the embryonic stages. Additionally, there is enrichment in the development of the head, brain, central nervous system and neuron differentiation.

The top three highest enrichment ratios all occur in the shrew and are “anterior/posterior

pattern specification” (89.1), “embryonic organ development” (65.9) and “regionalization” (62.0). In contrast, the three least enrichment processes occur in the pale spear-nosed bat and are “anatomical structure formation involved in morphogenesis” (3.7), “negative regulation of transcription by RNA polymerase II’q” (3.8) and “head development”(4.0).

When it comes to globally enriched biological terms, the Rodentia have the highest overall enrichment ratio (33.9), followed by the Primates (30.6) and the Eulipotyphla (30.2). On the other hand, the Chiroptera (6.7), Carnivora (7.8) and Dasyuromorphia (9.9) have the lowest global enrichment ratios.

After correcting for multiple testing, the P -values for each of the terms shown in Figure 2.12 vary between 3×10^{-4} and 2×10^{-3} as can be seen in Figure 2.14.

The abridged gene2go files constructed with the aid of *eggNOG-mapper* contained 22,423 distinct GO terms for each species. After performing the enrichment analysis, I extracted a total of 241,273 entries describing enriched processes across the 18 species and window sizes. Of those, only 6,145 GO terms were distinct. The exact P -value could only be calculated for 276 of these entries due to the limited number of Monte Carlo iterations. All P -values presented in Figure 2.14 are uncertain and must all be read as *less than*.

To further explore the P -values of the enrichment study, let’s consider the null distribution of the expected number of genes associated with a given term. The most globally enriched GO term was “anterior/posterior pattern specification” (GO:0009952), while the lowest of the top 20 was “anatomical structure formation involved in morphogenesis” (GO:0048646), as seen in Figure 2.13 for a 10 Kb window. These two terms had 88 and 176 observed genes in the human highly complex regions, respectively. Randomly shuffling these regions across the genome 10^6 times, results, on average, in 1.7 and 6.9 genes associated with these terms, respectively. After correcting for multiple testing, the enrichment ratio has a P -value of $< 1.7 \times 10^{-3}$ for both terms, meaning these two terms are significantly enriched. However, the P -value is deceptively large due to the limited number of iterations that are feasible in the Monte Carlo test.

To visualize the difference between the expected and the observed number of genes I show the distribution of the number of genes in shuffled regions in comparison to the observed. For the “anterior/posterior pattern specification”, although 88 genes have been observed within the highly complex regions, on average, only 1.8 genes were found by chance alone, while for the “anatomical structure formation involved in morphogenesis”, of the 176 observed genes, on average only 7.2 genes were found by chance.

Both of the null distributions shown in Figure 2.15 are right-skewed, with the “ante-

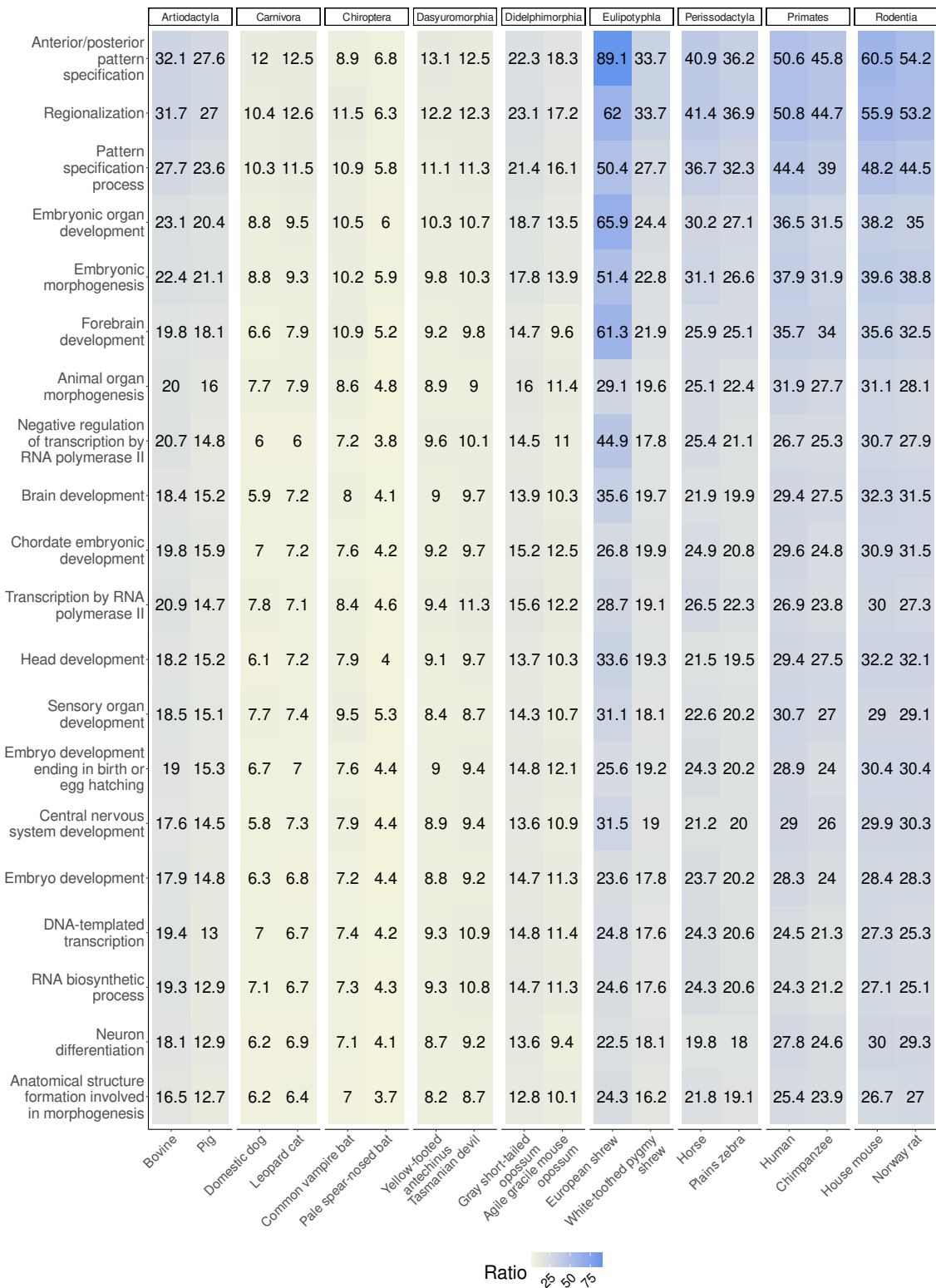


Figure 2.13: Enriched biological processes shared across the 18 species. The top 20 processes with the highest overall enrichment ratio are shown. The respective enrichment ratio for each species for each term is shown. The darker the hue, the higher the enrichment ratio.

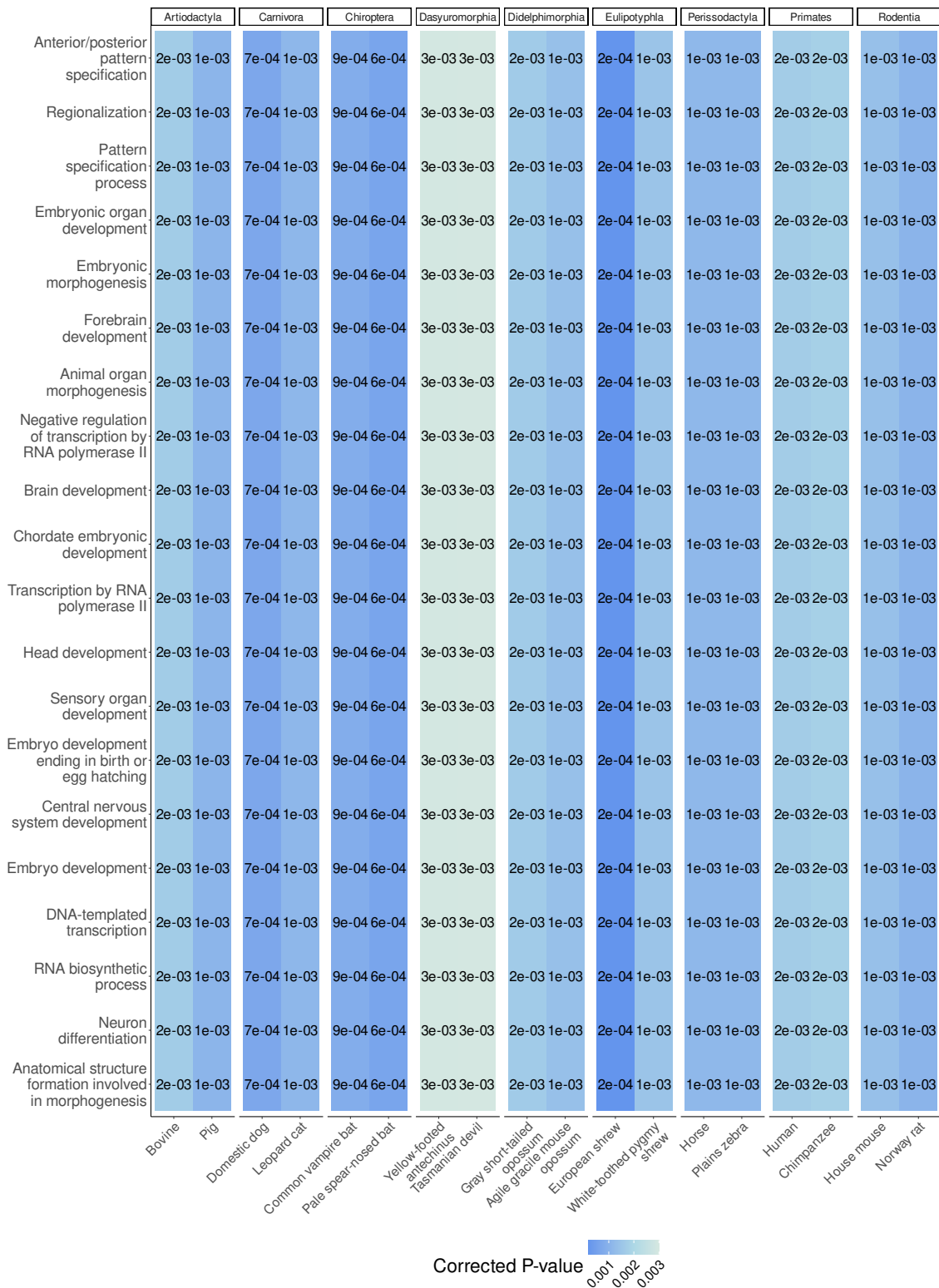


Figure 2.14: Enriched biological processes shared across the 18 species. The top 20 processes with the highest overall enrichment ratio are shown. The respective corrected P -value for each species for each term is shown. All P -values should be read as less than. The darker the hue, the lower the corrected P -value.

rior/posterior patterns specification” having a skewness of 1 and the “anatomical structure formation involved in morphogenesis” a skewness of 0.4. Meaning that the first distribution is on the verge of being highly skewed. Although right skewness means that there is a long tail on the right side of the distribution, there is still a large distance between the maximum of the null distribution and the number of genes observed within the highly complex regions.

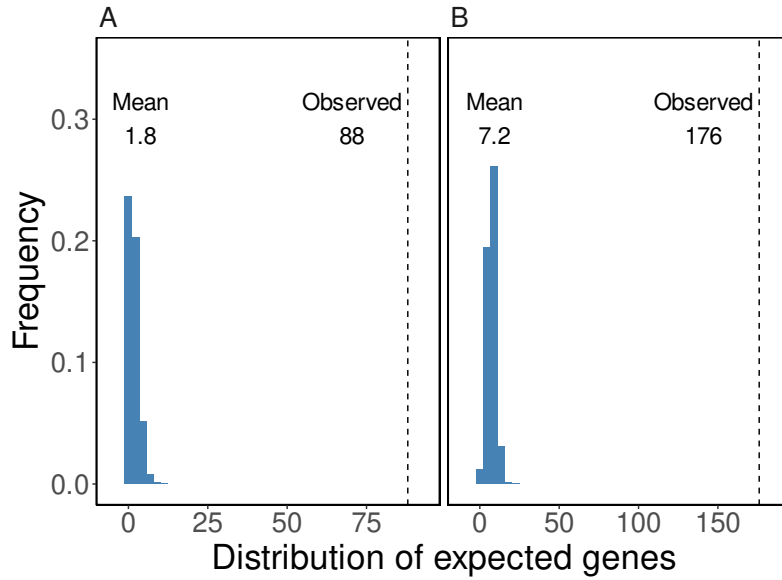


Figure 2.15: Null distribution of the number of genes obtained by shuffling the 1234 highly complex regions in human, for a 10 Kb window, 10^6 times. A) “anterior/posterior pattern specification” and B) “anatomical structure formation involved in morphogenesis”.

This distance implies that even if the number of iterations for the enrichment process was increased, the likelihood that we obtain, purely by chance, as many genes associated with this specific GO term as the number observed in the highly complex regions is vanishingly small.

If, in spite of the observed skew, one were to assume a normal distribution for the null distribution of GO occupancy, one can calculate the probability of randomly picking regions that intersect at least as many genes as observed for a given term. This approach was implemented in an experimental version of *ego* which calculates the complement of the cumulative density function (CDF) of the normal distribution. I can then compare the P -values obtained from the Monte Carlo test (P_m), with those obtained from the parametric test (P_p).

Of the 241,273 enriched GO terms ($P_m < 10^{-2}$), 73,587 terms had a $P_p > 0$. Meaning,

in 69.5% of all of our enriched terms the probability of, merely by chance, obtaining such a high number of observed genes for a given term is zero. The maximum observed P_p is 4.7×10^{-2} for “sensory perception” in the common vampire bat for a window of 1000. In fact, the smaller the window, the more likely it is that $P_p > 0$. In a 1 Kb window size, 56.9% of the terms had a $P_p > 0$, while for a 2 Kb window this percentage decreased to 30.8%. For a 10 Kb window, only 3.4% of the enriched terms have a $P_p > 0$.

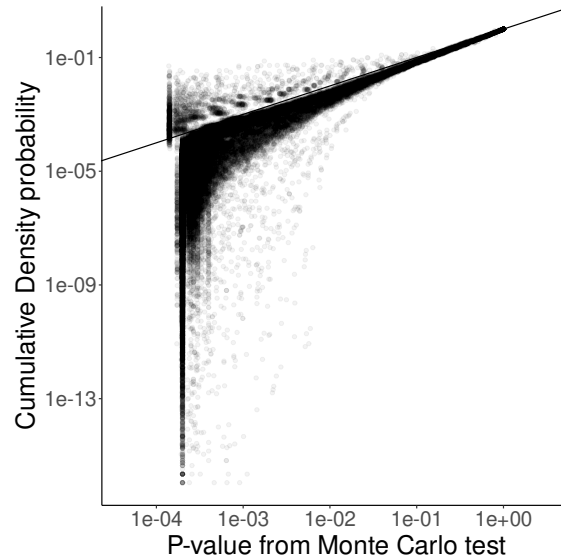


Figure 2.16: Correlation between P_m and P_p for all described GO terms, where $P_m > 0$ and $P_p > 0$.

Figure 2.16 shows the correlation between all the 59,997 described GO terms with $P_m > 0$ and $P_p > 0$. For larger P -values (>0.1) there is a strong association between both measures. However, for smaller P -values the Monte Carlo test is more conservative. This may be due to the skewness of the distribution. Because the parametric test is based on a normal distribution, it ignores the long tail of the true null distribution, thus values that fall on the right tail of the distribution are considered to be outliers.

Supplementary Material SM1 lists all enriched biological processes detected with a 10 Kb window for 10^6 Monte Carlo iterations associated with each organism, as well as a comprehensive table listing all the species for which a given term was considered to be enriched.

2.3.4 Time and memory consumption

In this chapter, I expanded the analysis of high-complexity regions from human and mouse to a sample of 18 genomes covering the full mammalian diversity. The time and

memory required to run the analyses presented in this chapter are described in this section.

2.3.4.1 Genome analysis

The 18 mammalian genomes were indexed with *Macle*, which takes time roughly proportional to the length of the genome ranging from 36 mins 13 s, elapsed wall clock time, to 1 h 28 mins (Figure 2.17A). Similarly, memory consumption was linear in genome length and ranged from 139.8 GB for the pale spear-nosed bat to 238.1 GB for the gray short-tailed opossum (Figure 2.17B).

RepeatMasker required 44 hours and 80.1 GB of memory to mask all repetitive regions in the human genome. In contrast, indexing the human genome with *Macle* required 196 GB. However, while *RepeatMasker* required 44 h to mask the entirety of the human genome, constructing the *Macle* index required only one hour, meaning it was 44x times faster.

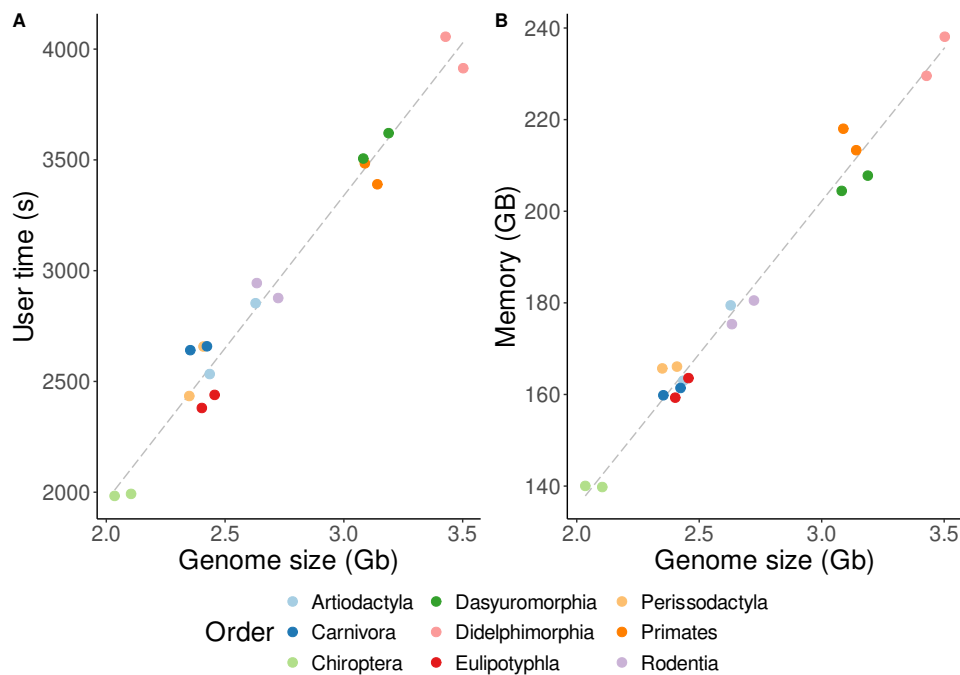


Figure 2.17: A) User time (s) and B) Memory (GB) required by *Macle* to index each genome.

2.3.4.2 Orthologous annotation

Annotating orthologous genes with *eggNOG-mapper* required between 19 m 43 s for the european shrew and 1 h 6 m for the human genome. During the annotation process, the *eggNOG* database was loaded into memory, speeding up the annotation but incurring

higher memory costs. As seen in Figure 2.18 the human and mouse required the highest memory (49 GB and 47.5 GB), while the white-toothed shrew had the lowest memory requirement of 44.7 GB.

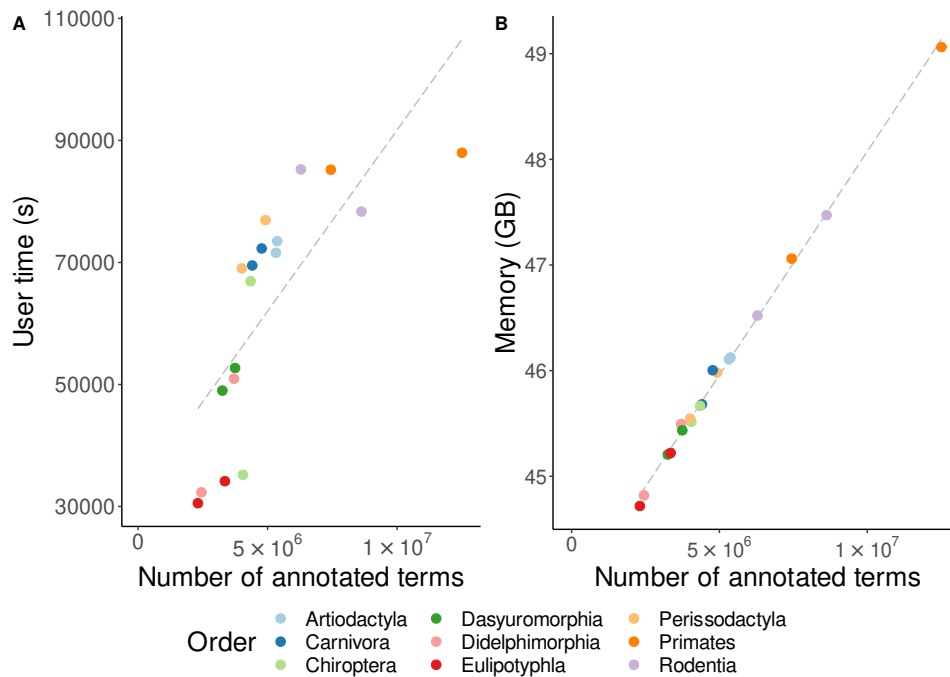


Figure 2.18: A) User time (s) and B) Memory (GB) required to annotate orthologous genes.

Overall, the time and memory required to annotate each genome corresponds to the number of annotated ortholog genes, independently of genome size.

2.4 Discussion

Pirogov et al. observed that highly complex regions in human and mouse are enriched for developmental genes. They explained this by generalizing the classical observation from the first draft of the human genome that the *Hox* clusters are almost free of transposons [15]. There are four *Hox* clusters in mammals, each of which contains up to eleven transcription factors involved in early development. The authors of the first draft of the human genome proposed that the human *Hox* clusters lack transposons due to purifying selection on highly regulated genomic regions. In this chapter, I started by directly investigating the relationship between sequence complexity and repetitive elements. Then, I extended the sample of organisms studied from two, humans and mouse, to 18, encompassing a wide diversity of placentals and marsupials.

There is a long tradition of studying repetitive elements, like transposons, in complex genomes [109]. This is done mainly by using homology searches against known repetitive elements, as implemented in programs like *RepeatMasker* [110], but there is a large range of tools for annotating transposable elements including tools for *de novo* annotations [68]. Pirogov et al. also devised a method for identifying repetitive regions that does not rely on *a priori* knowledge. This reliance on exact matches was motivated by the efficiency with which they can be looked up using suffix arrays. The approach is based on the observation that regions with homology elsewhere have, on average, longer exact matches than regions without homology. By using the null distribution of match lengths [19], Pirogov et al. defined the match complexity, C_m , as the ratio between the observed number of match factors in a given window and the expected number. Regions with $C_m \approx 1$ have no homology while regions with $C_m = 0$ have an exact match elsewhere. The computation of C_m is implemented in the program *Macle*, which I show is 44 times faster than *RepeatMasker* when first indexing the human genome and can scan the index of the human genome in a matter of seconds compared to 44 h for *RepeatMasker*.

A quick comparison shows that the regions obtained by *Macle* are not entirely free of repeats masked by *RepeatMasker*. This is because *RepeatMasker* is more sensitive than *Macle*. Through simulations, I found that regions picked by *Macle* are expected to be free of transposons that have diverged by less than 20% (Figure 2.9). This observation is further validated when looking at real data, in this case, the human genome, where on average, masked repeats that intersected highly complex regions had a divergence rate of 24.7%. The overlap between highly complex regions and masked regions is also small, counting for less than 10% of the complex regions and averaging a length of 148.5 bp (Figure 2.2B).

Assuming a mutation rate of $0.5 \times 10^{-9} \text{bp}^{-1} \text{year}^{-1}$ [106], *Macle* can detect duplication events that occurred in the human genome up to 200 million years ago, at the end of the Triassic and the start of the Jurassic. It was during the Triassic that the cynodonts, which eventually gave rise to all mammals, diverged [111]. In contrast, *RepeatMasker* can detect transposon events that occurred up to 400 million years ago, after the duplication of the *Hox* clusters that is present in all vertebrates. Thus, highly complex regions are free of transposons or duplication events that have occurred since the emergence of placental mammals. Older events, like the duplication of the *Hox* clusters, are too far diverged to be considered repeats by *Macle*.

The HERV-Fc2 retrovirus that intersected a 7.2Kb high complex region on the human X chromosome, belongs to a class of retroviruses with very limited penetration in the

Homininae lineage [112]. According to the *DFAM* database, the HERV-Fc2-LTR (accession DF000000180.5) contains seven non-redundant hits in the human genome with an average divergence rate of 26.1%. In the GRCh38.p14 human reference genome, only four of them were masked in my *RepeatMasker* run: one on chromosome 1, one on chromosome 14 and two on the X chromosome. The two LTRs located on the X chromosome have a divergence rate of 2.7% and 2.2% and are interspersed by the HERV-Fc2-int component contained within a highly complex region. The HERV-Fc2-int encodes the ERVWE2 pseudogene, which despite producing a truncated transcript is thought to be associated with multiple sclerosis in humans [113]. If I extract the highly complex interval (chrX:97,841,772-97,849,17) and blast it against the human genome with *megablast*, besides one short fragment (86 bp) on chromosome X, only one other match is found. The match is located on chromosome 11 and is 1953 bp long with 553 mismatches/gaps, covering only 18.9% of the highly complex interval. Dividing the total number of mismatches (505 bp) by the total aligned region (1905 bp) yields 0.27 mismatches per bp which is above *Macle*'s detection rate of 0.2 mutations/site. This explains why it escaped detection by *Macle*.

2.4.1 Detection of highly-complex regions

The proportion of each genome that is considered to be complex is dependent on the *Macle* window size. The shorter the window, the larger the proportion of the genome that is considered to be complex. However, the details of this general trend depend on the species investigated. While for some species the proportion of highly complex regions can go up to 34.4% of the genome, for others this proportion is around 10.8%. For an extremely large window of 50 Kb, although on average less than 0.1% of the genome was considered to be highly complex, but in the tasmanian devil this proportion was of 0.7%.

A shorter window length also means that a higher number of promoters intersect these complex regions. For a 1 Kb window more than 50.9% of all genes were located within these regions. As the window increases, the proportion of annotated promoters decreases, until at a window of 5 Kb less than 36.4% of all promoters are annotated. The reduction in the number of annotated promoters also leads to a reduction in the number of enrichment processes, albeit not necessarily in the same magnitude. For example, doubling the window size from 1 Kb to 2 Kb, reduced the average proportion of annotated genes from 69.4% to 47.8%, and the proportion of enrichment terms from 36.5% to 31.8%

2.4.2 Enrichment analysis

The *eggNOG* database [51] contains fine-grained associations between orthologs and paralogues across a wide range of organisms. With *eggNOG-mapper*, it is possible to accurately retrieve information for curated terms of a given taxon, while reducing the number of false negatives (non-curated) [79], meaning that I can be confident about the GO terms assigned to each protein.

If I compare the abridged gene2go file for the mouse (22,091 unique terms) with the existing gene2go files, for example, the NCBI gene2go file (17th July, 2024) contains 18,966 unique entries, there is a much larger number of described GO terms. This may be due to two reasons. First, since I obtained the description of the GO terms directly from the GO consortium database, I expect the list to be as complete and as up-to-date as possible regarding existing GO terms. Secondly, *eggNOG-mapper* lists the entire GO ontology graph for each term instead of only the tips [114]. By including all parental terms instead of only the tips, the total number of described GO terms will be increased.

Previously, the results from Pirogov *et al.* suggested that longer windows leads to a higher enrichment signal, as could be seen when comparing the enrichment ratios for human and mouse across the three windows tested (5 Kb, 10 Kb and 20 Kb). However, in Figure 2.12 it was shown that this is not, in fact, a general pattern. While in primates and shrews, the mean enrichment increases with the size of the sliding window, for other species the mean enrichment ratio tends to plateau.

Enriched terms are highly associated with developmental processes ranging from regulatory, to developmental processes such as organ development or embryonic morphogenesis. 16 of the 20 most globally enriched GO terms are directly associated with embryonic morphogenesis, development of the central nervous system or regionalization, meaning the association between highly complex regions and developmental factors is strong. I have shown that this is the case across the mammalian clade.

Finally, to calculate the P -values a Monte Carlo approach was used. While this method allows for an accurate estimation of P , it is limited by the number of iterations that can be performed. To solve this issue, the complement of the CDF of the normal distribution of the expected number of genes per term was applied. However, the true null distribution is not normal (Figure 2.15). Due to the right-skewness the P_p -values obtained from the CDF are overestimated compared to the truth obtained with the Monte Carlo, as was shown in Figure 2.16. The smaller the P -value, the less reliable the CDF is. Thus, the Monte Carlo approach should be preferred due to its accuracy. A future step for this study would be,

therefore, to develop another measure that accurately reflects the null distribution of the expected number of genes for a given term.

2.5 Summary

In this chapter, I confirm that highly complex regions are free of recent transposition or duplication events. By assuming a constant mutation rate equal to the human mutation rate, it was shown both through simulations and real data, that these regions do not contain transposons or duplications that have occurred since the emergence of currently extant mammals.

I also expanded the prior investigation of complexity in human and mouse genomes in two directions. First, a framework was proposed to search for highly complex regions and perform an enrichment analysis in genomes that are not functionally annotated. With this, the search for highly complex regions was expanded from two to 18 species. While only mammalian orders were considered in this chapter, theoretically, expanding this framework to other vertebrate species should be possible.

Secondly, I confirm that highly complex regions are generally associated with developmental genes across a wide range of species. The most globally enriched terms are highly associated with embryonic development and organ or skeletal morphogenesis. Because highly complex regions can be identified even in newly assembled genome sequences, this could be a method for identifying regions of functional interest in less studied organisms. Furthermore, identified complex regions with no known associated gene or function might be prioritized for further experimental studies.

Chapter 3

Large-scale marker discovery in prokaryote genomes

Beatriz Vieira Mourato¹, Ivan Tsers¹, Svenja Denker¹², Fabian Klötzl³ and Bernhard Haubold¹

Affiliations:

¹Max Planck Institute for Evolutionary Biology, RG Bioinformatics, Plön, Germany

²University of Lübeck, Lübeck, Germany

³Cambridge CB2 1LB, United Kingdom

Published manuscript:

This manuscript was published on Bioinformatics Advances in July 2024.

DOI: <https://doi.org/10.1093/bioadv/vbae113> [115]

3.1 Introduction

3.1.1 Finding diagnostic markers

Diagnostic PCR markers are routinely used to track diseases such as Covid-19. A good marker amplifies all target pathogens and nothing else. In other words, a good marker has high sensitivity and specificity. Traditionally, marker regions for designing PCR primers were selected based on literature and experimental evidence, where genes that encode a given protein or toxin essential for pathogenicity tended to be favored. However, with fully sequenced target genomes, the entire genome may be a potential marker. In this situation, the recommendation is to avoid running premature specificity checks on a large database like the non-redundant collection of nucleotides, *nt*, not only because they are computationally exhausting, but because there is the risk of incurring false positives in organisms that cannot co-occur with the target [116]. Instead, the preferred method to find marker candidates is based on the simple evolutionary insight that cross-reactive material is concentrated in a target's closest relatives. So the current method to find marker candidates is to look for regions common to the targets that are absent from a set of close relatives, neighbors. Figure 3.1 illustrates this approach for a sample of three targets and four neighbors, which can be thought of as complete genome sequences.

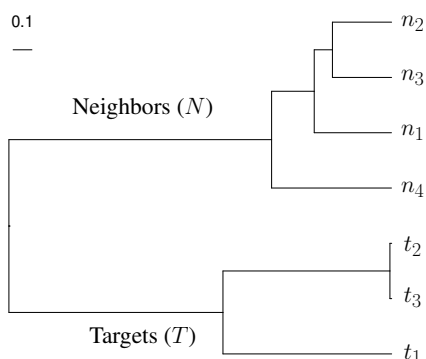


Figure 3.1: Schematic representation of the phylogeny of targets T and neighbors N used in marker discovery. Where the targets T must be monophyletic.

Since there are relatively few published tools for finding diagnostic markers regions currently available, I start by first listing existing tools and their main features and current issues. I started by surveying three published programs for finding markers from targets and neighbors, *Uniqprimer* [117], *Kec* [20], and *Fur* [21].

Uniqprimer starts by aligning one of the targets, the target representative, r , with all

neighbors. The alignment is calculated with *NUCmer*, which is part of the *MUMmer* software package for comparing genome sequences [118]. *MUMmer* is based on a suffix tree for finding maximal unique matches, hence the name. The regions in r not found in the neighbors are then intersected with the remaining targets through iterative alignments to find marker candidates. These can be subjected to primer design using a program like *primer3* [119]. The authors of *Uniqprimer* used it to design diagnostic primers against the bacterial pathogen *Dickeya dianthicola*, which causes soft rot in potatoes.

The first tool, *Uniqprimer* is written in Python and its code is freely available. However, it was designed as a tool for the Rice Galaxy and the command line version, which can be used for large scale testing of diagnostic markers, has not been maintained. Additionally, a small test run of *Uniqprimer* on the Rice Galaxy server, showed that it could not identify a 1 Kb marker region in a set of 10 simulated targets and neighbors.

The second tool for marker discovery from targets and neighbors, *Kec*, uses k-mer elimination by cross-reference. For each target, *Kec* identifies regions containing k-mers that are absent from the neighbors. These regions can then be intersected to using some other tool to yield marker candidates. *Kec* has been used to design diagnostic markers for *Xanthomonas euvesicatoria*, which causes bacterial spot disease in tomato [120], and for *Exserohilum turicum*, a fungal pathogen that causes northern corn leaf blight [121].

In contrast to *Uniqprimer*, *Kec* (v1.0) is run on the command line but only checks whether the unique kmers for a given target sequence are present in at least another targets. Thus performing a partial intersection. Furthermore, by default *Kec* does not include the reverse complement in the analysis. This means that neither *Uniqprimer* nor *Kec* are fully functional tools for markers discovery.

The third surveyed tool, *Fur* is based on maximal matches like *Uniqprimer*. These matches allow *Fur* to identify marker regions with no homology in the neighbor set. Because the length distribution of maximal matches is known [19], it is possible to pick regions with effectively random matches that have no homology from the target genome quite easily. *Fur* relies on an enhanced suffix array (*esa*) to look up matches. The rate-limiting step in constructing an enhanced suffix array is constructing the underlying suffix array. There are highly optimized libraries for constructing enhanced suffix arrays, such as *libdivsufsort* [122], on which *Fur* is based.

Fur is split into two programs: *makeFurDb* and *fur*. *makeFurDb* calculates the match factors of a target representative and constructs a *Fur* database that can be repeatedly queried with *fur*. This is useful because the enhanced suffix array only needs to be calculated once

for each dataset, while identifying the marker regions depends on the applied search parameters. The query itself with *fur* is divided into three steps: subtraction I, intersection and subtraction II as shown in Figure 3.2. In subtraction I, *fur* counts the number of match factors in a sliding window to detect unique regions in the target representative, t_1 in Figure 3.2. In the intersection step, the regions identified in subtraction I are intersected with all the other target genomes. The second subtraction step is a quality check with *blastn*, which compares the regions returned by the intersection against the neighborhood and removes all remaining homologous regions.

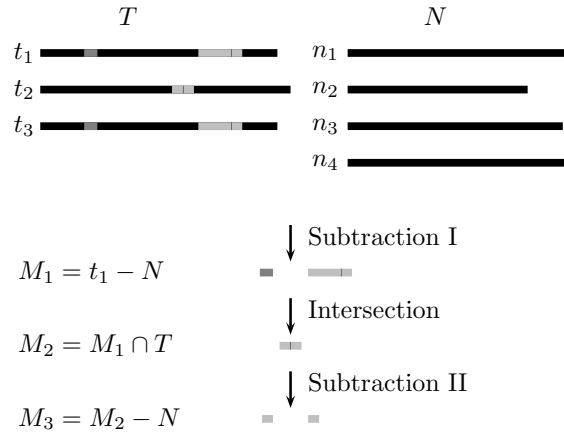


Figure 3.2: The three *fur* steps.

Marker regions developed with *Fur* (v3.2) are highly accurate when tested *in silico* on *Escherichia coli* and *in vitro* on *Lactobacilli*. What is still lacking, however, is a systematic evaluation of *Fur*, when applied to a representative sample of bacteria. Before I can carry out such a survey later in this chapter, let's revisit the *Fur* algorithm for two reasons. The first is that the memory requirement of *Fur* scales with the size of the neighborhood, which is problematic when analyzing samples of hundreds or even thousands of bacterial genomes collected during pathogenic outbreaks. Requiring approximately 65 bytes of RAM per bp, larger datasets may fail to run due to a lack of available memory. If the memory constraint of *Fur* could be overcome, then it might be possible to expand the range of taxa that can be analyzed with *Fur* on conventional hardware like ordinary laptops. The second reason for revisiting the algorithm is that so far matches within the target cannot be distinguished from matches anywhere else. This means that repeats cannot be markers even if confined to the target.

To explain why *Fur* has memory issues, let's look at its algorithm in more detail. Up to version 3.2 of *Fur*, *makeFurDb* concatenated the target representative and all elements of the neighborhood N into a single sequence, S , and converted it into an enhanced suffix

array, *esa* as described in Algorithm 2, lines 1-2. The memory requirement for constructing the *esa* is linear in the size of the input. Then, it walks along one target sequence, the target representative, *r*, and looks up the length, *k*, of the maximal match starting at the current focal position, *i* (lines 6-9). It stores this length at position *i* in the array of match lengths, *ml* (line 10). Then, it advances the focal position by the match length, line 11, and repeat.

Algorithm 2 Algorithm for finding match factors in *Fur* v3.2

Require: *r* {target representative}
Require: $N = \{n_1, n_2, \dots, n_m\}$ {set of *m* neighbor sequences}
Ensure: *ml* {array of match lengths along *r*}

- 1: $S \leftarrow r + n_1 + n_2 + \dots + n_m$ {concatenate *r* and *N*}
- 2: *esa* \leftarrow **getEsa**(*S*)
- 3: *esa*.Lcp[|*r*| + 1] \leftarrow 0
- 4: *i* \leftarrow 1
- 5: **while** *i* \leq |*r*| **do**
- 6: *j* \leftarrow *esa*.Isa[*i*]
- 7: $\ell_1 \leftarrow$ *esa*.Lcp[*j*]
- 8: $\ell_2 \leftarrow$ *esa*.Lcp[*j* + 1]
- 9: $k \leftarrow \max(\ell_1, \ell_2, 1)$
- 10: *ml*[*i*] \leftarrow *k*
- 11: *i* $\leftarrow i + k$
- 12: **end while**

The memory issues of *Fur* arise from the first step of the algorithm, where the target representative is concatenated with all neighbor sequences, and to be exact, their reverse complement sequences before the *esa* is calculated from them (lines 1 and 2). However, instead of comparing all the suffixes of *r* and the neighbors at once, this can be done sequentially. Here, there is a choice between indexing *r* and sequentially comparing n_1, n_2, \dots to it or sequentially indexing n_1, n_2, \dots and comparing *r* to each index. So, in this chapter, I compare two new *Fur* algorithms whose memory consumption is proportional to either the length of the longest neighbor sequence or the length of the reference target.

Apart from saving memory, another advantage of the sequential algorithms is that they consider only matches between *r* and *N*. As a consequence, repeats in *r* that are absent from *N* can now also be picked as markers.

3.1.2 Curating targets and neighbor

With a memory-optimized *Fur* in hand, it can now be used to search for diagnostic markers across a wide range of prokaryotic pathogens. However, this raises the question of

how to curate good sets of targets and neighbors automatically. So far, these sets have been manually curated by relying on the literature. Nevertheless, if the goal is to search for diagnostic primers at a large scale, then the search for targets and neighbors also needs to be automated.

Although there are programs for obtaining all sequences associated with a given organism or strain, like the *datasets* tools from the NCBI, their application relies on the assumption that the underlying taxonomy is correct. However, it is known that many of the genomes available at the NCBI are wrongly classified [123]. Relying exclusively on taxonomical information to distinguish between targets and neighbors can thus lead to true targets among the neighbors or true neighbors among the targets. A true target among neighbors may reduce the output of subtraction I to nothing, while a true neighbor among targets may significantly reduce the output of the intersection. Currently, the NCBI relies on the Average Nucleotide Identity (ANI) to assess prokaryotic species boundaries [124, 125] and to confirm the classification of a submitted sequence. ANI is proven a reliable measure to identify species boundaries in prokaryotes to [126–128], where organisms with a 96% ANI or more are considered as belonging to the same species [124].

Another difficulty in the application of the NCBI taxonomy is that, when the first bacterial genomes were submitted to GenBank, each strain was assigned a unique “taxid” (taxon ID) in the NCBI taxonomy database. As the number of available sequences grew, this was no longer feasible [125, 129]. Nowadays, instead of assigning a new taxon ID to a newly submitted strain, this information is recorded in the metadata. Unfortunately, this change was not applied retroactively. Thus, depending on the type strain one is interested in, the taxonomic ranks may differ. Let us consider two pathogenic *Escherichia coli* strains and the number of taxonomic ranks one needs to climb to reach *species* level. As shown in Figure 3.3, the first strain, “O157:H7 str. Sakai”, needs to climb two ranks (strain → serotype → species), while the second strain, “IAI39”, only needs to climb one rank (strain → species).

Genus: <i>Escherichia</i>	Genus: <i>Escherichia</i>
→ Species: <i>E. coli</i>	→ Species: <i>E. coli</i>
→ Serotype: <i>E. coli</i> O157:H7	→ Strain: <i>E. coli</i> IAI39
→ Strain: str. Sakai	

Figure 3.3: Taxonomic ranks of *E.coli* str.Sakai and IAI39.

It is thus crucial to have a tool that can not only navigate the NCBI taxonomy database, but that can also group sequences into the phylogenetically correct target and neighbor sets. With this in mind, the package *Neighbors* was developed for finding target and neighbor

genomes from the NCBI taxonomy, which are then sorted into phylogenetic targets and neighbors.

The automatic identification of targets and neighbor genomes, would be the first step in performing marker discovery in the large. After identifying the target and neighbor sets, the marker regions can be found, ideally with a tool that can deal with large datasets like *Fur*. The markers returned by *Fur* can then be assessed by evaluating primers constructed from them, as explained in the next section.

3.1.3 Evaluating primer pairs

Given a set of markers, primers are routinely constructed for them using a program like *primer3*. However, this still leaves the question of how to evaluate these primers *in silico* before carrying out *in vitro* tests.

Up to version 3.2, *Fur* relied on awk scripts to design and test primers. Now, these tools have been reimplemented in the *Prim* package. *Prim* converts the marker regions into primer pairs and then scores their sensitivity and specificity using *in silico* PCR. Let t_p be the proportion of true positive nucleotides identified by *in silico* PCR, f_p the proportion of false positives and f_n the proportion of false negatives. Then, the sensitivity (S_n) denotes the proportion of true positives among all true elements,

$$S_n = \frac{t_p}{t_p + f_n} \quad (3.1)$$

while specificity (S_p) denotes the proportion of true positives among all predicted elements,

$$S_p = \frac{t_p}{t_p + f_p} \quad (3.2)$$

Primer pairs are scored twice. First, according to the NCBI taxonomy classification. Then, the taxonomy-based scores are checked using phylogenetic distances.

3.1.4 *Mapro*: markers for prokaryotes

So far, in our quest to make *Fur* applicable to large-scale diagnostic marker finding, I have described the need to revisit its algorithm to reduce its memory requirements and to allow any target regions to be marker candidates, even if repetitive. Then, I introduced

two new packages, *Neighbors* and *Prim*, to facilitate the design of marker regions and to evaluate the sensitivity and specificity of the designed primers.

These three packages have been joined into a pipeline named *Mapro*, which guides the user through the steps required to design and score primer pairs for a given bacterial strain. The *Mapro* pipeline can be split into three stages: first, search for genomes with *Neighbors*, then search for marker regions with *Fur*, and finally, find and evaluate primer pairs with *Prim*. Figure 3.4 shows the three *Mapro* steps.

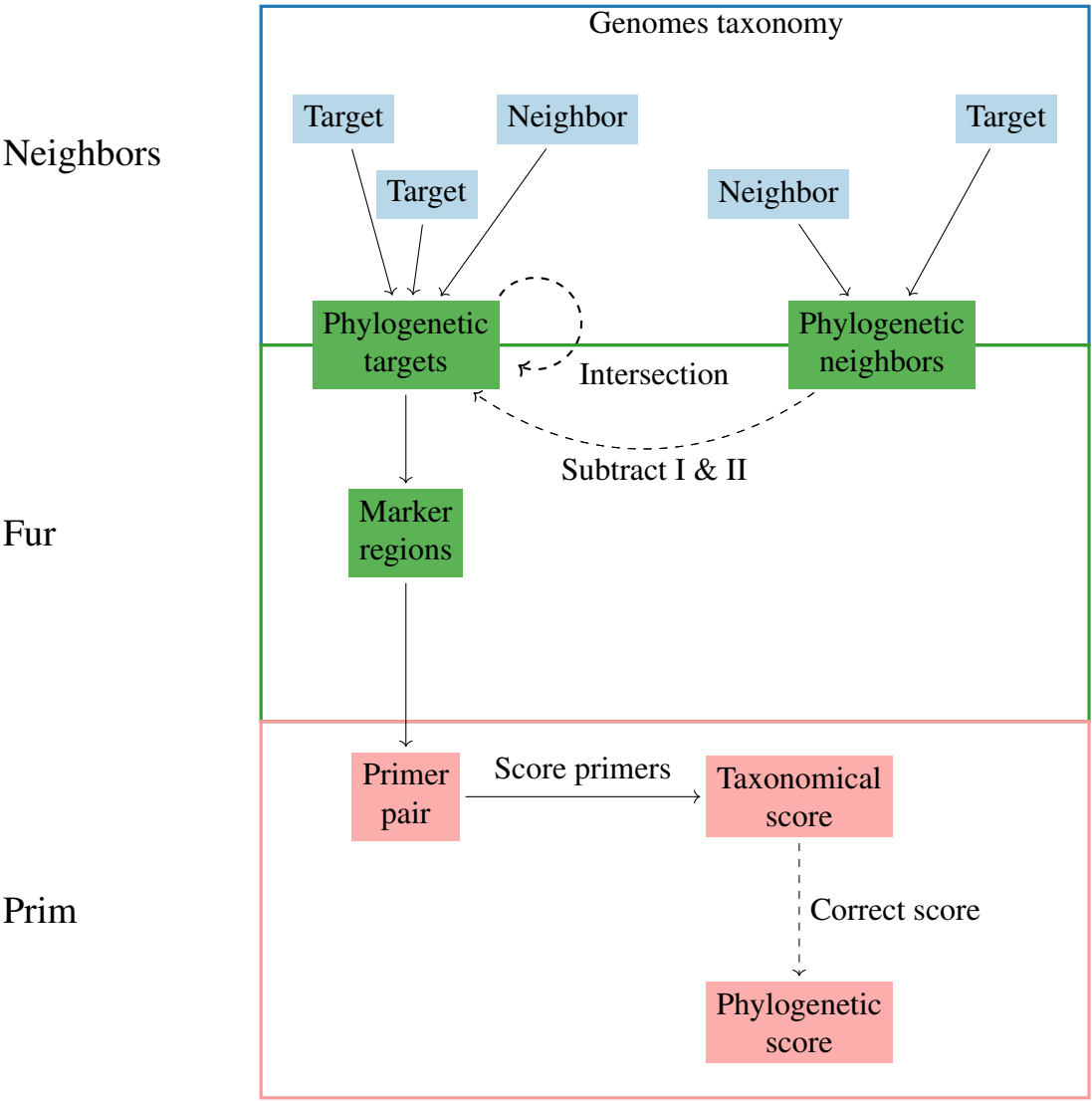


Figure 3.4: The three steps of the workflow: Finding genomes with *Neighbors*, finding markers with *Fur*, and finding and evaluating primers with *Prim*.

Mapro starts by using *Neighbors* to obtain a list of all taxonomical target and neighbor genomes and sorts them into monophyletic targets and neighbors, the complement. This

is the input for the second step, finding markers, which is done with *Fur*. Finally, marker regions are taken as an input to *Prim* where primer pairs are designed and scored on their sensitivity and specificity based on the NCBI taxonomy of *in silico* PCR hits. The classification of these hits is then checked based on their phylogenetic distances, perhaps leading to corrected sensitivity and specificity scores.

3.1.5 Chapter aims

In this chapter, I start by comparing two new memory-efficient *Fur* algorithms that enable the search for markers in large datasets using simulated data.

Then, I present the *Mapro* pipeline with its three steps:

1. *Neighbors*: Finding genomes
2. *Fur*: Finding markers
3. *Prim*: Finding primers

I test this pipeline by evaluating 120 prokaryotic reference strains.

3.2 Methods

3.2.1 Fur algorithm

As explained above, the reason why *Fur* has memory issues is because it concatenates the target representative and all elements of the neighborhood into a single sequence that is converted to an enhanced suffix array, leading to memory requirements that are proportional to the size of the neighborhood. By reducing the length of the sequence that is being indexed, the memory requirements of *Fur* can be reduced. Here, I compare two candidate algorithms based on one or more small indexes. First, Algorithm 3, *nesa*, iterates over the neighborhood (N) and calculates the *esa* for each element. This means that the *esa* is calculated for every neighbor sequence. Alternatively, since the target representative r remains constant, it could be indexed instead. The second algorithm, Algorithm 4, *resa*, indexes the target representative r and streams the neighbors' sequences against it. Furthermore, both *nesa* and *resa* lend themselves to parallelization, decreasing the runtime required for analyzing a given dataset.

3.2.1.1 *nesa*: Streaming the neighbors N

The *nesa* algorithm calculates an enhanced suffix array, *esa*, for every neighbor sequence (line 2). Then it streams the target representative, r , against the *esa* of the current neighbor n_i . Specifically, it calls the method `MatchPrefix` to match the prefix of the suffix of r starting at position j against n_i , which returns the length of the match, ℓ (line 5). This then compared to the match length already stored at position j of the match lengths array (`ml`). If ℓ is greater than the current entry, `ml[j]` is updated to ℓ (line 6). The maximization of the match lengths is possible because the order in which the matches are found is irrelevant. This step allows the replacement of the full index used in the previous algorithm with partial indexes. Finally, the algorithm skips the match by its length + 1 (line 7). In the previous *Fur* algorithm (Algorithm 2, line 11), it skipped to the position after the end of the match, as inherited from the Lempel-Ziv decomposition. However, since that position is a mismatch it also needs to be skipped to avoid creating short random matches.

Algorithm 3 Iterative algorithm, *nesa*, for finding match factors in *Fur*

Require: r {target representative}

Require: $N = \{n_1, n_2, \dots, n_m\}$ {set of m neighbor sequences}

Ensure: `ml` {array of match lengths along r }

```
1: for  $i \leftarrow 1$  to  $m$  do
2:    $esa \leftarrow \text{getEsa}(n_i)$ 
3:    $j \leftarrow 1$ 
4:   while  $j \leq |r|$  do
5:      $\ell \leftarrow \text{esa.MatchPrefix}(r[j\dots])$ 
6:      $\text{ml}[j] \leftarrow \max(\text{ml}[j], \ell)$ 
7:      $j \leftarrow j + \ell + 1$ 
8:   end while
9: end for
```

By indexing each neighbor sequence one at a time, the memory requirement of this algorithm is proportional to the size of the longest neighbor. Additionally, streaming the target representative against the neighbors means that markers no longer need to be globally unique within the targets. They only need to be absent from the neighbors.

3.2.1.2 *resa*: Streaming the target representative r

resa is an alternative algorithm where instead of constructing one index for each neighbor sequence, the target representative gets indexed, as seen in Algorithm 4, line 1. It streams each neighbor sequence against the *esa* of r by calling its method `MatchPrefix` on the suffix starting at position j of the current neighbor sequence, n_i . Because now it is also

necessary to locate the matches in the target representative, r , MatchPrefix returns the start, s , end, e , and match length, ℓ , of the matches in the suffix array of r (line 5). The match positions are iterated over, and for every position, p , the entry in the match length array (ml) is updated whenever $ml[p] < \ell$ (lines 6-9). Finally, the algorithm skips the match plus one to avoid biasing towards short matches (line 10).

Algorithm 4 Inverted streaming algorithm, *resa*, for finding match factors in *Fur*

Require: r {target representative}
Require: $N = \{n_1, n_2, \dots, n_m\}$ {set of m neighbor sequences}
Ensure: ml {array of match lengths along r }

```

1:  $esa \leftarrow \text{getEsa}(r)$ 
2: for  $i \leftarrow 1$  to  $m$  do
3:    $j \leftarrow 1$ 
4:   while  $j \leq |n_i|$  do
5:      $(\ell, s, e) \leftarrow \text{esa.MatchPrefix}(n_i[j\dots])$ 
6:     for  $k \leftarrow s$  to  $e$  do
7:        $p \leftarrow \text{esa.Sa}[k]$ 
8:        $ml[p] \leftarrow \max(ml[p], \ell)$ 
9:     end for
10:     $j \leftarrow j + \ell + 1$ 
11:   end while
12: end for

```

The logic behind the inverted streaming in *resa* is also used in *phylonium*, a tool for calculating pairwise distances between large samples of bacterial genomes [130].

3.2.2 Simulating data

I use simulated data to compare the performance of the two Fur algorithms, both in terms of resource consumption and accuracy.

This data is simulated with *stan* [131], a program developed for simulating targets with markers and their neighbors. *stan* first creates a genealogy of targets, T , and neighbors, N , and then populates it with simulated genomes, such that the targets, T , contain one or more marker region that are deleted in the neighbors.

3.2.2.1 Time and memory consumption

I start by evaluating how the length of a sequence or the size of the neighborhood affects the memory requirements of either *resa* or *nesa* when constructing the database with *makeFurDb*.

Starting with a single target and neighbor sequence of 1 Mb, I measure the time and memory required for constructing a *Fur* database with *makeFurDb* for both of the algorithms. Then, the sequence length is increased up to a maximum of 100 Mb.

Next, I vary the number of 1 Mb neighbor sequences and compare the time and memory required for constructing the *Fur* database with *makeFurDb* for each of the algorithms. Starting with a single neighbor sequence, I increase the neighborhood up to 1000 neighbor sequences.

3.2.2.2 Algorithm parallelization

I evaluate the parallelization of *makeFurDb* twice. First, I repeat the same step described above where I start with a single target and neighbor sequence of 1 Mb, and increased up to a maximum of 100 Mb with two threads.

Then, given a fixed neighborhood of 100 sequences, 1 Mb long, I evaluate the runtime and memory required for constructing the database while varying the number of threads from 1 up to 60.

Hardware

Timed runs of *makeFurDb* and *fur* were performed on a server running Ubuntu 22.04.4. The server contains 16 Intel CPU cores (Xeon 2.10GHz) with 64 threads and 264 GB of RAM. The runtime and memory required for running each process was measured with the utility program *time*. This was called using */usr/bin/time* to distinguish it from the *bash* command *time*.

3.2.2.3 Accuracy of marker detection

To measure the accuracy of marker detections, I evaluate how the size of the applied window used with *fur* affects the detection of marker regions.

I start by simulating ten targets and ten neighbors, each 10 Kb long, with a central marker region ranging from 200 to 1600 bp, and construct the database with *makeFurDb*. For each marker length, I query the database repeatedly with *fur* varying the window size from 100 to the maximum size of the marker region in steps of 100. i.e., sequences with a 400 bp marker will be queried four times with the following window sizes: 100, 200, 300 and 400 bp.

The accuracy of the detection of the marker region was calculated following equation 3.3 [13, p122],

$$A = \frac{t_p t_n - f_p f_n}{\sqrt{(t_p + f_p)(t_n + f_p)(t_n + f_p)(t_p + f_p)}} \quad (3.3)$$

where t_n is the proportion of true negatives. This measure reflects both the sensitivity (Equation 3.1) and specificity (Equation 3.2) of a given marker region.

The accuracy was then averaged over 100 simulations and plotted together with the standard error of the mean (SEM).

3.2.3 Large-scale diagnostic marker finding

As shown in Figure 3.4, the pipeline for finding diagnostic markers on a large scale, *Mapro* (Markers for Prokaryotes), is split into three steps. First, it finds genomes and classifies them as either targets or neighbors using *Neighbors*, then it finds marker regions with *Fur*, and finally, it finds and evaluates the designed primer pairs with *Prim*.

This pipeline relies on several tools from the *Biobox* package, such as *upgma*, *nj*, *midRoot*, *keyMat*, *getSeq* and *cutSeq*.

Mapro, and its dependencies (*Biobox*, *Neighbors*, *Fur* and *Prim*) can be found in the EvolBioInf repository at <https://github.com/evolbioinf/package>. Where *package* is the package name.

Mapro is also available as a docker container at <https://hub.docker.com/r/beatrizvm/mapro>. To pull the latest image and run it iteratively with a shared folder, please see the example of a docker command on Figure 3.5.

```
#Get the latest version
docker pull beatrizvm/mapro

#Run iterative session
docker run -it mapro
```

Figure 3.5: Docker commands used for pulling and running the *Mapro* docker.

To evaluate the performance of *Mapro*, I search for marker regions across a list of 120 reference prokaryotic strains posted at NCBI at https://ftp.ncbi.nlm.nih.gov/genomes/GENOME_REPORTS/prok_reference_genomes.txt.

3.2.3.1 Finding genomes

The package *Neighbors* was developed to construct phylogenetically accurate target and neighbor sets that can be used to search for marker regions with *Fur*.

Figure 3.6 shows a schematic representation of the steps required for automatically finding targets and neighbors. Starting with a description of the type strain, *taxi* (*Neighbors*) is used to obtain its corresponding taxon ID. This is because, given a taxon ID, target sequences are those that share the same taxon, while neighbor sequences are the complement that share the parental taxon ID.

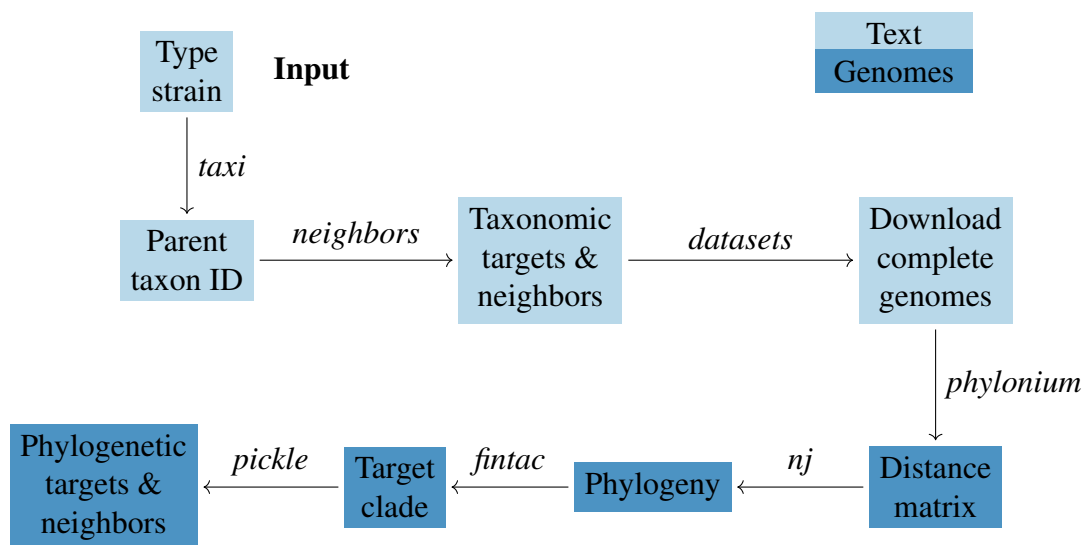


Figure 3.6: Schematic representation of how *Mapro* finds genomes.

The accessions listed by *neighbors* are then downloaded with the NCBI *datasets* (v15.12) tool. Only complete genomes were considered, and “atypical genomes” were excluded, as shown in the example command in Figure 3.7.

```

datasets download genome accession
--inputfile target_accessions.txt
--assembly-level complete
--exclude-atypical
--dehydrated --filename target_data.zip
  
```

Figure 3.7: *Datasets* command for obtaining the genomes of the 120 bacteria used in this Chapter

where the file *target_accessions* contains a list of all genome accessions identified as a target by the program *neighbors*. The process is then repeated with the list of all genome accessions identified as neighbors by *neighbors*.

At this point, the list of targets and neighbors is divided according to their taxonomy. I check the taxonomy with phylogeny. For this, I calculate the pairwise distance using *phylonium* [130] and summarize it into a midpoint-rooted phylogeny using *nj* (*Biobox*), or if only two genomes are available, with *upgma* (*Biobox*). The tree is rooted at the midpoint with *midRoot* (*Biobox*) and the internal nodes of the tree are labeled with *land* (*Neighbors*).

From this labeled tree, I identify the clade containing the maximum number of target sequences and the minimum number of neighbor sequences using *fintac* (*Neighbors*). To calculate the split score for a given node, v , *fintac* divides the number of target sequences inside the node (v_t) plus the number of neighbor sequences outside the node (v_n^*) by the total number of target (n_t) plus neighbor (n_n) sequences. Multiplied by 100, this gives the percent split of node v as shown in Equation 3.4.

$$s(v) = \frac{v_t + v_n^*}{n_n + n_t} \times 100 \quad (3.4)$$

The percent split score ranges from 0, where v contains all taxonomic neighbors and no targets, to 100, where v contains all taxonomic targets and no neighbors and taxonomy and phylogeny agree. This score can also be used to quantify the error in the taxonomic database. For example, one could take a sample of taxa and look up the fraction with $s(v) < 100$. However, since congruency between taxonomy and phylogeny cannot be assumed, the node that maximizes the percentage split score between targets and neighbors is picked.

Finally, genome accessions located in the target clade are considered to be phylogenetic targets and can be picked with *pickle* (*Neighbors*), which lists all genome accessions located in the given clade. The genomes in the target clade will be used as input to search for marker regions with *Fur*. The complement of the accessions in the target clade are classified as phylogenetic neighbors.

Because the selection of targets and neighbors relies on *phylonium* plus *fintac*, it is also interesting to evaluate the accuracy of the resulting classification. For this, *fastANI* (v1.33) [128] was used to calculate ANI, a measure whose cutoff point of 96% is used to define prokaryotic species [124]. *fastANI* was run with a kmer size of 18 and 64 threads with the matrix output option.

3.2.3.2 Finding markers

After obtaining the phylogenetically correct set of targets and neighbors, I search for marker regions with *Fur*, using the *nesa* algorithm (Algorithm 3). Figure 3.8 shows how marker regions are found in the Mapro pipeline. First, the phylogenetic targets and neighbors, found in the previous step, are given as input to *makeFurDb* to construct a Fur database which is then queried with *fur*.

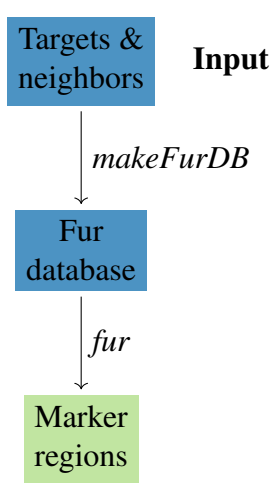


Figure 3.8: Schematic representation of how *Mapro* finds marker regions.

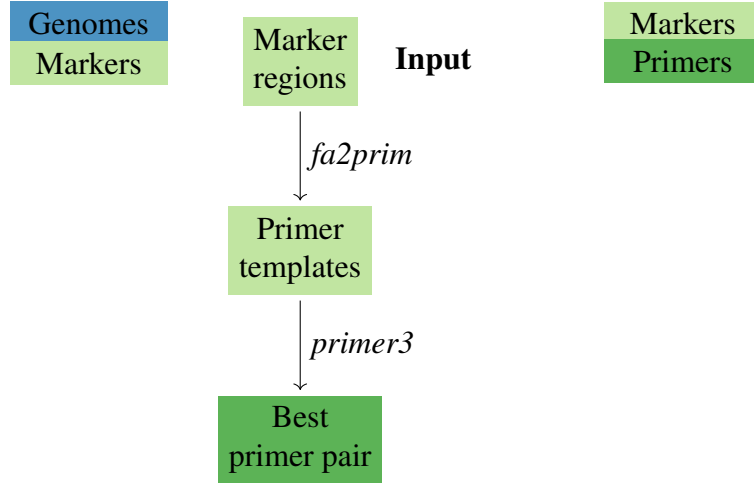


Figure 3.9: Schematic representation of how *Mapro* finds primers.

3.2.3.3 Finding and scoring primers

After obtaining the marker regions with *Fur*, these are converted to primer templates with *fa2prim* (*Prim*) and then *primer3* [119, 132] picks primers from them, as shown in Figure 3.9. For each taxon, I select the highest-scoring pair of primers whose amplicon is between 70 and 150 bp and evaluate its sensitivity and specificity.

Figure 3.10 shows the workflow for evaluating primer scores. First, I score the primers with *scop* (*Prim*), which uses taxonomical information. It performs an *in silico* PCR based on Blast searches against *nt*, downloaded on the 24th November 2023 from <https://ftp.ncbi.nlm.nih.gov/blast/db>. As previously mentioned, targets and neighbor sets do not necessarily follow their taxonomical classifications. Thus, it is necessary to correct the primer scores based on their phylogeny. This is done with *cops* (*Prim*), which evaluates whether the false negatives and false positives marked by *scop* are true positives based on the phylogeny.

To do so, *cops* compares the pairwise distance between each accession and the reference target. As shown in Table 3.1, true positives are found when the pairwise distance between

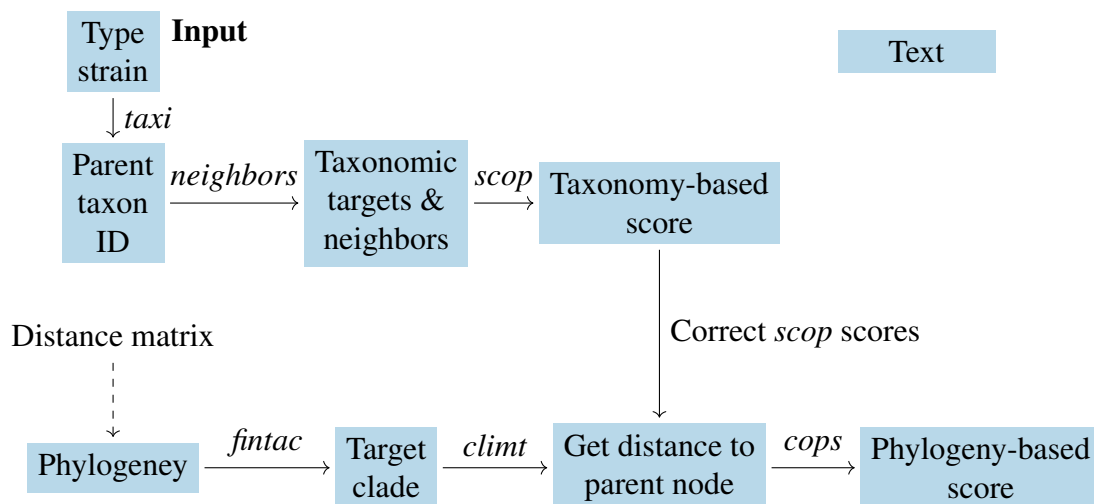


Figure 3.10: Schematic representation on how *Mapro* scores primers.

the accession and the reference target is smaller than the threshold value. Otherwise, they are false positives. Likewise, true negatives have a pairwise distance larger than the threshold. Otherwise, they are false negatives.

Table 3.1: Reclassification of a taxon i based on the relationship between its distance to the reference strain, d_i , and the threshold distance, d_t .

Original Classification	Expectation	Reclassification
True positive	$d_i \leq d_t$	False positive
False positive	$d_i > d_t$	True positive
False negative	$d_i \leq d_t$	True negative

In this analysis, the threshold was set to twice the branch length of the target clade to its parent.

3.2.3.4 Amplicon annotation

From a biological point of view, it is also interesting to know where the amplicons picked with *Prim* are located and whether they intersect any functional regions. Figure 3.11 shows how *Mapro* annotates the primer amplicons. First, using *keyMat* (*Biobox*), it matches the primers to their corresponding marker region, extracts the amplicon using *getSeq* (*Biobox*) and places it on the genome of the type strain (*blastn*). The amplicon is annotated with the GFF3 file from the type strain, obtained with *datasets*.

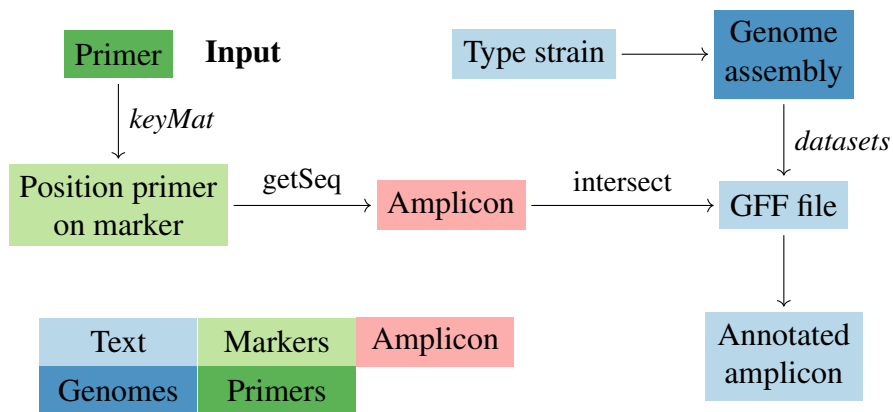


Figure 3.11: Schematic representation of how *Mapro* annotates amplicons.

3.2.3.5 Enrichment in marker regions

I also evaluate whether the marker regions are enriched for any biological function. The higher the phylogenetic distance between the targets and the neighbors, the higher the amount of marker material that will be detected. This superset of marker material may not necessarily reflect biological differences between the two sets, but instead a lack of information regarding other organisms that have yet to be sequenced. To bypass this scenario, I restrict the analysis to the top 10 taxa with the highest number of genomes, for which a primer pair could be designed. By restricting the analysis to the organisms with the higher number of sequenced genomes, I increase the likelihood that the marker regions reflect a biological difference and are not merely due to a sampling gap.

I start by obtaining a list of all genomes supported by the GO database. This was done through the PANTHER database [133] API. In total, 24 prokaryotes from our list are supported by the Gene Ontology (GO) database [52, 53], but only three of them are in the top 10 list: *Pseudomonas aeruginosa* PAO1, *Mycobacterium tuberculosis* H37Rv and *Streptococcus pneumoniae* R6.

To associate genes and GO terms, I obtained the GAF (Gene Association File) 2.2 (accessed 24th April, 2024) for each taxon from AmiGO [134], the web-based set of tools for searching and browsing the GO database. For the enrichment analysis, I converted the GAF file into a gene2go file by extracting the taxon ID, gene name, GO ID, description and category, the evidence, and associated Pubmed ID (PMID). This information is required to construct an abridged gene2go file for each taxon. Since the GAF contained no information regarding the qualifier of the GO terms, this column was left empty as a placeholder. The abridged gene2go file follows the format presented in Chapter 2.

Like the enrichment analysis of unique regions in mammalian genomes (Section 2.2.5),

the enrichment analysis on bacterial marker regions was done using the Monte Carlo approach implemented in *Gin* (v0.4). First, I mapped the marker regions for each taxon to the type strain assembly with *blastn*. Only hits that were as long as the query, the original marker region, were considered. CDS regions that intersect the mapped marker regions were annotated with *annotate*. For the Monte Carlo procedure, I used *shuffle* to shuffle the marker regions 10^6 times across the type strain assembly and annotate them once again with *annotate*. Finally, *ego*, together with the abridged gene2go file, was used to calculate the enrichment ratios of the observed GO terms for each taxon. The minimum occupancy for each GO term was set to two. For an example command see Figure 2.6.

3.3 Results

3.3.1 Simulated data

3.3.1.1 Time and Memory

I start by comparing the time and memory required for running both new *Fur* algorithms as a function of sequence length. The time (s) and memory (GB) required for running *fur* increases linearly regardless of the applied algorithm, with both algorithms behaving very similarly. Figure 3.12 shows the time and memory required for indexing and performing a window analysis for a sequence with varying lengths for each *fur* algorithm. Overall, *resa* was slightly faster than *nesa* when indexing the sequence. For example, to index a 1 Mb sequence, *nesa* required 0.5 s and *resa* 0.4 s. This 15.4% difference, however, reduces as the sequence length increases. For a 100 Mb sequence, *resa* is only 9.4% faster.

To index a 1 Mb sequence *resa* requires 82.8 Mb of memory while *nesa* requires 80.1 Mb. As the sequence length increases, so does the memory required. Still, it can be seen that there is no advantage to using one algorithm over another when it comes to sequence length. Both perform very similarly in terms of runtime and memory consumption. This is expected from the underlying algorithms, where an enhanced suffix array is computed either from the one neighbor I simulated (Algorithm 2, line 2), or from the target representative (Algorithm 3, line 1). Thus, for pairs of equally long targets and neighbors, such as I simulated here, the resource consumption should be identical, as observed.

I then evaluate resource consumption as a function of the size of the neighborhood. Figure 3.13 shows the runtime and memory required for indexing and performing a sliding window analysis on an increasingly larger neighborhood where all sequences are 10 Kb long.

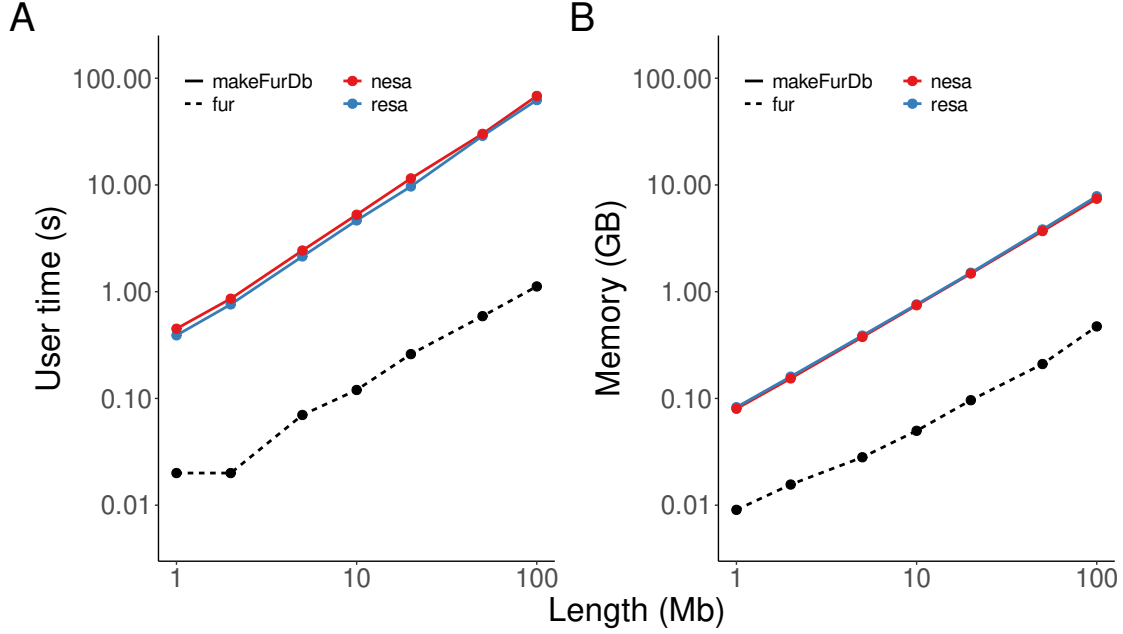


Figure 3.12: Time (A) and Memory (B) consumption of the two indexing algorithms as a function of sequence length. *fur* values shown are from the *nesa* run.

In this scenario, indexing only the target representative (*resa*) is both faster and consumes less memory than iteratively indexing the neighbor sequences. With *resa* being, on average, twice as fast as *nesa* and requiring 41.6% less memory.

The difference in run time reflects the fact that indexing plus streaming (Algorithm 3, lines 4-8) is slower than just streaming (Algorithm 4, lines 4-11). The difference in memory consumption is less intuitive, as both algorithms only hold a single index at a time, and should have identical memory consumption. However, in the Go programming language the memory of a variable is reclaimed automatically some time after it has become unreachable [135, p.35]. It is possible that this delay in releasing the memory can lead to the accumulation of occupied RAM by the repeated suffix array computations of *nesa* compared to the single suffix array computation of *resa*.

3.3.1.2 Algorithm parallelization

In the previous section *makeFurDb* was run with a single-thread. However, as it was mentioned before, the new algorithm can be parallelized, thus here I compare the runtime and memory required for running *makeFurDb* with multiple threads.

Figure 3.14A compares the elapsed wall-clock time (in seconds) required for running *makeFurDb* with one or two threads. When it comes to parallelization both *nesa* and

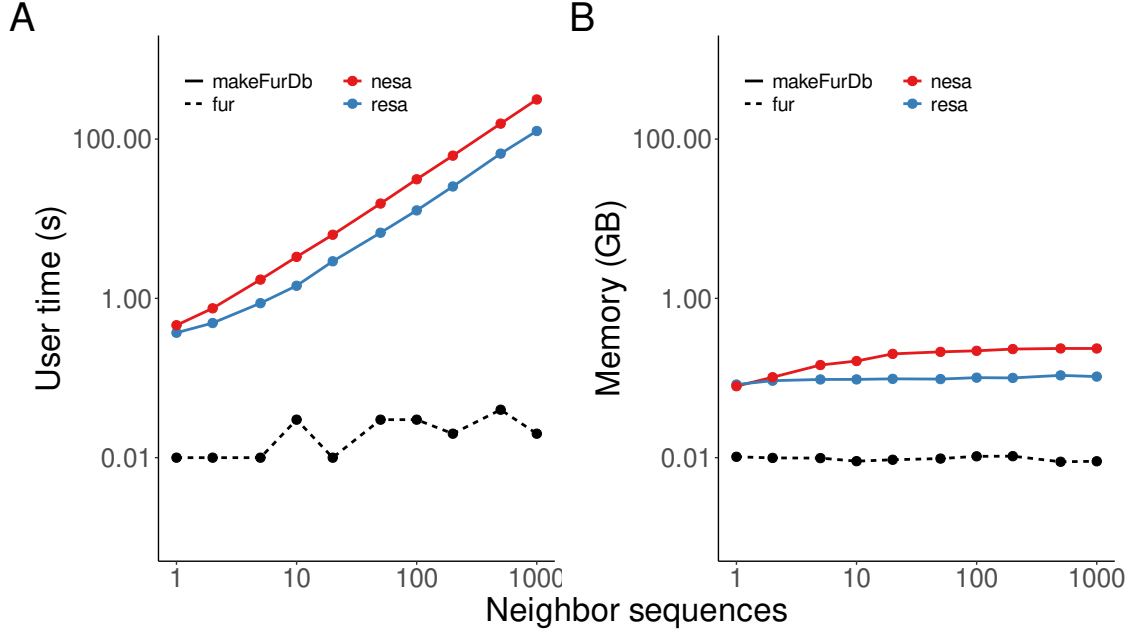


Figure 3.13: Time (A) and Memory (B) consumption of the two *fur* algorithms as a function of the number of neighbor sequences. *fur* values shown are from the *nesa* run.

resa behave quite differently. For two neighbor sequences, the *nesa* algorithm was 19.5% faster when using two threads, compared to a single one. In this case, as the number of neighbor sequences grows, so does the decrease in elapsed runtime. For 1000 neighbor sequences, constructing the index with two threads was 39.2% faster. In contrast, for *resa*, given two neighbor sequences, constructing the database with two threads was slower (45.1%), while for a 1000 neighbor sequences constructing the index with two threads was 27.1% faster.

Likewise, the increase in memory consumption for both *nesa* and *resa* differ as seen in Figure 3.14. To index two neighbor sequences, *nesa* required 29.5% more memory (GB) when using two threads compared to a single one. While for a 1000 sequence neighborhood, the memory consumption by 43.5%. In contrast, the memory increase in *resa* is less steep. To construct an index with two neighbor sequences, using two threads, resulted in a 15.8% increase in memory consumption. While indexing a 1000 neighborhood only had a 17.8% increase in memory. The difference in memory increase between double-threaded *nesa* and *resa*, is because while *nesa* simultaneously indexes two sequences, *resa* only streams them against a single index. As the neighborhood increases this different behaviour becomes more pronounced.

Although increasing the number of threads may initially decrease the runtime, with a

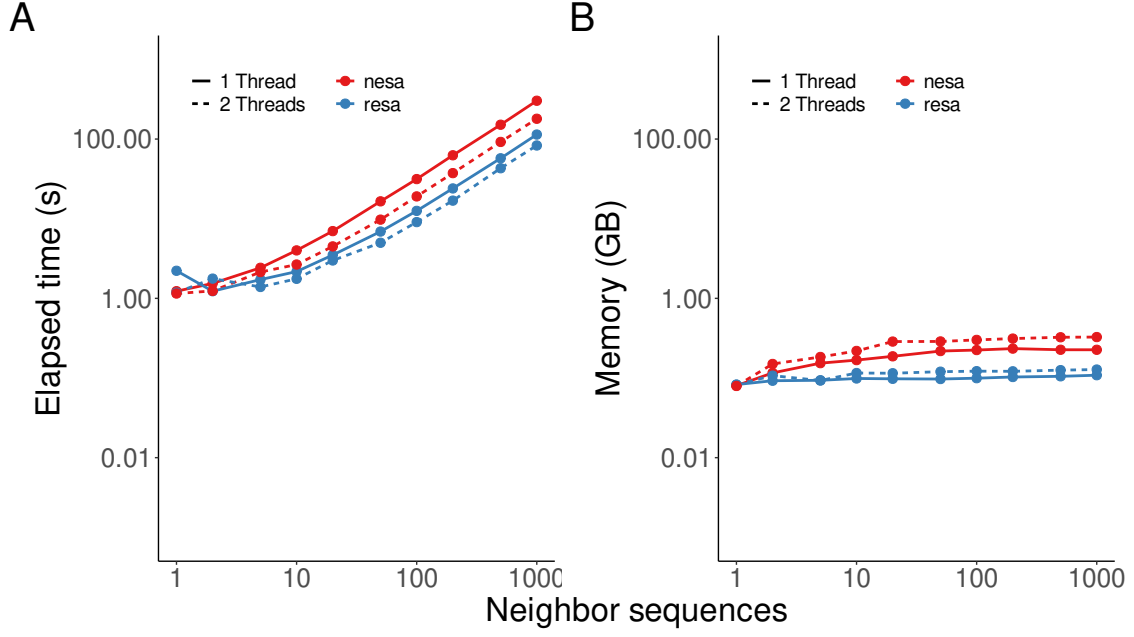


Figure 3.14: Time (A) and Memory (B) consumption of single and double-threaded *make-FurDb* as a function of the number of neighbor sequences.

penalty for memory, using too many threads can be disadvantageous. In a multi-threaded process, several threads share the same workload, thus the higher the number of threads the lower the workload given to each thread. If the number of threads is too high, however, the overhead required for starting and stopping each thread becomes larger than the workload itself. Furthermore, each additional thread causes an increase in memory consumption. Thus, the *Fur* run is being penalized both in runtime and in memory consumption.

In Figure 3.15A I evaluate the runtime and memory resource for indexing a neighborhood with 100 sequences across as a function of the applied number of threads. Starting with the *nesa* algorithm, increasing the number of threads from one to two, lead to a 39.5% reduction in runtime. However, increasing from 10 to 15 threads only caused a 1.6% reduction in runtime. Likewise for *resa*, increasing from single to double-threaded caused a 27.0% reduction in runtime, however, increasing 10 to 15 threads, only reduced the runtime in 0.7%.

Thus, it can be seen that as the number of threads starts to increase, the gains to be had in runtime quickly plateau. In contrast, the memory requirement keeps increasing for every additional thread as seen in Figure 3.15B. On average, the memory consumption for *nesa* increased 29.6% for every additional thread. While for *resa*, on average each additional

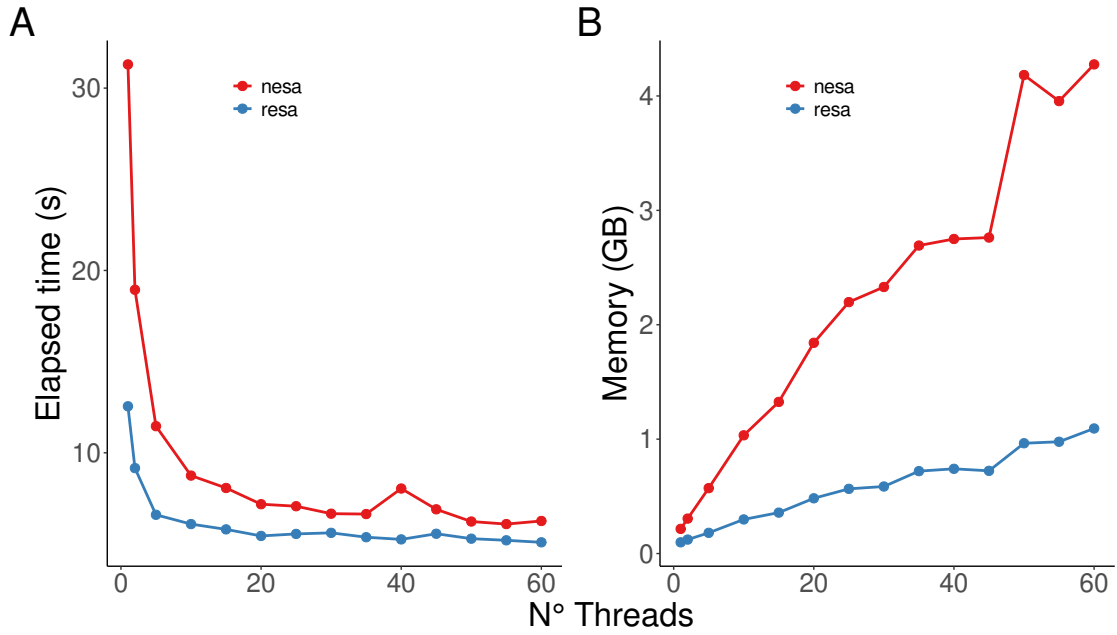


Figure 3.15: Time (A) and Memory (B) consumption *makeFurDb* as a function of the number of threads for a neighborhood of 100 sequences, 1 Mb long.

thread required 22.1% more memory.

3.3.1.3 Accuracy

While efficiency is necessary, it needs to be combined with accuracy to make a good computational tool. To evaluate the accuracy of *fur* I compare the candidate marker regions with blast after the first subtraction step. This ensures the accuracy of the two algorithms are directly compared. While both *nesa* and *resa* are highly accurate, the former tends to be more accurate than *resa*, as shown in Figure 3.16. Regardless of the size of the marker or the applied window size, iteratively indexing the neighbor sequences yields a higher accuracy in detecting the marker regions.

The longer the marker region, the higher the accuracy. For a 200 bp marker region, on average, *nesa* had an accuracy of 0.9 and *resa* of 0.8. For a 1600 bp marker region, *nesa* had an average accuracy of 0.98, while *resa* had an accuracy of 0.96.

3.3.2 Designing diagnostic primers in the large

To evaluate the *Mapro* pipeline, I start by summarizing the number of taxa that completed each step of the pipeline in Figure 3.17: from finding genomes to scoring the designed primers. Then, I explain in detail the results and conclusions that can be obtained from

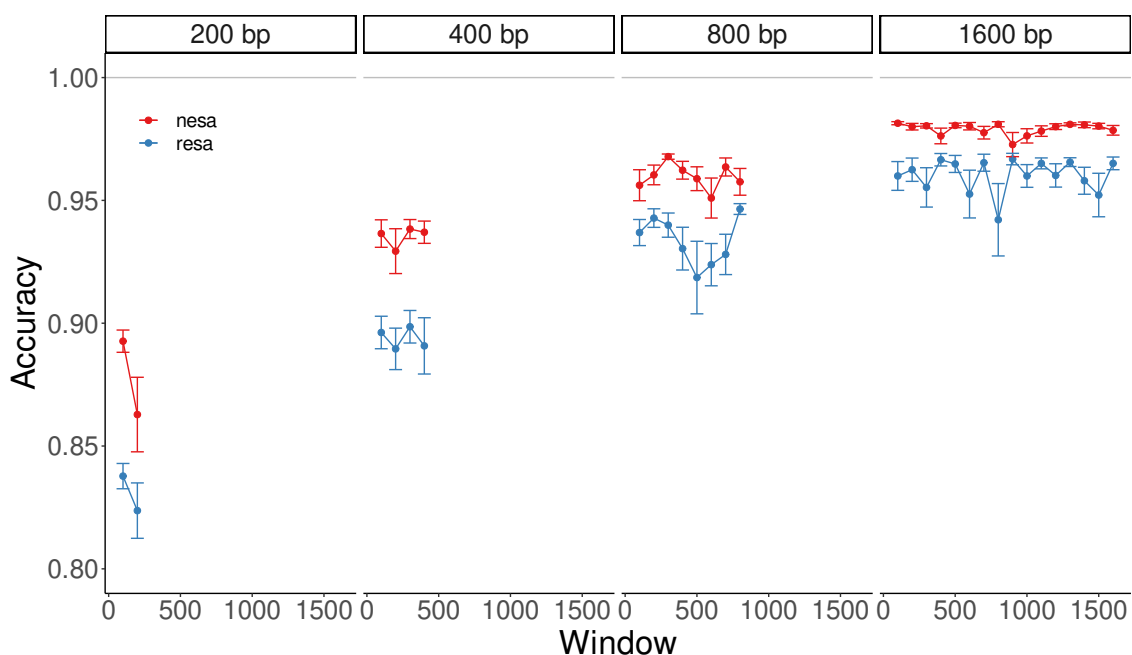


Figure 3.16: Accuracy of the two *fur* algorithms after the first subtraction step for markers ranging from 200 to 1600 bp. The accuracy is shown as a function of window length. Mean \pm SEM of 100 simulations are shown

each pipeline step.

Starting with 120 organisms, 110 of them had at least one available target and neighbor genome assembled to the “complete” level. Of those, 96 had at least one marker region shared across all targets and absent from all neighbors, and I could design a primer pair for 89 of them. Finally, although all of the primers could be scored, it was only possible to correct the primer score for 82 of them. For six primer pairs, the type strain could not be found in the local *nt* database and the score could not be corrected. To correct the primer score for *Bordetella pertussis* the type strain accession was changed from “BX470248.1” to “CP039022.1”, the complete genome accession present in our local *nt* database.

3.3.2.1 Finding genomes

First, I obtained a list of all associated genome accessions for each taxon in our list. At least two genome assemblies must be present: one target and one neighbor. Ten of the 120 reference prokaryotes were excluded from the start because this requirement was not fulfilled.

For seven list entries, the reference type was not found in the neighbors’ database. I manually curated these seven entries to reflect changes in the nomenclature since the

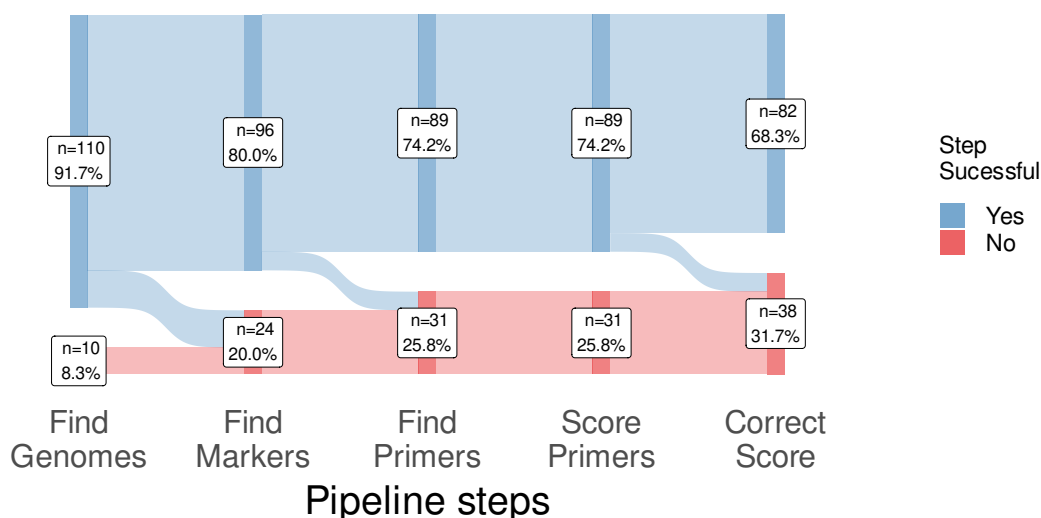


Figure 3.17: Number (and percentage) of prokaryote strains that successfully passed the different steps of the large-scale pipeline

creation of the list in 2019. For example, three of the missing type strains were classified as being *Lactobacillus*. However, a recent reorganization of the *Lactobacillus* genus [136] meant that the original type strain name was no longer correct. The curated names can be found in Supplementary Material SM2.

Figure 3.18 shows the distribution of the number of genomes obtained for each organism. The distribution is heavily skewed, with half of the organisms having less than 99 genomes available, while *Escherichia coli* UMN026 and *E. coli* IAI39 had the highest number of complete genomes: 3190.

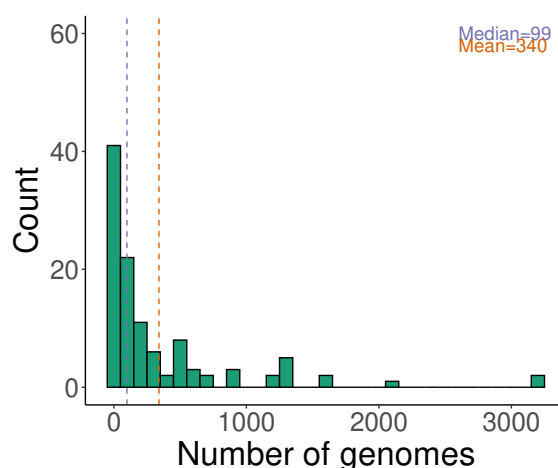


Figure 3.18: Distribution of the number of genomes obtained for each organism

To classify the genomes into targets and neighbors. *fintac* relies on the phylogeny inferred from the pairwise distances calculated with *phylonium*. A perfect split (100%) between target and neighbor sequences as calculated by *fintac* means that all taxonomic targets are contained within the target clade and all the neighbors are located outside this clade. Therefore, the underlying taxonomy is correct.

By looking at the split percentages for each of the 110 taxa with genomes available, I can infer the error rate of the taxonomical information in the NCBI. The majority of the taxa, 61, had a perfect split of 100%. The lowest split percentage was 69.2% for *Vibrio cholerae*, where the target clade consists of the entire phylogenetic tree, hence the neighbors could not be distinguished from the targets. This happens twice more for *Bacillus subtilis* and *Klebsiella pneumoniae*, who have a split score of 78.2% and 90.2%, respectively. Figure 3.19 shows the distribution of the split as a function of the number of available genomes for each organism.

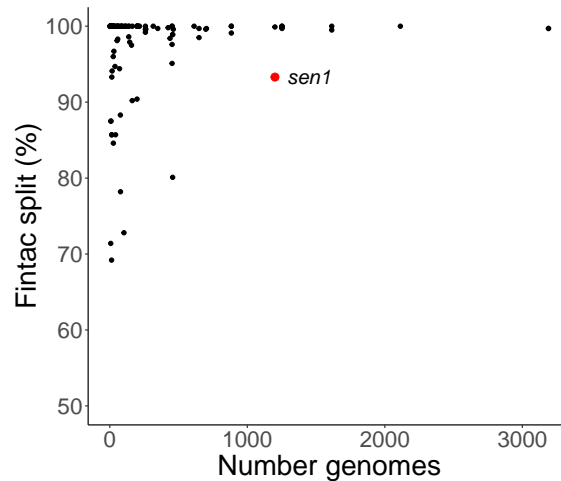


Figure 3.19: *fintac* split score as a function of the number of genomes sampled. In red, the outlier *S. enterica* which contains a lower split score despite the higher number of genomes.

Generally speaking, taxa with more genomes had a better split, but there are some exceptions. One of these exceptions is *Salmonella enterica*, marked in red in Figure 3.19. Despite having 1201 genomes, the split score is relatively low, 93.3%, compared to the other taxa.

I went on to investigate the accuracy of *phylonium* and *fintac* to see whether this lower split score is an artifact or reflects truly misclassified genomes. Since the curation of taxonomical groups is based on ANI, I also use the same measure to classify the similarities between the obtained genomes for *S. enterica* and the type strain accession AE006468.2.

Figure 3.20A shows the calculated ANI for the *S. enterica* genomes grouped according to their taxonomy, while Figure 3.20B shows the variation in ANI according to their phylogeny as calculated by *phylonium*. It can be seen that when the genomes are grouped according to their taxonomical information, there is an overlap in ANI between targets and neighbors. This means that even though some of these genomes are taxonomically distinct (different strains), they are more similar to the type strain and vice versa. If they are grouped genomes according to their phylogenetic grouping as calculated by *phylonium* and *fintac*, all target genomes are more closely related to other targets and vice-versa.

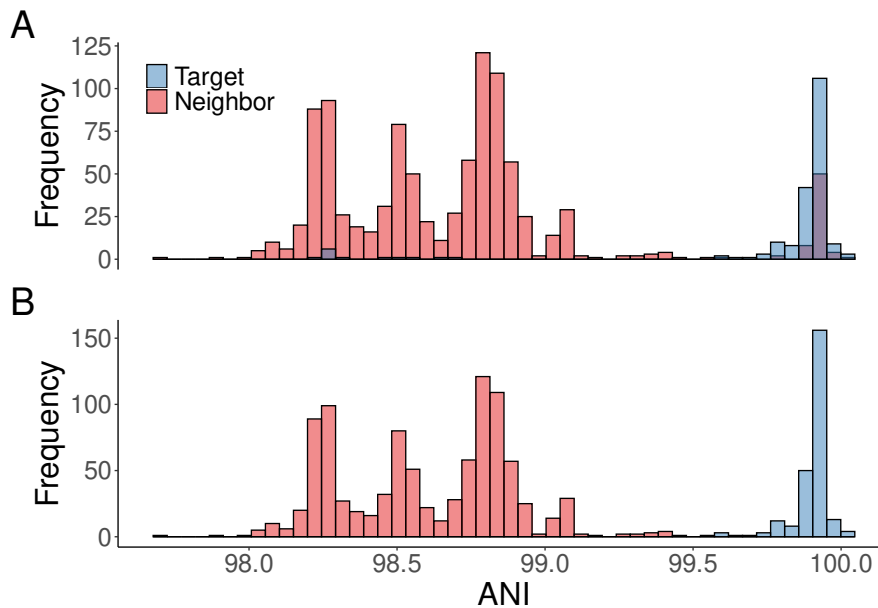


Figure 3.20: ANI of the A) taxonomic and B) phylogenetic targets and neighbors. The type strain for measuring ANI is the reference genome, AE006468.2.

These results suggest that although there has been an effort to curate prokaryotic organism at the species level with ANI, there is still room for improvement.

3.3.2.2 Finding markers

After splitting the downloaded genomes into target and neighbor sets, I run *Fur* to search for marker regions. As mentioned above, I could find at least one marker region for 96 of the taxa, with the number of detected marker regions varying widely. Figure 3.21A shows the distribution of the number of markers found for each taxon. On average, 420 marker regions are detected across all organisms, but less than 111 marker regions could be detected for half of them.

The distribution is heavily skewed with *Rhodospirillum rubrum* having the highest num-

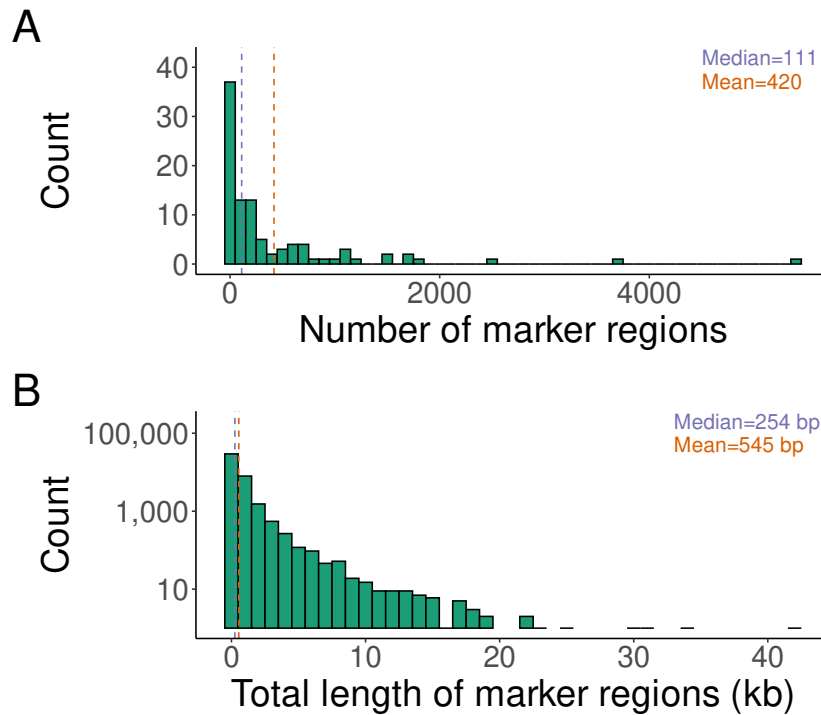


Figure 3.21: Distribution of the number of markers (A) and their length (B) obtained for each organism

ber of marker regions: 5376. The marker regions for *R. rubrum* comprise a total of 2.9 Mb, or 66.8% of the genome. This is most likely because the analysis for *R. rubrum* was comprised of just four genomes, only one of which was classified as a neighbor. In this case, since there is only one neighbor, the more distant they are from the targets, the higher the proportion of the target genome that is considered to be unique. In Figure 3.22 it can be seen that the three target genomes are extremely similar while the neighbor sequence is quite distinct with an average pairwise distance of 0.2.

For this analysis, only genomic regions that are at least 100 bp long are considered to be marker regions. Figure 3.21B shows the distribution of the obtained marker length. Its distribution is also skewed, with the average marker region being 545 bp, but with a median of less than half of that, 254 bp. The longest marker region is 41.5 Kb and belongs to *Rhodopirellula baltica* where only one target and one neighbor genome, with a pairwise distance of 0.1, were compared. As explained above, if the phylogenetic distance between the targets and neighbors is relatively large, then a higher number of marker regions may be found. Additionally, since only two genomes are available, there might be a sampling gap in our dataset, where a closer neighbor might be missing. In this case, the marker regions for *R. baltica* comprise 22.2% of the genome.

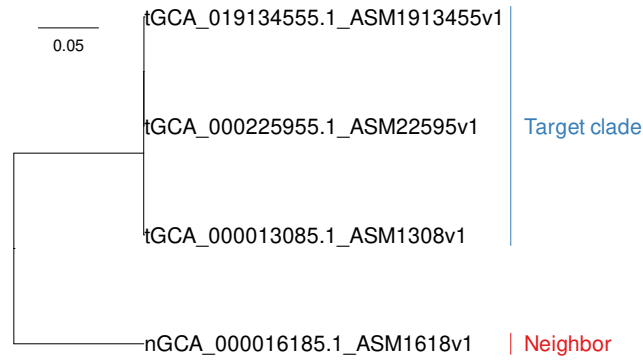


Figure 3.22: Phylogenetic tree for *R. rubrum*. Tip labels are the genome accessions. The tree is rooted at the midpoint.

3.3.2.3 Finding and scoring primers

After obtaining the marker regions, the next step in the workflow is to design primer pairs and evaluate their accuracy. The designed primer pairs vary between 17 and 22 bp with the majority being 20 bp long (94.7%). Primers designed with *primer3* return a primer pair penalty score that evaluates how stringent the requirements for calling a primer are [137]. The minimum observed primer penalty is 1.4×10^{-4} , and occurs five times for *Amycolatopsis mediterranei*, *Chloroflexus aurantiacus*, *Desulfovibrio vulgaris*, *Moorella thermoacetica* and *R. baltica*. While the maximum primer penalty is 5.3 for *Bordetella parapertussis*. On average, the primer penalty is 0.3 and the median penalty is 0.02, showing that the distribution is again highly skewed.

Finally, I scored primers according to their sensitivity and specificity. 29 primer pairs had a perfect sensitivity and specificity score of 1, after running *scop*, but this increased to 46 primer pairs after running *cops* to account for misclassified sequences.

Figure 3.23 shows the distribution of the sensitivity (3.23A) and the specificity (3.23B) calculated with *scop* and *cops*. On average, primers evaluated by *scop* had a sensitivity of 0.9 and a specificity of 0.8, but after correcting with *cops* the average sensitivity increased to 0.98 and the average specificity to 0.93. In either case, as mentioned above, more than half of the primer pairs had a perfect score of 1 for both specificity and sensitivity.

It was not possible to correct the score for seven taxa. This is because our local *nt* database did not contain a complete genome assembly for the respective taxon, either because it was missing or fragmented into several accessions. For *M. thermoacetica*, specifically, it is because the type strain accession “CP000232.1” has been suppressed.

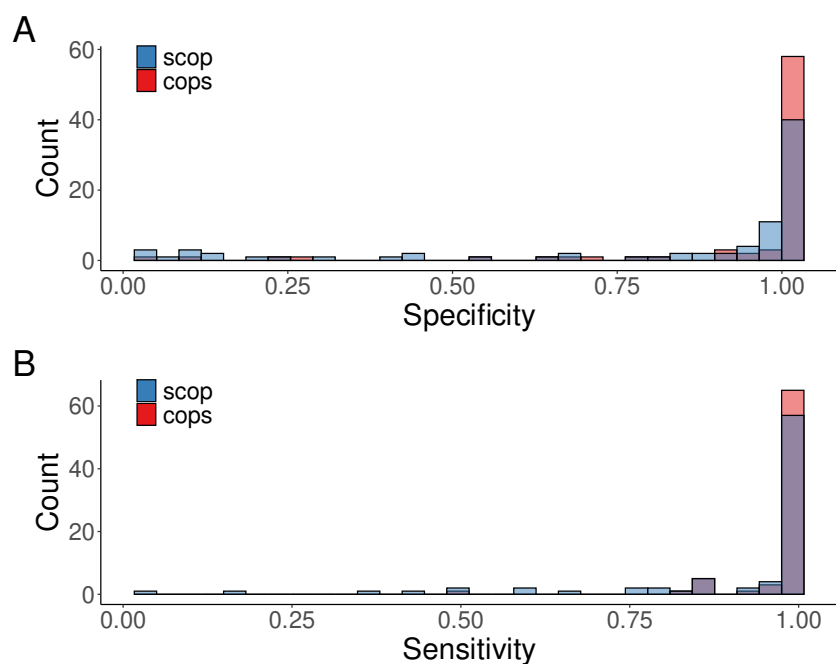


Figure 3.23: Distribution of sensitivity (A) and specificity (B) scores for both *scop* and *cops*.

For a complete list of all designed primer pairs and their score, please see Supplementary Material SM2.

Repeated primer pairs

Because in the list of 120 reference prokaryotes, some belong to the same *species*, I check whether there is any repeated set of primers. If that is the case, the target representative must be the same. Eight of the 89 primer pairs were repeated, as shown in Table 3.2. These are the primer pairs belonging to *Chlamydia trachomatis*, *Clostridium botulinum*, *Caulobacter vibrioides* and *Bacillus anthracis*.

Table 3.2: Repeated forward primers

Count	Forward Primer	Strains
2	TATTCTATTTGCTGCGAGCC	<i>C. trachomatis</i> D/UW-3/CX and 434/Bu
2	TAGCACACCCAGTTAAATGC	<i>C. botulinum</i> A str. ATCC 3502 and str. Hall
2	GCCGATATACATCAAGCAGC	<i>C. vibrioides</i> CB15 and NA1000
2	CGCCATCGCTAAAAGGAATA	<i>B. anthracis</i> str. Ames and str. Sterne

Checking the other taxa with non-repeated primer pairs, shows that *E. coli* UMN026 and

E. coli IAI39 also share the same target set, with all *E. coli* genomes set as the target and all other *Escherichia sp.* strains as the neighbors, explaining why they have the highest number of associated genomes. Because in these two strains no marker region was found, they do not appear in Table 3.2.

Previously, I had climbed two taxonomical ranks when defining our target and neighbor sets. This is because I assumed that the type strain would have its own taxonomical ID and that all associated assembly strains would share the same parental node. However, this taxonomical structure has been retired since 2014 [125], and may not necessarily be applicable to these five type strains.

For example, both *E. coli* UMN026 and *E. coli* IAI39, are classified as “strains” and share the same parent taxon ID (562) *E. coli*. Thus, by giving the parent taxon ID to *neighbors* as the target, all the genome associated with *Escherichia coli* will be included in the target set, and all the other *Escherichia sp.* as the neighbors. In contrast, the parent taxa ID for the type strain “*Escherichia coli* O157:H7 str. Sakai” is the serotype “*Escherichia coli* O157:H7”. In this case, I am searching for marker regions that distinguish the serotype “*Escherichia coli* O157:H7” from other *E.coli* strains. Figure 3.3 shows an example of the two taxonomic naming systems.

The question now arises of what would happen if I set the type strain as the target taxon ID for these 10 cases. Thus, I repeat the entire analysis of these ten strains using this scenario.

The number of target genomes in these taxa is now decreased to one in nine strains, with only *Bacillus anthracis str. Sterne* having two target genomes. Only three primer pairs could be designed: *C. botulinum A str. ATCC 3502*, *C. vibrioides NA1000* and *E. coli* IAI39. The new primers for *C. botulinum A str. ATCC 3502* have exactly the same sensitivity (0.9) and specificity (1). The primers for *C. vibrioides NA1000*, which previously had a perfect score, now have a lower specificity (0.7), meaning they might amplify regions from non-target organisms. The decrease in specificity suggests that the two *C. vibrioides* strains are extremely similar. Figure 3.24 compares the target clades from the two analysis. In the first analysis *C. vibrioides* was set as the target clade (in blue). This clade had a deeper split, 0.05, from the neighbor sequences (in black). However, in the second analysis, *C. vibrioides NA1000*, the red accession, was set as the target. It can be seen, that this accession is extremely close to that of its neighbor accession “tGCA_000006905.1_ASM690v1” with a pairwise distance of 3.7×10^{-7} . Hence explaining why the specificity of this second pair of primers is so low.

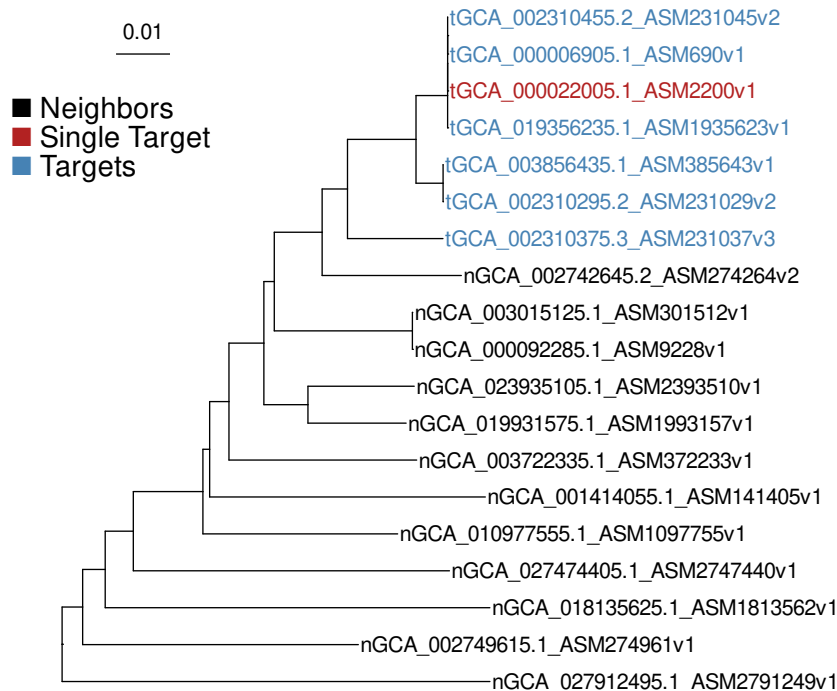


Figure 3.24: Phylogenetic tree for *C. vibrioides*. Target clade from the first analysis is marked in blue, while the single target genome from the second analysis is marker in red.

Finally, I was interested in looking at the score for the *E. coli* IAI39 primer since in the previous analysis, no marker regions were found. However, while this primer contains a perfect sensitivity (1), the specificity is almost zero (0.1), showing the difficulty in distinguishing the *Escherichia coli* IAI39 strain from other *E.coli* organisms.

Additionally, I note that it is not possible to distinguish *B. anthracis* str. Ames from *B. anthracis* str. Sterne, nor *C. trachomatis* D/UW-3/CX from *C. trachomatis* 434/Bu since no primer pair was found.

3.3.2.4 Amplicon annotation

After obtaining and evaluating a primer pair, I check whether its amplicon intersects any biological function. The amplicons were annotated with the GFF3 file from the respective reference accession type for 87 taxa, Figure 3.17.

I then summarise the annotated features intersected by the amplicon. For six taxa, *A. mediterranei*, *Bifidobacterium longum*, *Burkholderia mallei*, *Prochlorococcus marinus*, *Streptomyces coelicolor* and *Thermosynechococcus vestitus*, the amplicon does not intersect any annotated feature.

Table 3.3A lists the frequency of the annotated features across all taxa. 81 genes and CDS were annotated. However, equality of the gene count and CDS count is a coincidence. For example, the amplicon for *C. trachomatis* 434/Bu intersects a pseudogene and its CDS region, while the amplicon for *Rickettsia prowazekii* intersects a gene and a pseudogene (sequence feature) but no CDS.

There are 12 taxa, including *C. trachomatis* 434/Bu, where the amplicon intersects no gene, and four, where the amplicon intersects two. The remaining 73 amplicons intersect a single gene and its respective CDS region.

Table 3.3: The most common amplicon sequence features (A) and CDS products (B).

A		B	
Count	Feature	Count	Product
81	Gene	24	Hypothetical protein
81	CDS	11	Conserved hypothetical protein
8	Sequence feature	3	Conserved domain protein
2	Repeat region		
1	Signal peptide region of CDS		
1	Pseudogene		

I also analysed the CDS products. Of the 81 intersected CDS regions, 23 (28.4%) of them code hypothetical proteins, 11 (13.6%) for conserved hypothetical proteins and 3 (3.7%) for conserved domain proteins as shown in Table 3.3B. The remaining 43 CDS products are associated with transcription regulators, putative transcription factors and several metabolic processes. For a full list of all CDS products and the respective taxon in which it was annotated, please see Supplementary Material SM2.

3.3.2.5 Marker regions and gene function

I restricted the search for functional enrichment in marker regions to the top 10 taxa with the highest number of target genomes for which primers could be designed (Table 3.4).

Of the ten most sequenced reference taxa (Table 3.4), only three were also contained in the GO database, *P. aeruginosa*, *M. tuberculosis* and *S. pneumoniae*, so I restricted the enrichment analysis to these. However, the annotated genes for *M. tuberculosis* and *S. pneumoniae* are not contained in the GAF files, and thus, the enrichment analysis could only be performed for *P. aeruginosa*. This occurs because the obtained GAF files are not complete. Genes with an unknown function or whose function is not curated will not be present in the GAF files.

Table 3.4: Number of target (n_t) and neighbor (n_n) genomes for the top ten most sequenced referenced prokaryotes with a primer pair. Ordered by the number of target sequences.

Species	Type strain	n_t	n_n
<i>Pseudomonas aeruginosa</i>	PAO1	675	23
<i>Bordetella pertussis</i>	Tohama I	659	225
<i>Mycobacterium tuberculosis</i>	H37Rv	437	1
<i>Enterococcus faecium</i>	DO	332	317
<i>Lactiplantibacillus plantarum</i>	WCFS1	247	15
<i>Enterococcus faecalis</i>	V583	228	421
<i>Streptococcus pneumoniae</i>	R6	178	1074
<i>Staphylococcus epidermidis</i>	ATCC 12228	154	1460
<i>Salmonella enterica subsp. enterica</i>	Typhi str. CT18	151	1050
<i>Burkholderia pseudomallei</i>	K96243	141	61

Table 3.5 lists all terms with at least two associated genes. This is a very low threshold compared to the 10 genes used in the enrichment analysis described in Chapter 2, Section 2.2.5. Still, if I take this approach, I find seven terms with occupancy ranging between 2 and 4. Six of these terms are “biological processes”, one, “Extracellular space” (GO:0005615) is a “cellular component”.

Of all described processes, only "cellular response to phosphate starvation" is significantly enriched with two observed genes. In fact, there are only two genes associated with this category described in the GAF file.

Table 3.5: List of biological processes and observed (Obs) annotated genes, associated with marker regions in *Pseudomonas aeruginosa* PAO1, and the corrected P -value.

GO ID	Obs	Ratio	P -value	Description
GO:0016036	2	17.3	0.02	Cellular response to phosphate starvation
GO:0051872	2	5.7	n.s.	Sphingosine catabolic process
GO:0043952	3	3.6	n.s.	Protein transport by the Sec complex
GO:0015628	4	2.4	n.s.	Protein secretion by the type II secretion system
GO:0005615	2	2.3	n.s.	Extracellular space ^a
GO:0071978	2	1.3	n.s.	Bacterial-type flagellum-dependent swarming motility
GO:0044010	2	0.6	n.s.	Single-species biofilm formation

^a Cellular Component

3.3.2.6 Runtime and resources

The three steps of *Mapro* shown in Figure 3.4 are divided into two stages: primer design and primer checking. In the first stage, I create the target and neighbor sets with *Neighbors*, obtain the marker regions with *Fur* and design the primer pairs with *Prim*. This step required 9 h 29 min and 30.3 GB of memory. In the second stage, I scored the designed primer pairs with *scop* and *cops* from *Prim*. The scoring of the primers was slightly faster, requiring 8 h 38 mins, but had a much higher memory consumption of 260.2 GB, caused by the Blast runs on *nt*.

Calculating the pairwise ANI for the 1201 *S. enterica* genome assemblies with *fastANI* required 7h 25 min and 15.3 GB on 64 threads. As a comparison, running *phylonium* on the same data set, also with 64 threads, required 1 min 06 s and 7.5 GB. This means that *phylonium* was 400 times faster than *fastANI* at calculating all pairwise distances while requiring half the memory.

3.4 Discussion

In this chapter, I designed and implemented the pipeline *Mapro* for automatic marker discovery in prokaryotes. This pipeline integrates three kinds of software (Figure 3.4): (1) software for finding markers for a target organism by comparing its genomes to those of its close relatives, the neighbors; (2) software to find the input to (1), all available target and neighbor genomes; and (3) software to assess the output of (1), the markers. For the central software, I considered *Uniqprimer*, *Kec* and *Fur*. Out of these three, *Fur* had the most mature implementation. However, it used too much memory to analyze samples of hundreds of bacterial genomes in this chapter, let alone lineage-specific regions in mammals as discussed in the next chapter.

As to finding the input to marker discovery and checking its output, the programs that eventually coalesced into the packages *Neighbors* and *Prim* still had to be designed and tested. They are not specific to *Fur* and could in the future be used in the context of a different marker detection tool, should one become available.

To make *Fur* fit for large-scale diagnostic marker finding, I revisited the algorithm in order to reduce the memory consumption of this first step. To do so, I here described two iterative *Fur* algorithms, *nesa* and *resa*, Algorithm 3 and Algorithm 4, respectively. The advantage of changing to an iterative algorithm is that first, the memory requirement becomes constant depending on either the length of the longest neighbor sequence (*nesa*)

or the length of the target representative (*resa*), and secondly, as I show here it can also be parallelized, further reducing the runtime.

3.4.1 *nesa* and *resa*

The two *Fur* algorithms are very similar regarding resource consumption and runtime when analyzing a single neighbor sequence, regardless of its length, as shown in Figure 3.12. The main difference occurs when there are multiple neighbor sequences, where indexing a single target sequence *r* (*resa*, Algorithm 4) and streaming the neighbors is faster and consumes less memory than indexing each neighbor *n* (*nesa*, Algorithm 3) and streaming *r* (Figure 3.13). For a single neighbor sequence, the memory consumption between *nesa* and *resa* is identical, but as the neighborhood increases in size, *resa*, on average, uses 60% the memory of *nesa* to construct the database. Likewise, in terms of user time, *resa* tends to be twice as fast.

The reduction in memory and time with *resa* is because now it only calculates the enhanced suffix array once, for the target representative. This difference also affects their behaviour during parallelization. Because *nesa* constructs an index for every neighbor sequence, it benefits the most by having additional threads when compared to *resa* which only needs to stream the neighbor sequences. However, because it is constructing multiple indexes at the same time, the additional memory consumption per thread for *nesa* is higher than for *resa*.

Another difference between the two algorithms is their accuracy. Generally speaking, both algorithms identify larger marker regions more accurately than smaller ones. For example, for a 1600 bp long marker, *nesa* had an accuracy of 0.98 across all tested windows, while *resa* had an accuracy of 0.96 (Figure 3.16). In practical terms, this means that on average, *nesa* missed the truth by 32 bp while *resa* missed it by 64 bp. For a short marker region of 200 bp, *nesa* had an average accuracy of 0.88, while *resa* had an accuracy of 0.81. Once again, this means that *nesa* missed the truth by 24 bp while *resa* missed by 38 bp.

Because the number of mismatched bp is similar across marker lengths, there is the possibility that this might be the error margin of *Fur* itself, and shorter regions are penalized due to this.

For the time being, I decided to continue with *nesa* due to the slightly greater accuracy. However, the slight difference in accuracy, together with the reduction in memory and runtime and the lower memory penalty when multithreaded, makes *resa* an attractive

alternative, especially if future work could further improve its accuracy.

3.4.2 Large-scale marker discovery

In this chapter, I went through a list of 120 prokaryote reference strains which was last updated in 2019. Since then, several organisms or strains have been reclassified or even suppressed by the NCBI. So, it was necessary to curate the input list.

I also mentioned previously that the NCBI taxonomy database changed its naming system in 2014. The fact that the changes were not retroactively applied means that I am dealing with two different classification systems, depending on when a given strain was first classified. In the future, one might be able to accommodate this by parsing the number of taxonomical ranks between the given strain and the *genus*. This could be done, for example, with *ants* (Neighbors), which lists the ancestors of a given taxon.

As seen in Figure 3.18, the number of available genomes varied widely between taxa. However, more important than the number of available genomes is how they are classified in the taxonomical database. With *phylonium* and *fintac*, it is possible to assess the error rate of the taxonomical classifications. The split between targets and neighbors estimated with *fintac*, as a function of the number of genomes sampled, follows a curve where the higher the number of genomes, the lower the misclassification rate. This pattern may occur because highly sequenced prokaryotes have a higher biomedical importance, and thus their classification is under higher scrutiny.

Using *S. enterica* as an example, I investigated the split given by *phylonium* and *fintac*. For this, I started by comparing all *S. enterica* sequences to the type strain and comparing the taxonomical and phylogenetic targets and neighbor groups to each other. The ANI results confirm the *phylonium* estimate, where sequences that belong to the same phylogenetic clade also have a higher ANI. The matrix output from *fastANI* was reformatted into a distance matrix and from it I constructed a phylogenetic tree using the neighbor-joining algorithm (*nj*). Applying *fintac* on the neighbor-joined tree from *fastANI* yielded three target clades, all with the same split score of 87.3%. The first target clade, which has the greatest distance to the parents, was composed of the same genome accessions as the *phylonium*-based tree. Thus, *phylonium* plus *fintac* could also be used to classify strains in the taxonomical database. It can quickly resolve possible taxonomical conflicts because it is orders of magnitude faster than *fastANI*.

Regardless of the method used to calculate phylogenetic distances, I reiterate the importance of having a taxonomically correct database at all levels. For example, when scoring

the primer pairs initially, only 29 of them had a perfect score. If, instead, the primers are scored based on their phylogenetic distance, 46 of them have a perfect score. If I were to rely exclusively on the taxonomy, I would risk rejecting 17 perfect primer pairs with a lower score due to taxonomical misclassification. By applying this correction to the Mapro results, I verify that 51% of the primer pairs Mapro identified have perfect *in silico* sensitivity and specificity.

The advantage of using whole genomes for finding markers is that it is not restricted to hypotheses regarding the biological functions of a given strain. Still, it remains interesting to investigate where the amplicons are located and whether they intersect any known function or genes associated with their pathogenicity. However, the most common intersected CDS has no known function, so if I were to rely on functional hypotheses, these regions might not have been considered. Half of the hypothetical proteins, 5 of the conserved hypothetical proteins and the three conserved domain proteins were located in the taxa with a perfect primer score, emphasizing the importance of hypotheses-free marker discovery.

Amplicons are but a small portion of the detected marker regions. Thus, I am also interested in whether the detected marker regions are functionally enriched. I proceeded to do an enrichment analysis for *Pseudomonas aeruginosa*, an “opportunistic” pathogen that tends to infect patients with a compromised immune system. In patients with cystic fibrosis, the infection often occurs early in life and can result in a progressive loss of lung function up to death [138].

The marker regions for *P. aeruginosa* comprise 65 Kb and intersect 221 unique genes. The enrichment analysis allows us to identify functions that might be shared across the annotated genes. The only significant enrichment ratio is 17.3 for “cellular response to phosphate starvation” (Table 3.5). Bacterial pathogens are known to regulate virulence factors in response to environmental stimuli, such as nutrient starvation. Phosphate is a nutrient essential for cell functions and as such, it is involved in many biochemical paths. Moreover, its presence (or absence) influences the expression of virulence trains in many bacterial species [139]. In *Pseudomonas aeruginosa*, phosphate starvation can lead to the production of toxins like pyocyanin [140, 141], thus being associated with an increase in pathogenicity [142].

The two observed genes associated with phosphate starvation are “ppx” and “PA4351”. “ppx” is an exopolyphosphatase that produces factors associated with acute and chronic infections [143] and is also associated with “bacterial-type flagellum-dependent swarming motility” and “single-species biofilm formation”, both also present in the list of marker-

associated processes in *P.aeruginosa* (Table 3.5). In fact, both the ability of *P. aeruginosa* to create a biofilm and its motility are involved in important virulence factors [138, 144]. The second gene “PA4351”, or *olsA*, is also associated with phosphate starvation because it produces a phosphate-free ornithine lipid (OL) [145].

The process with the highest number of associated genes is the "protein secretion by the type II secretion system" (Table 3.5), a two-step process in which proteins are translocated across the inner membrane by the Sec or Tat pathway [146]. The first of these was also observed in the enrichment analysis with three associated genes. The type II secretion system is also associated with pathogenicity since it is present in many pathogens and it causes damage to host cells and tissues [146]. In *P. aeruginosa*, the type II secretion system, has been shown to cause death in lung infections [147]. Finally, the observed genes associated with “Sphingosine catabolic process” are *sphA* and *sphR*, which influence the survival of the bacteria in the mouse lung [148].

The pathogenicity of *P.aeruginosa* can be attributed to several factors such as its ability to form multicellular biofilms, exopolysaccharides, which help resist stressors and flagella that are important for swimming [138, 144], all processes that were found in our markers regions, meaning that all of the biological processes found for *P. aeruginosa* are associated with pathogenicity. Thus, in the future, it would be interesting to know the function of the hypothetical proteins located in the amplicons, as they might provide clues regarding pathogenicity mechanisms.

3.5 Summary

Fur is the most mature program available for marker discovery from genome sequences. In this chapter, I aimed to not only improve *Fur* for large-scale diagnostic marker discovery but also to developing a pipeline that can accurately identify sets of target and neighbor sequences for arbitrary bacterial taxa.

I started by improving the *Fur* algorithm which, although accurate, had unfeasible memory requirements. To do so, I compared two iterative *Fur* algorithms: *nesa*, which iteratively indexes and compares every neighborhood element, N , and *resa*, which indexes the target representative and iteratively streams the neighbor sequences. Although *resa* uses half the memory and is twice as fast as *nesa*, it is slightly less accurate than *nesa* at identifying marker regions. Because both are iterative algorithms, the memory requirements are now constant and dependent on the length of the longest neighbor (*nesa*) or the length of the target representative (*resa*). Furthermore, due to their iterative properties,

both algorithms can be parallelized, further decreasing their runtime.

In order to apply the optimized *Fur* to large-scale marker discovery, I constructed the *Mapro* pipeline. In addition to marker discovery with *Fur*, *Mapro* also contains modules for discovering phylogenetically correct targets and neighbors and constructing and evaluating primers. I applied *Mapro* to a list of 120 prokaryotes reference strains to evaluate its efficiency in searching for marker regions and designing diagnostic primer pairs. I designed primers for 74% of the taxa analyzed, 38% of which have perfect accuracy. This level of accuracy is due to *Neighbors*, which is crucial to finding monophyletic targets and their neighbors. I also show that with *phylonium* it is possible to quickly reclassify highly similar strain, resulting in clades that are also supported by the established ANI method.

When looking at the function of the obtained amplicons, a high proportion of them code for hypothetical proteins, showing the importance of hypotheses-free marker discovery from whole genome sequences. After performing an enrichment analysis on *P. aeruginosa*, all biological processes associated with the marker regions could be associated with pathogenicity. These results further suggest that studying the function of marker regions detected by *Fur* might yield interesting insights regarding pathogenic traits of a given strain.

Chapter 4

Lineage-specific regions in mammalian genomes

Beatriz Vieira Mourato¹ and Bernhard Haubold¹

Affiliations:

¹Max Planck Institute for Evolutionary Biology, RG Bioinformatics, Plön, Germany

4.1 Introduction

In Chapter 3, I described a pipeline for hypothesis-free detection of diagnostic markers in bacteria. The main idea was to compare a set of target genomes to their closest distinct relatives, the neighbors, and from there identify regions shared across all targets, but absent from the neighbors. The target-specific regions detected by *Fur* [21, 115] can then be considered potential diagnostic markers. To assess their sensitivity, I performed an *in silico* PCR test against the non-redundant collection of nucleotides, *nt*. This extra step ensured that the final markers are truly absent from all sequenced organisms.

The regions given by *Fur* are target-specific insertions in the sense that they are absent from the neighbors. However, as the mechanism upon which they originated is unknown, they may just as well have been deleted from the neighbors. Still, since our focus is on target-specific regions, they are henceforth called target-specific insertions for ease of understanding.

The simplicity of comparing genome A to genome B to obtain a list of target-specific insertions suggests that *Fur* can be used in contexts beyond marker discovery. There is a long tradition of searching for lineage-specific indels that can explain the genetic underpinnings of species-specific traits [2]. Strictly speaking, lineage-specific indels comprise lineage-specific insertions and lineage-specific deletions. However, lineage-specific *regions* can be either ubiquitous target insertions or ubiquitous neighbor deletions, and *Fur* is an efficient tool for detecting them even on the scale of mammalian genomes, which are roughly one thousand times longer than the bacterial genomes I analysed on chapter 3. Here I demonstrate the suitability of *Fur* for such large-scale analyses by applying it to the genomes of great apes and mice.

4.1.1 Humans and chimpanzees

When the first chimpanzee genome was compared to the human genome approximately 40-45 Mb of the human genome was classified as lineage-specific [2]. Although no reverse comparison was performed, the assumption was that a similar proportion of the chimpanzee genome would be considered lineage-specific, corresponding to a total of 90 Mb of lineage-specific regions between both species. Since humans are the only extant species of the *Homo* genus, the closest living relatives of humans are the common chimpanzees, (*Pan troglodytes*). However, some regions in the human genome are closer to the chimpanzees while others are closer to the gorillas [149, 150]. This incomplete lineage-sorting is due to fact that the human-chimpanzee split occurred shortly after the gorilla

split [151]. Thus, even though the chimpanzee is the closest living relative of humans, in approximately 23% of the human genome, the gorilla is the closest genomic relative [152].

Taxonomically speaking, humans and chimpanzees form the Hominini tribe, which expands to the Homininae subfamily when including the gorillas. Figure 4.1 shows the relation between the different ape species and their taxonomic groups.

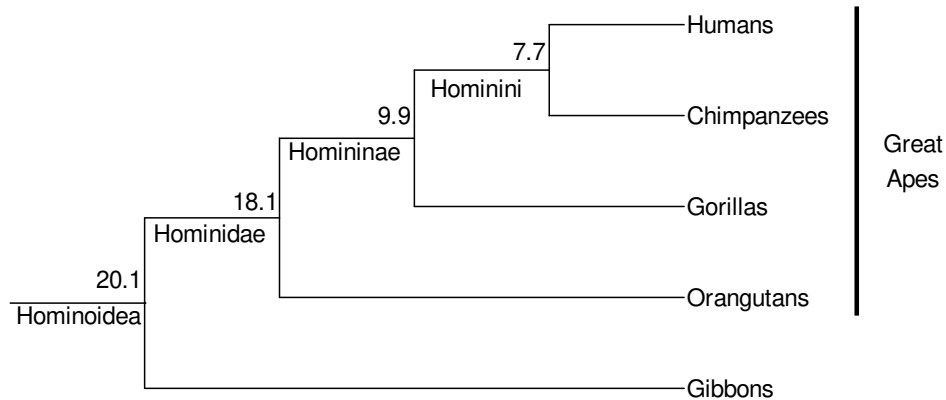


Figure 4.1: Cladogram showing the common names and taxonomic ranks of extant ape species and estimated divergence dates in millions of years before present. Divergence times based on estimates from Shao *et al.* [153].

The first two human genome sequences were released in 2001 and were constructed based on two distinct sequencing strategies. One relied on mapped BAC (bacterial artificial chromosomes) clones while the other relied on whole genome shotgun sequencing [31]. Since then, whole genome shotgun sequencing has become the norm. However, there have been massive improvements to the sequencing technologies available, from short-reads, of between 150 to 800 bp, to long-reads, which can be up to 4 Mb long [154]. The current human reference genome GRCh38, released in December 2013, has since been patched 14 times, last in May 2022. Even so, it still contains 349 gaps between scaffolds and 161.6 Mb of unsolved regions, mostly in satellite and centromeric regions. In 2022, the Telomere to Telomere (T2T) consortium released the first truly complete human genome [41] filling the remaining 8% gaps in the human reference genome. Thus, for the first time, it is possible to also analyse centromeric regions in a genomic context.

By finding species-specific regions, I can identify the presence of species-specific genes and their respective functions. In the previous human-chimpanzee comparison, while there was the expectation that most regions are neutrally evolving, they also detected

genes in the human genome involved in apoptosis, inflammatory response, parasitic response, and sialic acid biology [2]. The question is whether analysis of the current assemblies leads to the same conclusions.

The initial human and chimpanzee comparison was a 1-to-1 comparison between reference genomes [2]. Nowadays there are at least 11 high-quality human genomes available that can be included in these comparisons. Since a single reference genome cannot represent the genomic diversity present in the population [155, 156], I include all 11 genomes in the comparison. Additionally, I take into account the incomplete lineage-sorting present in the Homininae subfamily and expand the search for human-specific regions by excluding regions present in the gorilla genome.

In my search for human-specific regions, I compare the human genomes to both chimpanzees and gorillas. Although the initial available genomes for both Chimpanzee and Gorilla were of low quality, recently the Telomere-to-Telomere Primate project released new ape assemblies that were sequenced using the same technology as the complete human genome [43]. In other words, the available reference genomes for these three species are now of very similar quality. For example, the common chimpanzee reference PanTro3-v2.1, is fully resolved at the chromosome level with no gaps, while the gorilla reference contains two gaps of 1 Mb each, in chromosomes 22 and 23 [43].

4.1.2 The house mouse superspecies complex

As the first two mammalian genomes to be sequenced, the human and mouse genome were compared to each other in 2002 with the aim of finding similarities and conserved regions that could give insight into the evolutionary history of mammals [9]. Comparative studies between these two species have shown that GO categories associated with the most divergent regions between Primates and Murines were related to spermatogenesis and metabolic processes [2]. Since my aim is to search for species-specific traits, ideally these comparisons should be made between closely related organisms rather than across mammalian orders. Fortunately, the genomics of mice have been extensively studied since the publication of the first mouse genome, and thus there are several *Mus* genomes available.

Figure 4.2 shows the Eurasian and Asian clades of *Mus* species, with the three *M. musculus* subspecies clustering together at the top of the three in the Eurasian clade. Their closest relatives are *M. spicilegus*, *M. macedonicus* and *M. spretus*, also part of the Eurasian clade. While *M. cooki*, *M. cervicolor* and *M. caroli* are part of the Asian clade.

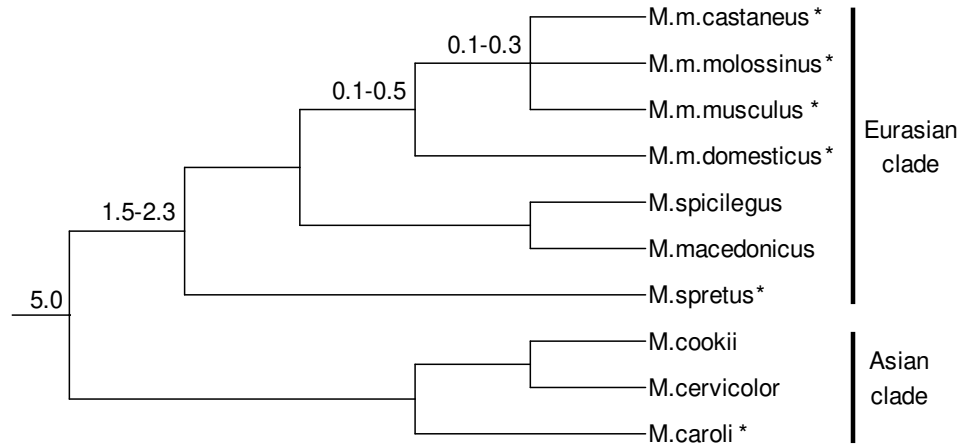


Figure 4.2: Cladogram showing ten species of the *Mus* genus. Numbers above nodes are estimated divergence dates in millions of years before present. Based on the phylogenetic tree from Runck *et al.*, 2009 [157] and the phylogeny shown in Figure 4.12. Divergence time for *M. m. domesticus*, *M. m. musculus* and *M. m. castaneus* based on estimates from Phifer-Rixey *et al.* [158]. The species studied in this chapter are marked with an asterisk.

The commonly denominated house mouse, *Mus musculus*, is a superspecies complex of three primary subspecies: the western european house mouse, *M. m. domesticus* (DOM), the eastern european house mouse, *M. m. musculus* (MUS) and the southeastern Asian house mouse, *M. m. castaneus* (CAS). These three subspecies are not reproductively isolated, and their hybridization is known to have originated other subspecies. The japanese wild mouse, *M. m. molossinus*, is an example of a subspecies whose origin was traced to an hybrid contact between *M. m. castaneus* and *M. m. musculus* [159]. It is estimated that *M. m. domesticus* was the first subspecies to diverge between 130 to 500 thousand years ago, followed by *M. m. castaneus* and *M. m. musculus* between 110 and 320 thousand years ago [158, 159]. Despite their short divergence time, the average nucleotide diversity for each of these subspecies is quite high, from 0.2% (*M. m. domesticus*) to 0.5% (*M. m. castaneus*), which is much higher than what can be found in humans (0.08 to 0.12%) [159].

Before searching for lineage-specific regions between the three subspecies of *M. m. spp.*. It should be noted that the available genomes for *M. m. domesticus* and *M. m. castaneus* are marked by the NCBI to be of “atypical” length. The genome for *M. m. castaneus* standing at 3 Gb long is considered to be too large, while the genome for *M. m. domesticus* at 2.5 Gb is considered to be too short (Table 4.3). As a reference the reference genome for the house mouse is 2.7 Gb long.

The use of genomes with atypical length can potentially lead to an over/underestimation of the obtained regions. As demonstrated in Figure 4.3, short genomes may lead to an over estimation of the number of unique regions (Figure 4.3B), while genomes that are longer than expected, if they contain extra sequences from the target organism, lead to removal of true unique regions (Figure 4.3B).

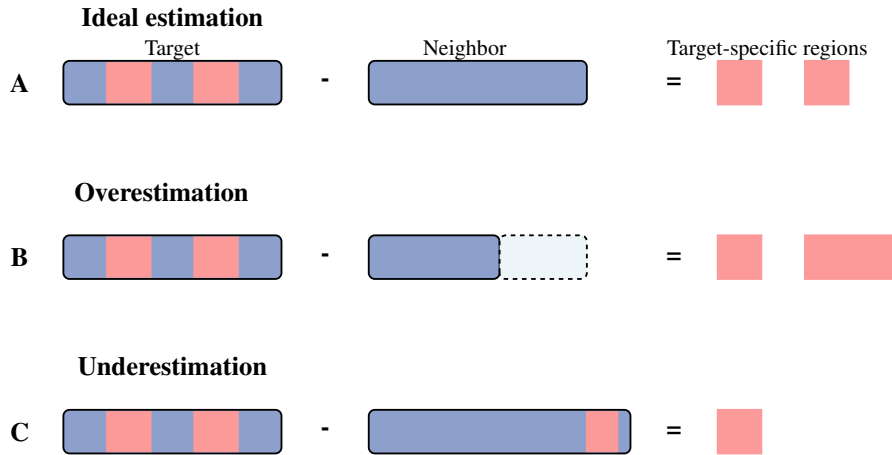


Figure 4.3: Compared to an ideal estimation (A), “atypical” neighbor genomes can lead to either an over (B) or underrepresentation (C) of unique regions.

The laboratory house mouse, widely used in biomedical research, descends mostly from the *M. m. domesticus* subspecies (94.3% of its genome). However, it also contains regions that are associated with both *M. m. musculus* (5.4%) and *M. m. castaneus* (0.3%). This is because the first laboratory strains arose by interbreeding individuals of distinct populations [9, 160, 161]. The result is a mosaic genome where different strains can have different genetic contributions from different subspecies. A study on 11 laboratory strains estimated that two-thirds of the genome with known ancestry was descendent from *M. m. domesticus*, and 11% of the genome was identical by descent across all strains [162].

Here I aim first at identifying genomic regions that are unique to each of the three house mouse subspecies. Then I identify genomic regions that are unique to *M. m. domesticus*, the subspecies from which all laboratory mouse strains have been derived.

4.1.3 Chapter aims

In this chapter, I identify and annotate lineage-specific regions in human and mouse. First, I explore the comparisons in the Homininae subfamily listed in Table 4.1 and represented in Figure 4.4.

Starting with the assembly-specific regions, I quantify the T2T-specific regions that are

Table 4.1: Comparisons for the Homininae subfamily

	Name	Description
1	Assembly-specific	between the T2T and the GRCh38 human genome
2	Haploid-specific	between phased haplotypes of a diploid assembly
3	Reference-specific	1-to-1 comparison between reference genomes
4	Human-specific	Multiple human genomes against the Chimpanzee reference
5	Homininae-specific	Comparison between all Homininae species
6	Chimpanzee-specific	1-to-1 comparison with the T2T assembly

absent from the GRCh38 reference genome (Figure 4.4A). Following that, I use *Fur* to search for haplotype-specific differences that can be found between a phased human diploid genome (Figure 4.4B). The search for human-specific regions will be done in several steps. First, I compare both the T2T and the GRCh38 reference genome, one at a time, to the common chimpanzee reference genome, obtaining a list of reference-specific differences. These differences are the upper limit of human-specific regions that can be detected (Figure 4.4C). Then, I explore how increasing the number of genomes in the comparison reduces the amount of lineage-specific material obtained. Finally, to quantify the effect of incomplete lineage sorting, the neighborhood is extended to comprise the entirety of the Homininae subfamily with the inclusion of gorilla (Figure 4.4D). As a complement, I also report on chimpanzee-specific regions on the background of the human T2T assembly (Figure 4.4C).

Then, in the second part, I search for species-specific regions in house mice. Here I start by searching for species-specific regions in single genomes of the three subspecies of *M. musculus* spp.. Then I include the laboratory strains with *M. m. domesticus* to search for regions that are shared among all members of the best studied mouse species, but are absent from its closest relatives.

In each comparison, the target-specific regions found will be annotated and subjected to an enrichment analysis of biological functions.

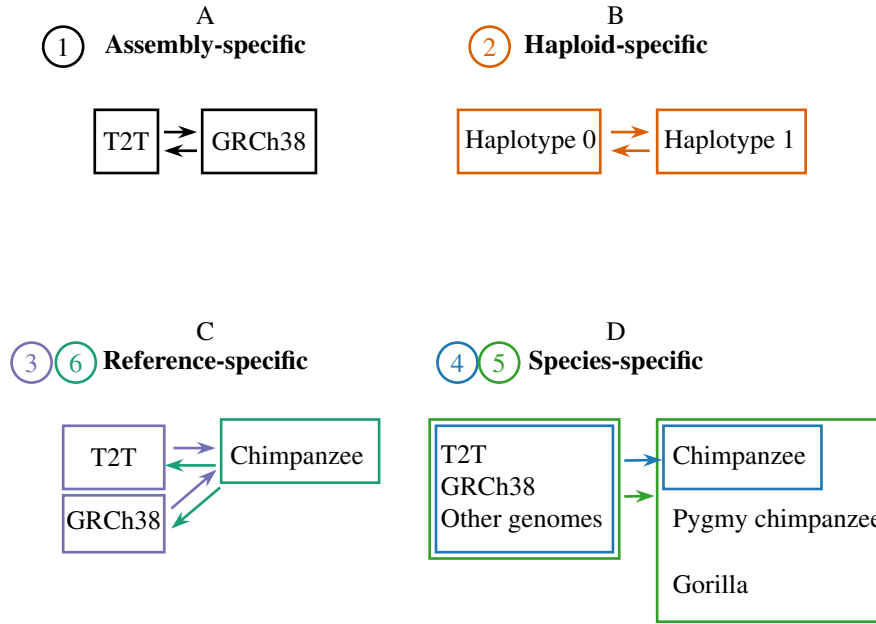


Figure 4.4: Schematic representation of the Homininae comparisons performed in this chapter. Numbers 1-6 refer to the comparisons listed in Table 4.1

4.2 Methods

The methodology for this chapter is grounded in Chapter 3. Using the *Fur* algorithm described in Section 3, as implemented in *Fur* version 4.2, I search for target-specific regions in two eukaryotic datasets. For the primate comparison, I considered the Homininae subfamily and not the Hominini tribe for two reasons. First, the NCBI taxonomic ranking system does not include the tribe category (Figure 4.1). Secondly, by restricting the comparison to only humans and chimpanzees, the Hominini, I would be ignoring the well-known fact of incomplete lineage sorting between the closely related triad of humans, chimpanzees and gorillas.

For the murine comparison, the target clade was set to *Mus musculus ssp.* and the remaining *Mus sp.* as the neighbors. In both the human and the mouse analysis, *taxi*, *ants* and *neighbors* from the package *Neighbors* were used to list all available genomes associated with the respective target clade and neighbor genomes.

The enrichment analysis of the detected regions was performed with *Gin* (v1.2) [99]. The mapping between genes and GO terms used the abridged gene2go files prepared in Chapter 2 [54], and available at <https://doi.org/10.17617/3.4IKQAG>.

Additionally, I also used from the *Biobox* package [163] the programs *nj*, *cres*, *getSeq* and *cutSeq*.

4.2.1 Data collection

The taxonomic ID of the target clade was obtained with *taxi* and *ants* from the *Neighbors* package. Using the program *neighbors* I extracted the complete list of available genomes associated with the given taxonomic ID. The corresponding metadata for each of the available genomes was obtained using the summary function of the NCBI command-line tool *datasets* (v15.12.0). Accessions marked as “suppressed”, or that were otherwise highly-fragmented (> 1000 scaffolds) or comprised of pseudohaplotypes were filtered out. The latest version of the remaining accessions were obtained with the download function of *datasets*. Only assemblies at the complete or chromosomal level were considered and atypical assemblies were excluded, unless stated otherwise.

4.2.1.1 The Homininae subfamily

The complete dataset for the *Homo sapiens* comparison contains 17 genomes: 11 human genomes, four chimpanzee genomes and two gorilla genomes. The corresponding accessions, genome sizes and the scaffold N50s are listed in Table 4.2.

4.2.1.2 The house mouse

In total, 22 genomes were obtained for the analysis of *Mus musculus*, 15 laboratory strains, and five wild species. Table 4.3 lists all the genome accessions, their corresponding species names, and laboratory strains if available. As mentioned in Section 4.1.2, the latest available genome for *M. m. castaneus*, is 3.0 Gb long and marked as “atypical” by the NCBI, while the *M. m. domesticus* genome is cautioned to be too small at 2.5 Gb.

4.2.2 Phylogenetic tree

The *HoxD* cluster was used to construct the Homininae phylogenetic tree. The start and end coordinates of the *HoxD* cluster were extracted from the annotation file of the T2T assembly. The region was extracted and mapped to each of the other assemblies using *blastn*. The subject coordinates of the best hitscore were then used to extract the *HoxD* cluster from each respective assembly. For the mouse genomes, the phylogenetic tree was constructed based on the smallest chromosome, chromosome 19.

For both trees *phylonium* [130] was used to calculate the distance matrix, and the neighbor-joining algorithm, as implemented in *nj* (*Biobox*), was applied to construct the trees. These trees rooted using the minimal ancestor deviation [91]. For visualization of phylogenies *ggtree* [92] was used.

Table 4.2: List of the Hominiinae genome accessions obtained. Ordered by accession and species name.

Species	Common name	Accession	Size (Gb)	Scaffold N50 (Mb)
<i>Gorilla gorilla gorilla</i>	Western lowland gorilla	GCA_02885495.2	3.6	151.6
		GCA_029281585.2 ^a	3.5	150.8
<i>Homo sapiens</i>	Human	GCA_000001405.29 ^a	3.1	67.8
		GCA_003634875.1 ^b	2.9	152.5
		GCA_009914755.4 ^c	3.1	150.6
		GCA_011064465.2	3.1	150.2
		GCA_014905855.1	3.1	142.0
		GCA_016695395.2	2.9	150.5
		GCA_018852605.2	2.9	146.8
		GCA_018852615.2	3.0	154.3
		GCA_018873775.2	3.1	149.7
		GCA_021951015.1	3.0	154.4
<i>Pan paniscus</i>	Pygmy chimpanzee	GCA_024586135.1	3.1	148.0
		GCA_028858825.2	3.1	147.5
		GCA_028858845.2	3.2	147.0
<i>Pan troglodytes</i>	Chimpanzee	GCA_029289425.2 ^a	3.2	147.0
		GCA_028858775.2 ^a	3.2	146.3

^a Reference genome.

^b *H. sapiens* phased diploid genome.

^c *H. sapiens* Telomere to telomere (T2T) assembly.

Table 4.3: List of the *Mus sp* genomes accessions obtained. Ordered by accession and species name.

Species	Common name	Strain	Accession	Size (Gb)	Scaffold N50 (Mb)
<i>M. caroli</i>	Ryukyu mouse	-	GCA_900094665.2	2.5	122.6
<i>M. musculus</i>	House mouse	C57BL/6J	GCA_000001635.9 ^a	2.7	106.1
		B10.RIII	GCA_030265425.1	2.7	144.4
		BALB/c Nude	GCA_031761455.1	2.5	101.0
		NOD/SCID	GCA_031763685.1	2.6	121.8
		C3H_HeJ	GCA_921997125.2	2.5	127.2
		BALB_cJ	GCA_921997145.2	2.5	127.2
		DBA_2J	GCA_921998315.2	2.6	130.9
		NOD_ShiLtJ	GCA_921998325.2	2.5	127.3
		A_J	GCA_921998355.2	2.5	127.2
		129S1_SvImJ	GCA_921998555.2	2.5	126.8
		CBA_J	GCA_921998905.2	2.5	127.0
		C57BL_6NJ	GCA_921999865.2	2.5	126.9
		NZO_HILtJ	GCA_947593165.1	2.7	137.9
		LP_J	GCA_947599735.1	2.6	134.5
		-	GCA_949316315.1	2.7	132.2
<i>M. musculus castaneus</i>	Southeastern mouse	Asian house mouse	CAST_EiJ	3.0 ^b	139.0
<i>M. musculus domesticus</i>	Western mouse	European house mouse	WSB_EiJ	2.5 ^b	126.9
<i>M. musculus molossinus</i>	Japanese wild mouse		JF1_MsJ	2.6	133.1
<i>M. musculus musculus</i>	Eastern mouse	European house mouse	PWK_PhJ	2.5	127.0
<i>M. spretus</i>	Western wild mouse		GCA_001624865.1	2.6	132.0
			GCA_921997135.2	2.5	129.1

^a *Mus musculus* reference genome, GRCm39.

^b Genome with atypical length

4.2.3 Human-specific regions

For each of the six comparisons listed in Table 4.1, I search for target-specific insertions that are at least 500 bp long and quantify their distribution across the genome. Then, I annotate all promoter regions, located in a 2 Kb region upstream of the transcription start site. Finally, I perform an enrichment analysis with the package *Gin* to identify biological functions enriched in the identified regions.

The enrichment analysis with *Gin* is based on a Monte Carlo approach, where I test the null hypothesis of no enrichment for each GO category occupied by at least 10 genes. For this, the lineage-specific regions were shuffled 10^6 times by splitting the iterations into 20

For a more detailed description of the enriched analysis, please see Chapter 2, Section 2.2.5.

4.2.3.1 Assembly-specific regions

To search for assembly-specific regions (Figure 4.4A), I compared the T2T assembly against the GRCh38 human reference genome. From this comparison, I obtained a list of newly assembled regions in the T2T assembly that were absent from the GRCh38 human reference genome. Assembly-specific regions in the T2T assembly were annotated with the T2T annotation file.

4.2.3.2 Haplotype-specific regions

The genome with the accession GCA_003634875.1 is a phased diploid assembly. I split it into two haploid genomes for each phased haplotype and used *Fur* to search for genomic regions that are present in the phase 0 haplotype but absent from phase 1 (Figure 4.4).

Because this diploid accession has no corresponding annotation file, I mapped the obtained regions to the T2T reference genome using *blastn*. The subject coordinates of the best hit score for each region were extracted and annotated with the T2T annotation file.

4.2.3.3 Reference-specific regions

In the reference-specific comparison (Figure 4.4C), I perform a pairwise comparison between a human and a chimpanzee genome. I start by comparing the T2T and the GRCh38 genomes, one at a time, against the common chimpanzee reference genome, and then do the reverse comparison.

From the T2T and the chimpanzee comparison, I obtain the superset of species-specific regions that can be obtained for either the humans or the chimpanzee.

Using the same parameters, I then compare both the GRCh38 and the T2T reference genome against the chimpanzee reference genome. For the T2T comparison, I suggest a new version of *Fur* that can handle highly repetitive regions by applying masking in the second subtraction step.

4.2.3.4 Hominini-specific regions

In the search for human-specific regions, I start by expanding the target clade to include all 11 human genome accessions against the common chimpanzee reference genome. The T2T assembly was set as the target representative. From this comparison, I obtain a list of genomic regions that are shared across all human genomes and absent from the chimpanzee reference genome.

Since there is only one *P. troglodytes* high-quality genome available, the reverse comparison was not performed.

4.2.3.5 Homininae-specific regions

Finally, I search for human-specific regions against the entire Homininae subfamily. Into this comparison I also include the three pygmy chimpanzee genomes and the two gorilla accessions. The T2T reference genome was set as the reference target. The end result is a list of all human-specific regions that are shared across the human genomes and excluded from both the chimpanzees and the gorillas. Thus, I also removed regions that might have remained due to incomplete lineage sorting between chimpanzee and gorilla.

4.2.4 House mouse-specific regions

For the house mouse dataset, I perform two types of comparisons. First I search for species-specific regions for each of the three house mouse subspecies: *M. m. castaneus*, *M. m. domesticus* and *M. m. musculus*. For this I perform the three comparison as listed in Table 4.4.

Then, I search for laboratory-strain specific regions that are shared across the laboratory strains and *M. m. domesticus*, and are absent from the other wild species. The GRCm39 reference genome was set as the target representative.

Table 4.4: Comparisons for the *M. musculus* subspecies.

	Target	Neighbors
1	DOM	CAS and MUS
2	CAS	DOM and MUS
3	MUS	CAS and DOM

4.2.5 Hardware/Software

Timed runs of *makeFurDb* and *fur* were performed on a server running Ubuntu 22.04.4. The server contains 16 Intel CPU cores (Xeon 2.10GHz) with 64 threads and 264 GB of RAM. The runtime and memory required for running each process was measured with the utility program *time*. This was called using */usr/bin/time* to distinguish it from the *bash* command *time*. The construction of the Fur database with *makeFurDb* was performed with 5 threads. For most of the comparisons *fur* itself was run with 25 threads for both the intersection and the second subtraction step with *blastn*.

4.2.6 Data availability

The complete list of detected regions and their respective coordinates, and annotated promoters, is included in the Supplementary Material SM3.

4.3 Results

Whole genome comparisons uncover differences between assemblies. These can be used to describe changes between assembly releases or simply to better understand what is new. Here I compare the telomere-to-telomere (T2T) human assembly and the GRCh38 human reference genome. The T2T assembly is the first ungapped, telomere-to-telomere human genome available, and it is therefore relevant to ask how and where it differs from the current human reference genome, the GRCh38. The UCSC browser [164] contains a track describing 615 new regions, 251.3 Mb, on the T2T assembly that are not present in the GRCh38 reference [41]. Albeit called unique to the T2T assembly, these regions in fact represent non-colinear blocks of at least 1 Mb long between the two assemblies.

In this chapter, however, I define unique regions as those present in the query and absent from the subject. Using the T2T and GRCh38 comparison as an example, *Fur* detects 23.6 Mb of T2T unique regions after the first subtraction step and 5.3 Mb of unique regions after the second subtraction step with *blastn*. This is but 2.1% of the regions described in

the UCSC browser. To understand this difference, let's consider an 8.1 Mb non-colinear block located at chr1:132,678,189-140,862,680 in the human T2T assembly. Despite being described as unique, this block has several matches in the GRCh38 reference genome both in the main chromosomal assemblies and in several scaffolds. Thus, although the T2T region is labeled as being unique on the UCSC browser, it is still present in the GRCh38 reference genome, and is thus not considered unique in this chapter.

Figure 4.5 shows the differences in the distribution of the UCSC regions compared to the regions obtained with *Fur*. The UCSC regions are distributed all across the genome, occupying between 1.8% of chromosome 2 and 60% of the Y chromosome. The largest cluster of newly expanded repeats is 29.7 Mb long (chrY:32,362,861-62,057,190), followed by 28.4 Mb on chr9:48,424,795-76,854,863 and 15.7 Mb at chr15:1-15,676,637. The regions detected in the *Fur* comparison are much sparser ranging from 0.02% in chromosome 16 to 1.9% on chromosome 9 (Figure 4.5). More than half of the unique regions, 57.1%, are located in chromosome 9. Of the 6,762 intervals detected by *fur*, 5,955 (88.1%) intersect the unique regions described by the UCSC tracker. All but three of the *Fur* regions are contained in their entirety (>90% of the query length) inside the non-colinear blocks. The remaining three regions are chr14:8,875,733-8,876,356 (87.2% overlap), chr20:32,497,863-32,498,376 (overlaps two regions for a total of 80.0%) and chr1:247,833,739-247,853,588 (53.4% of overlap).

4.3.1 Haplotype-specific regions

The diploid assembly GCA_003634875.1 consists of two haplotypes called phase 0 and phase 1, which are both 2.9 Gb long. I searched for haplotype-specific regions in phase 0 when compared to phase 1. From the first subtraction step of *fur* I 893 regions comprising 1.3 Mb of regions, that were reduced to 496 regions totalling 0.9 Mb after the second subtraction step with *blastn*. The 496 regions are distributed across all chromosomes, including on the X chromosome, with seven regions also located in unplaced scaffolds. The longest region is located at chr4:9,367,155-9,385,785 and is 18.6 Kb long, while the second and third largest regions are located at chr22:28,014,142-28,030,919 and chr22:28,052,053-28,064,609 and are 16.8 and 12.6 Kb long, respectively.

Since there is no annotation file for this haplotype assembly, I mapped the haplotype-specific regions to the T2T assembly using *blastn*, and took the annotation associated with the best bit-score for each interval. From this, I mapped 417 regions of the original 496, of which ten distinct intersected ten promoter regions: four uncharacterized lncRNA (long non-coding RNA), four immunoglobulin λ genes, COL6A5 (collagen type VI al-

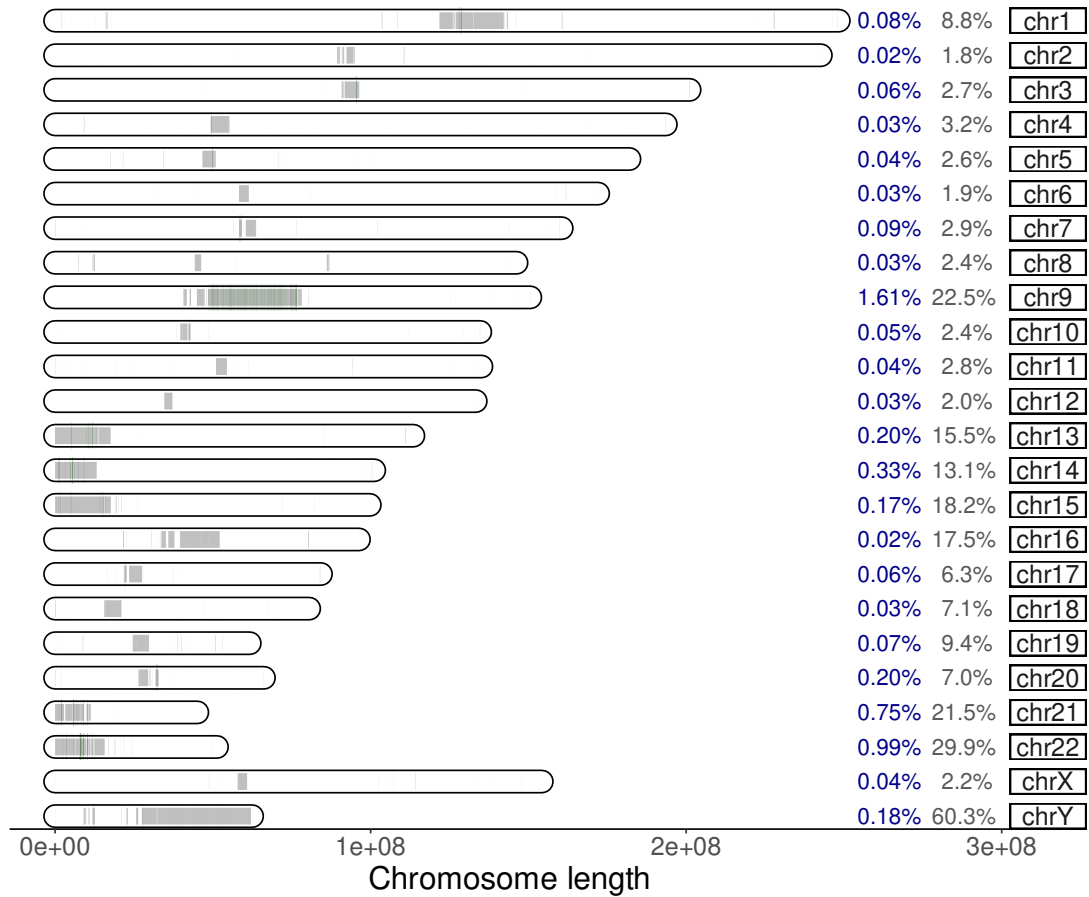


Figure 4.5: Distribution of the T2T unique regions described by the UCSC browser in grey, and the regions detected with *Fur* when comparing the T2T to the reference assembly GRCh38. In red are the regions that do not overlap with UCSC regions, in green regions that overlap with the UCSC regions. On the right the proportion of the chromosome occupied by the UCSC non-colinear blocks in grey and in blue the proportion occupied by the unique regions detected by *Fur*.

pha 5 chain) and GSDMC (gasdermin X). Chromosome 22 has the highest number of haplotype-specific regions, 0.08% of the chromosomes, followed by chromosomes 3 and 15 at 0.05 and 0.03%, respectively.

Additionally, I also annotated 208 genes whose transcripts intersect the haplotype-specific regions. A fifth of the annotated transcripts are long non-coding RNAs (lncRNAs) with an unknown function, which together with the intersected long intergenic non-protein coding RNAs (lincRNAs), means that one quarter of all annotated transcripts are RNA molecules (Table 4.5). As for protein-coding genes, the most common type of transcripts are associated with immunoglobulins from the λ locus (chr22q11.2) and metabolic genes as listed in Table 4.5.

Table 4.5: Description of annotated transcripts for the unique regions in phase 0 of the diploid comparison, with at least 5 entries.

Count	%	Description
42	20.2	Uncharacterized, lncRNA
14	6.7	Immunoglobulin λ locus
10	4.8	Long intergenic non-protein coding RNA, lincRNA
8	3.9	Ubiquitin specific peptidase
6	2.9	Solute carrier family
5	2.4	Scm like with four mbt domains
5	2.4	Collagen type VI/XXIV alpha

4.3.2 Human-specific regions

As mentioned in Section 4.1.1, there are currently two high-quality human assemblies that are prime candidates to serve as target representative in the first subtraction step of *fur*: the GRCh38 patch 14 of the human reference genome, released in May 2022 and the telomere-to-telomere (T2T) complete human assembly, released in April 2022.

I start by comparing both of them to the chimpanzee reference genome, and then perform the reverse comparison to search for chimpanzee-specific insertions in the chimpanzee reference genome.

4.3.2.1 GRCh38 and the common chimpanzee

Starting with the GRCh38 reference genome, after the first subtraction step, I obtained 39.1 Mb (1.2% of the genome) of GRCh38-specific regions. After the second subtraction step, this got reduced to 16.0 Mb, 0.5% of the genome, distributed across 8,760 regions. The first subtraction step of *fur* took 28.2 seconds, while the full *fur* run required 3 h 41 min. The runtime time difference is almost entirely due to the second subtraction step which is based on a call to *blastn*.

The Y chromosome, chromosome 21 and chromosome 9 contain the highest proportion of GRCh38-specific regions occupying 1.1% (0.6 Mb), 0.9% (0.4 Mb) and 0.6% (0.9 Mb) of their length, respectively (Figure 4.6). In contrast, chromosome 3 has the lowest proportion of lineage-specific regions at 0.2% (0.5 Mb). The distribution of these regions, on the GRCh38 assembly, shown in Figure 4.6, indicates that there are large gaps where no unique regions were found. These regions tend to match the non-colinear blocks shown in Figure 4.5. For example, chromosomes 1, 9 and the Y chromosome contain a large gap in the middle of the chromosome, which matches the centromeric satellite regions that

are hard to assemble and possibly collapsed. Additionally, for chromosomes 13 to 15 and 21 to 22, which are acrocentric, no GRCh38-specific regions could be found in the 5' chromosome cap. This is because the assembly of the 5' cap in the GRCh38 assembly is problematic.

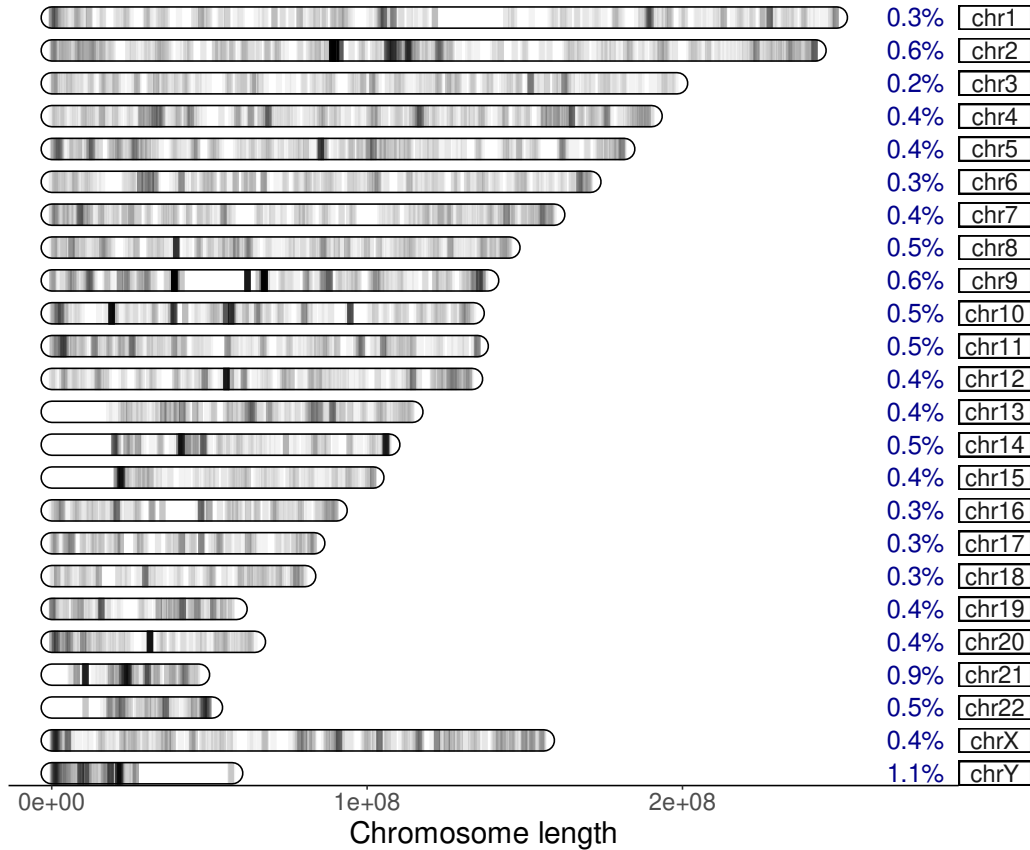


Figure 4.6: Distribution of the GRCh38-specific regions detected in the GRCh38-Chimpanzee comparison. GRCh38-specific regions are shown in grey. In blue, the proportion of the chromosome that is GRCh38-specific.

Figure 4.6 only shows the insertions located in the main chromosomal assemblies. There are, however, also 1253 GRCh38-specific regions totaling 2.7 Mb that are located in secondary scaffolds. Scaffold KZ208923.1, for example, which closes a gap on the Y chromosome, contains a 28.3 Kb GRCh38-specific regions which occupies 58.4% of the scaffold. A second 3.8 Kb region in a scaffold fix of chromosome 13 occupies 52.0% of its 7.3 Kb. The GRCh38 reference assembly contains 680 scaffolds and patches that have not been merged into the main chromosome assembly. Of these, 236 (34.7%) contain GRCh38-specific regions. In 10 of these scaffolds, GRCh38-specific regions occupy at least one quarter of it. However, the majority of the unplaced scaffolds have less than 1% of their length covered by GRCh38-specific regions.

The lineage-specific regions in the GRCh38-Chimpanzee comparison intersected 451 promoters. Interestingly, 90 of these promoters, that is one-fifth, are located in the non-merged scaffolds. Table 4.6 lists the function of the most common annotated promoters. More than a fourth, 27.7% of the intersected promoters are classified as “uncharacterized” lncRNAs, with only three “uncharacterized” protein-coding promoters identified. Together with long intergenic non-coding RNAs (lincRNAs), small nucleolar RNAs (snoRNAs) and microRNAs, approximately one third of the annotated promoters encode RNA molecules that are not mRNAs. The protein-coding genes, in turn, are associated with the immune system, intersecting the three immunoglobulin locus responsible for producing distinct polypeptide chains: the light κ chain, light λ chain and the heavy chain. Additionally, these regions also intersect five promoter associated with the major histocompatibility complex and 26 olfactory receptors.

Table 4.6: Description of the GRCh38-specific annotated promoters with at least 5 entries.

Count	%	Description
125	27.7	Uncharacterized
56	12.4	Immunoglobulin heavy/ κ / λ locus
26	5.8	Olfactory receptor
25	5.5	Long intergenic non-protein coding RNA
15	3.3	microRNA
13	2.9	Zinc finger
6	1.3	Small nucleolar RNA
5	1.1	Major histocompatibility complex, Class I and II

The next step is to perform an enrichment analysis on the GRCh38-specific regions to identify biological functions that might be preferentially associated with these regions. In this case, no enriched biological functions were found.

4.3.2.2 T2T and the common chimpanzee

Next, I compare the T2T assembly and the chimpanzee reference genome. After the first subtraction step of *fur*, I obtain 83.7 Mb (2.7% of the human genome) of human-specific regions. After the second subtraction step, with *blastn*, this value gets reduced to 15.8 Mb or 0.5% of the human genome. The first subtraction step of *fur* required just 30 seconds, while the full *fur* required 4 h 52 min, again almost all of it spent in the second subtraction step. The 10,279 detected regions totalling 15.8 Mb vary in length between 500 bp and 53,134 bp, with an average of 1,533 bp. The Y chromosome contains the highest proportion of T2T-specific regions, occupying 1.9% of its total length (1.2 Mb,

Figure 4.7). This is followed by chromosome 21 with 1.4% (0.6 Mb) and chromosome 22 with 1.2% (0.6 Mb). In contrast, chromosome 3 has the lowest proportion of lineage-specific regions with 0.3% (0.6 Mb).

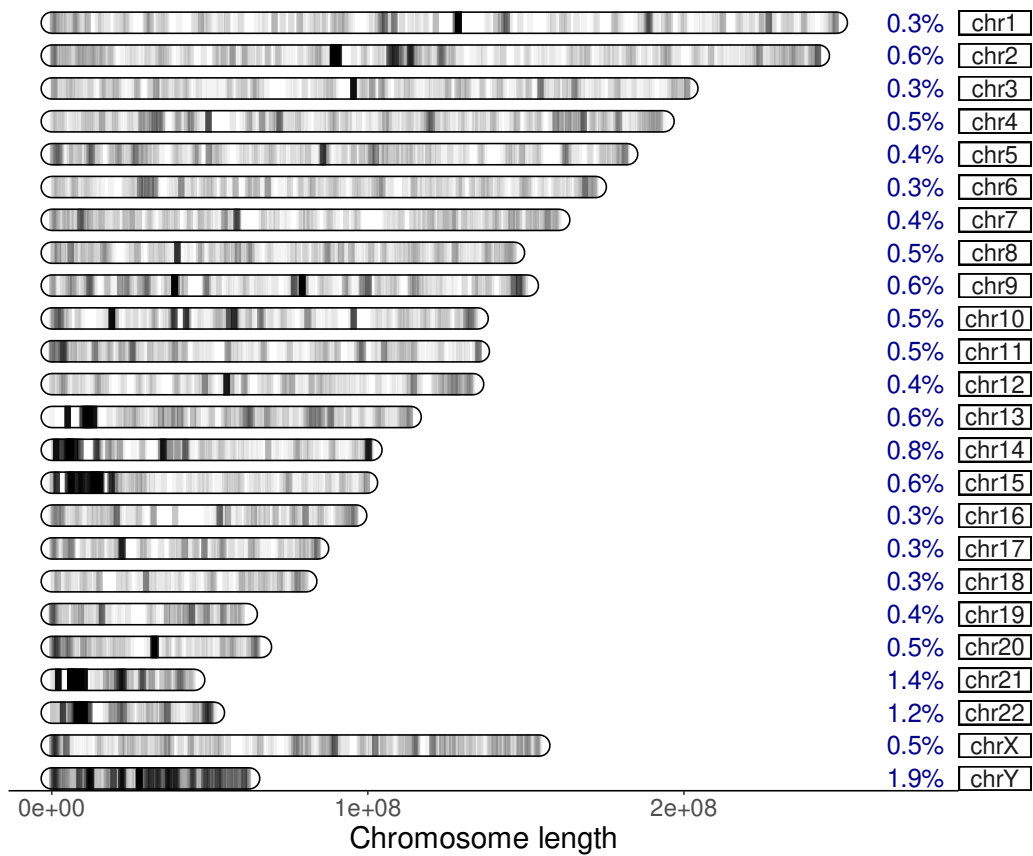


Figure 4.7: Distribution of the lineage-specific regions detected in the T2T-Chimpanzee comparison. Regions are shown in grey. In blue, the proportion of the chromosome that is T2T-specific.

Figure 4.7 shows the distribution of the human-specific regions based on the T2T assembly. These regions are distributed across all chromosomes, including the caps which previously were empty (Figure 4.6). Another major difference to the GRCh38 analysis, is that the Y chromosome now has human-specific regions distributed all across its length, while in GRCh38 these regions were restricted to the first half of the chromosome. It can also be seen that some regions have a higher number of human-specific regions. For example, Chromosome 14 contains 45 regions in an 85.4 Kb interval (chr14:1,345,801-1,431,163). On average, these regions are separated by 927 bp from each other. However, some are separated by as little as 9 bp. Because they are so close to each other, these regions merge into a solid block at the 5' of chromosome 14 in Figure 4.7. If all intervals separated by less than 100 bp are merged, the heatmap color becomes lighter (not shown).

The longest lineage-specific region, with 53.1 Kb, is located at chr8:46,552,139-46,605,272. It corresponds to a centromeric satellite (gsat_8_7 (GSAT)). The second largest region, 23.7 Kb, is located at chr2:90,620,593-90,644,312 in the immunoglobulin κ locus (IGK) region, intersecting one of two exons for IGKV6D-41 and IGKV1D-17. The third longest region, 23.0 Kb, is at chr14:35,255,922-35,278,964 and intersects an intron of a long intergenic non-coding RNA.

I next explore the functions associated with the human-specific regions shown in Figure 4.7. The 10,279 regions intersect 359 promoters, Table 4.7 lists the functions of the most common genes. Slightly more than a quarter of them, 28.1%, are “uncharacterized”, with 98 being lncRNAs and only three being protein coding. Together with the lncRNAs (7.5%), small nucleolar RNAs (4.5%), the microRNAs (3.6%), and the family with sequence similarity lncRNAs (2.2%), almost half (46.0%) of the annotated promoters encode RNA genes.

As for the protein-coding genes, the most common promoters are for the immunoglobulin κ and heavy locus, with 34 and 24 intersected promoters respectively. For the immunoglobulin λ locus only two promoters were present. Additionally, these regions also intersect olfactory receptors and keratin associated proteins. Like in the previous comparison with the GRCh38, no functional enrichment was found to be associated with these regions.

Table 4.7: Description of annotated promoters for the T2T-Chimpanzee comparison, with at least 5 entries.

Count	%	Description
101	28.1	Uncharacterized
60	16.7	Immunoglobulin heavy/ κ / λ locus
27	7.5	Long intergenic non-protein coding RNA
26	7.2	Olfactory
16	4.5	Small nucleolar RNA
13	3.6	microRNA
8	2.2	Family with sequence similarity ^a
6	1.7	Keratin associated protein

^a 7 lncRNAs and 1 protein coding gene.

4.3.2.3 The Homininae subfamily

After comparing both the T2T and the GRCh38 individually against the chimpanzee reference genome, I expanded the analysis to include the other 15 great ape genomes listed

in Table 4.2. I start by calculating the phylogenetic distances between the genomes that are being compared. Figure 4.8 shows the phylogenetic tree of the Homininae subfamily based on the *HoxD* cluster located on chromosome 2. The gorillas are the most divergent species, while chimpanzees and humans are slightly more closely related. On average, the entire dataset has a divergence rate of only 0.01 substitutions/site and the divergence is even lower within groups. For example, the human genomes have a genetic distance of 9.2×10^{-4} , while the two gorilla accessions are almost identical. The entire dataset contains four reference genomes, the starred taxa in Figure 4.8.

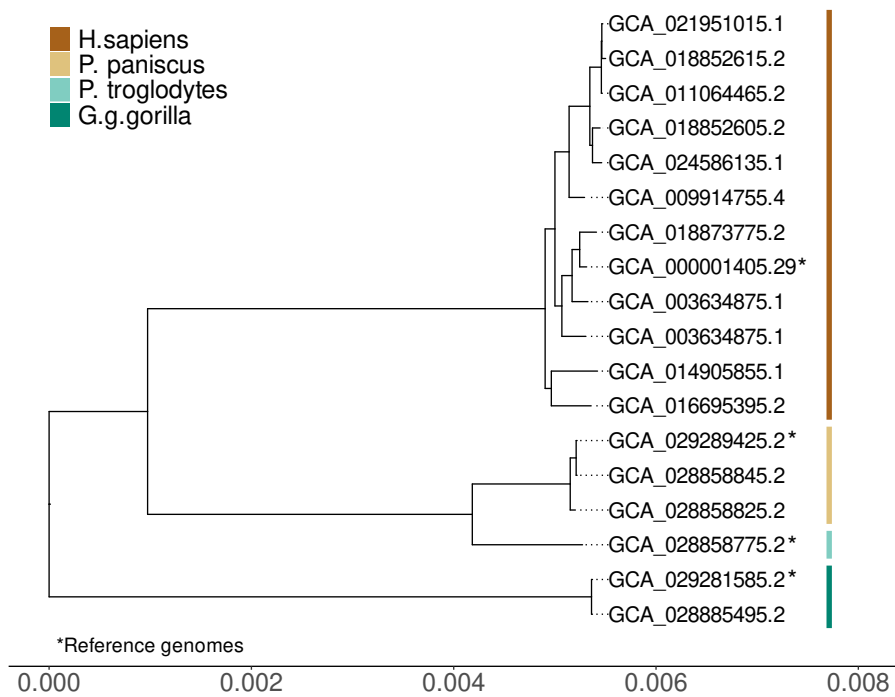


Figure 4.8: Phylogenetic tree based on the *HoxD* cluster of the Homininae genomes. On the x-axis is the phylogenetic distance in substitutions/site. The tree is rooted with minimal ancestor deviation (MAD).

When the T2T or the GRCh38 assembly was compared directly against the chimpanzee reference genome, I obtained more than 15 Mb of human-specific regions (Table 4.8). But when including all 11 human genomes in the analysis, the number of human-specific regions found almost halves to 8.1 Mb. Expanding the analysis to include the three pigmy chimpanzees further reduces the amount of human-specific material to 6.1 Mb (Table 4.8). The conclusion is that approximately 6.1 Mb of ubiquitous human sequences are absent from both chimpanzee species. Since it is known that approximately one-quarter of the human genome is closer to the gorilla lineage than the chimpanzees due to incomplete lineage-sorting [165, 166], I further extend this comparison by searching for

human-specific regions that are absent in both chimpanzees and gorillas. In this final comparison, the amount of human-specific regions is drastically reduced from 6.1 Mb to 0.8 Mb, comprising only 0.03% of the human genome (Table 4.8).

Table 4.8: Total length of regions (Mb) detected at each step of *fur* for the different target sets. The Homininae comparison (c) includes all assemblies described in Table 4.2. Except for the GRCh38 comparison, all others use the T2T assembly as the target representative.

	T2T	GRCh38	All humans ^a	All humans ^b	All humans ^c
Subtraction 1	83.7	39.1	83.7	67.6	44.0
Intersection	-	-	13.5	10.1	1.6
Subtraction 2	15.8	16.0	8.1	6.1	0.8

^a Against the common chimpanzee reference genome.

^b Against all chimpanzee genomes.

^c Against chimpanzee and gorilla genomes.

The Homininae comparison uncovered 600 human-specific regions (Figure 4.9). On average, the human-specific proportion of each chromosome that is considered to be human-specific is 0.03%, with chromosome 15 and the X chromosome being devoid of such regions. In this comparison, chromosome 21 has the highest proportion of human-specific regions, 0.2%, while the Y chromosome only has a single region spanning 600 bp.

Only 23 of these 600 regions intersect one of 20 distinct promoters (Supplementary Table S1). Once again, “uncharacterized” lncRNAs are the most common type of annotated promoters (Table 4.9). In total, nine of the 20 annotated promoters are RNA coding genes including lncRNAs, microRNAs and lincRNAs. Finally, I also detected two olfactory receptors and two keratin associated- proteins. No biological function was found to be enriched, most likely due to the low number of promoters found.

Table 4.9: Description of annotated promoters for the Homininae comparison, with at least 2 entries.

Count	%	Description
5	25	Uncharacterized
2	10	Olfactory receptor
2	10	microRNA
2	10	Long intergenic non-protein coding RNA
2	10	Keratin associated protein

I also annotated all transcripts that intersect the 600 human-specific regions instead of

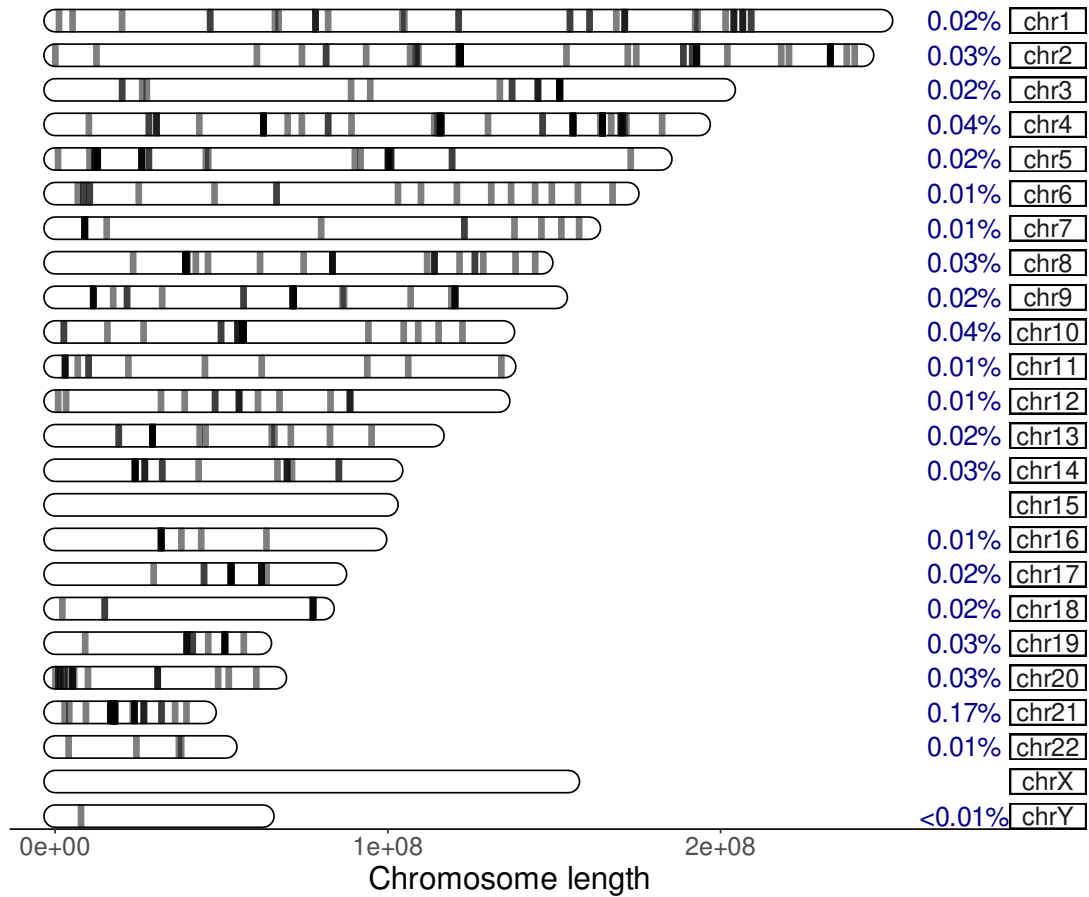


Figure 4.9: Distribution of the lineage-specific regions detected in the Homininae comparison. Regions are shown in grey. Color hue denotes the density of the regions. In blue, the proportion of the chromosome that is human-specific.

restricting the analysis to promoters. Still, as observed before, most of the intersected transcripts are associated with lncRNAs (30.1%) and lincRNA (10.7%). While the number of intersected olfactory receptors doubled from two to four (3.9%), the number of keratin-associated proteins remained the same (not shown).

4.3.3 Common chimpanzee-specific regions

Next, I move the focus from human-specific regions to chimpanzee-specific regions. I start by looking at the chimpanzee-specific regions obtained by comparing the common chimpanzee reference assembly against either the GRCh38 or the T2T human assembly. When compared to GRCh38, I obtain 13,930 regions that sum up to a total of 164.8 Mb or 5.2% of the chimpanzee genome. These regions are distributed across the genome with chromosome 23 having the highest proportion of chimpanzee-specific regions, 20.2% of its entirety. Figure 4.10 shows the distribution of the chimpanzee-specific regions found

in the comparison against the GRCh38 genome. Note that the common chimpanzee reference reinstates the old numbering system where chromosomes 2A and 2B are labelled 12 and 13.

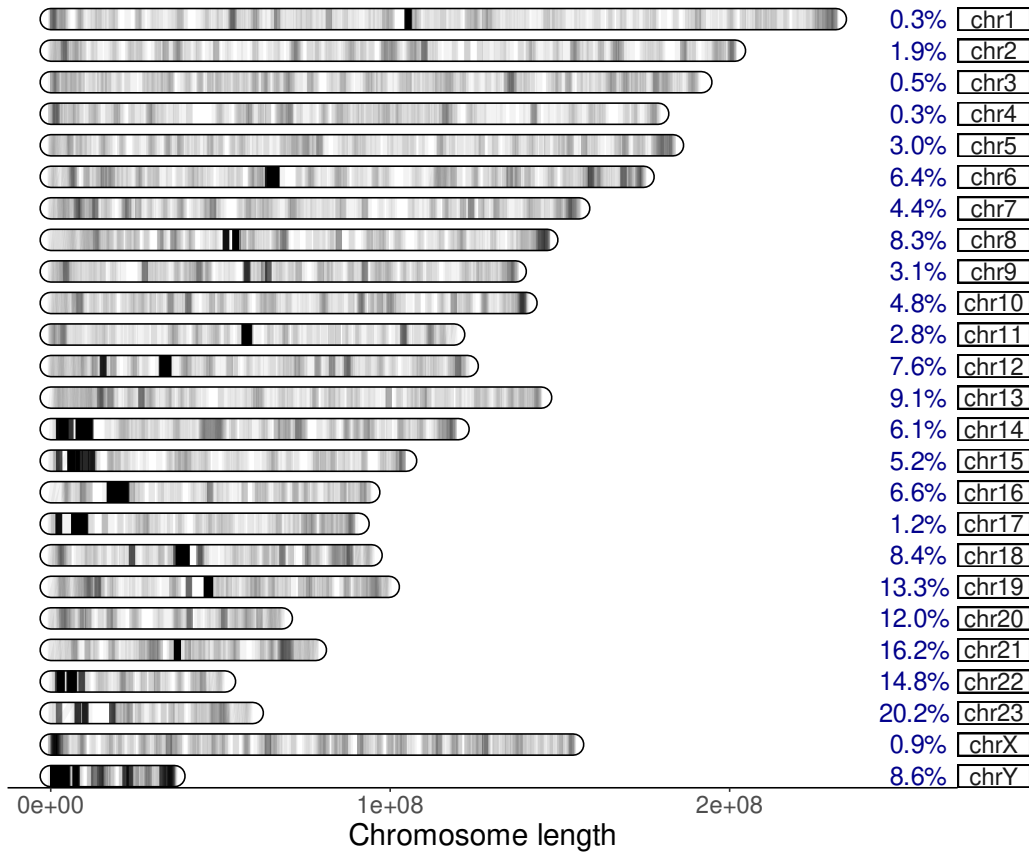


Figure 4.10: Distribution of the lineage-specific regions detected in the common chimpanzee-GRCh38 comparison. Regions are shown in grey. In blue, the proportion of the chromosome that is Chimpanzee-specific.

A total of 323 promoters intersected the chimpanzee-specific regions. Similar to the results for humans, one-third of the annotated promoters are “uncharacterized”, with 93 of the uncharacterized promoters being lncRNAs and 16 protein-coding promoters. The second most common type of annotated promoters are spliceosomal RNAs and olfactory receptors, with 11 annotated promoters each (Table 4.10). None of the functions associated with these regions were enriched.

4.3.3.1 Lineage-specific regions in telomere to telomere assemblies

I then search for chimpanzee-specific regions by comparing against the T2T assembly. Unlike in the previous comparisons, however, the second subtraction step of *fur* with *blastn* initially failed to complete even with 2 TB of RAM available and more than 90

Table 4.10: Description of annotated promoters for the Chimpanzee-GRCh38 comparison, with at least 5 entries.

Count	%	Description
109	33.8	Uncharacterized
12	3.7	U1/U6 Spliceosomal RNA
11	3.4	Olfactory receptor
9	2.8	Small nucleolar RNA
6	1.9	MAM and LDL-receptor class A domain-containing protein 2-like/1-like
5	1.6	Zinc protein
5	1.6	Putative (uncharacterized) protein
5	1.6	Ankyrin repeat domain

h of computation. The recommendation from the NCBI when dealing with repetitive regions is to mask them in either the query of the subject sequence. This might seem inconsistent under the assumption that species-specific regions are “unique”. However, these regions are only unique with respect to the neighbors, not with respect to the target.

With the advent of telomere-to-telomere assemblies, previous collapsed repetitive regions are now correctly assembled. For *fur* this means that if satellite regions in the targets differ from those in the neighbors, these regions are not removed in the first subtraction step and passed on to the second subtraction step with *blastn*. In the chimpanzee-T2T assembly comparison two problematic +3 Mb regions were identified in the first subtraction step, that overwhelmed the *blastn* run of the second subtraction step.

Since *blastn* is an integral part of *fur* and cannot be easily replaced, it is necessary to modify *Fur*. Previously, during the construction of the *Fur* database with *makeFurDb*, any masking information was stripped from both the target representative and the neighbor sequences. To allow masking in the second subtraction step, I redesigned its BLAST workflow and its integration in *makeFurDb* and *fur*.

Starting with *makeFurDb*, if the neighbor sequences contain masking information, this information is now extracted into a mask file with *convert2blastmask*, and incorporated into the BLAST database by running *makeblastdb* with the “mask_data” argument. Additionally, since BLAST databases have a size limit, each neighbor sequence now has its own BLAST database, which is then linked into a single database using the program *blastdb_aliastool*. If no masking information is available in the neighbor sequences, then a regular BLAST database is constructed.

The program *fur* now has an extra argument, -M, for masking during the second sub-

traction. This switches on the “db_soft_mask” argument of *blastn*. Additionally, since regions need only be present once in the neighbor sequence to be excluded from the final results, the number of high scoring pairs (max_hsp) per subject is set to 1, reducing the time and memory requirements of *blastn*.

Using the redesigned *makeFurDb* and *fur* I obtained 18,527 chimpanzee-specific regions, against the T2T assembly, totalling 165,2 Mb. These regions occupy 5.2% of the chimpanzee genome which is similar to the 5.2% of regions detected against the GRCh38 reference genome. Previously with the GRCh38 comparison, each chromosome contained on average 6.6% of chimpanzee-specific regions (Figure 4.10). With the T2T assembly comparison, the average percentage of each chromosome occupied by Chimpanzee-specific regions decreased to 6.5%. Figure 4.11 shows the distribution of the detected regions in the Chimpanzee-T2T comparison where chromosome 21 has the highest proportion of species-specific regions, 16.4%, while chromosome 1 has the lowest proportion, 0.5%.

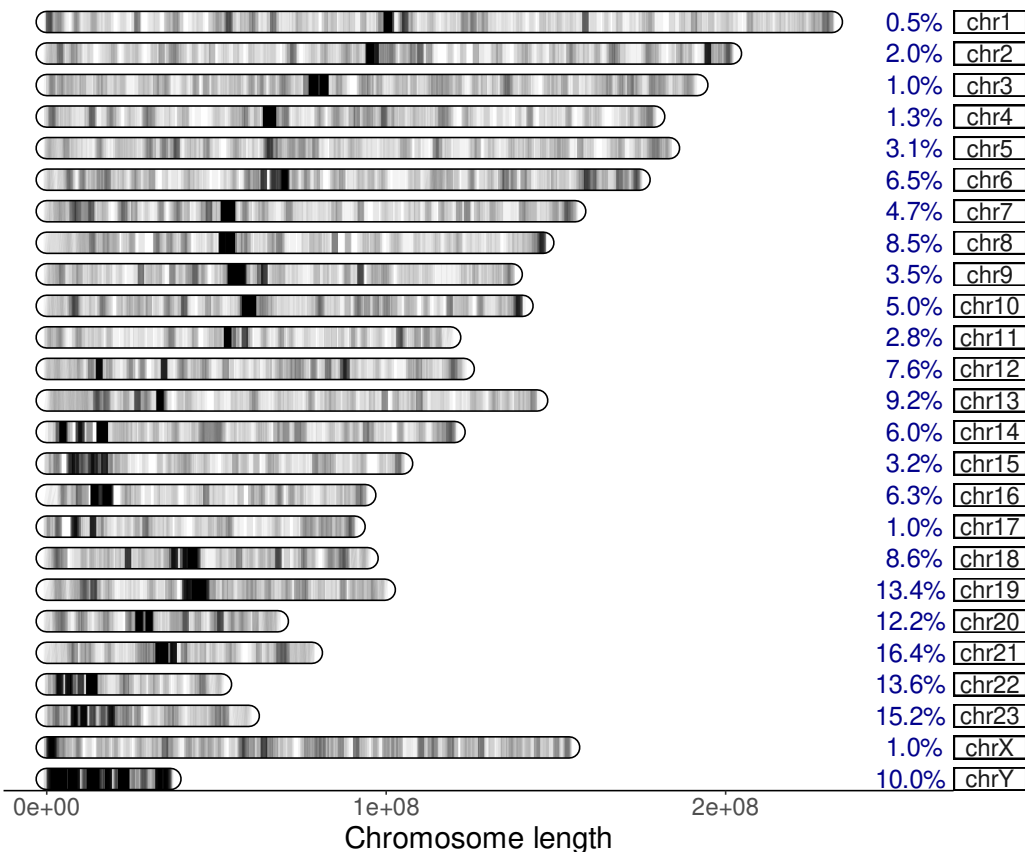


Figure 4.11: Distribution of the lineage-specific regions detected in the Chimpanzee-T2T comparison. Regions are shown in grey. In blue, the proportion of the chromosome that is Chimpanzee-specific.

In total 262 promoters were annotated. The most common type of promoters found were

once again “uncharacterized” promoters (Table 4.11). The majority, 85, of the “uncharacterized” promoters are lncRNAs while the remaining are protein coding. Besides the lncRNAs, there are also spliceosomal RNAs and small nucleolar RNAs. Among the protein-coding genes these regions intersect 13 olfactory receptors and 7 putative proteins (Table 4.11).

Table 4.11: Description of annotated promoters for the Chimpanzee-T2T comparison, with at least 5 entries.

Count	%	Description
100	38.2	Uncharacterized
13	5.0	Olfactory receptor
12	4.6	Small nucleolar RNA
10	3.8	U6 spliceosomal RNA
7	2.7	Putative (uncharacterized) protein

Still, no enriched biological function was found to be associated with these regions.

4.3.4 *Mus musculus* specific regions

I started by searching for species-specific regions between the three subspecies of *M. musculus* spp.. For this I compared one of the subspecies against the remaining two, for all possible combinations as listed in Table 4.4.

The total amount of subspecies-specific regions obtained for each comparison after the second *fur* subtraction varied between 2.0 Mb for *M. m. musculus* (MUS) and 5.4 for *M. m. castaneus* (CAS); (Table 4.12). *M. m. castaneus* which has an atypically large genome, has the highest amount of regions obtained both after the first and the second subtraction step, 18.6 Mb and 5.4 Mb, but *M. m. domesticus* (DOM) which has an atypically small genome does not have the smallest amount of subspecies-specific regions (Table 4.12). Hence excessive target material and not excessive neighbor material as illustrated in Figure 4.3B might have a stronger effect on the detection of unique regions.

Table 4.12: Total length of regions (Mb) detected after the two subtraction steps of *fur* for the three *M. musculus* spp..

	DOM	CAS	MUS
Subtraction I	8.3	18.6	5.5
Subtraction II	2.3	5.4	2.0

To annotate these regions I mapped them to the GRCm39 mouse reference genome using *blastn* and took the annotation associated with the best bit-score for each interval.

Starting with *M. m. domesticus*, a total of 1432 regions (79.5%) were mapped to the GRCm39 mouse reference genome for annotation purposes. The mapped DOM-specific regions intersect 35 promoters which cover 20 distinct functional categories (Supplementary Table S2). The only category with more than one member is “predicted gene” with 16 members. These 16 genes are made up of 13 are lncRNAs, one protein coding, one protein coding pseudogene and a snoRNA. For the *M. m. musculus* only 558 regions (34.1%) could be mapped to the GRCm39 reference genome. The MUS-specific regions intersect 17 promoters, eight of which are “predicted genes” (Supplementary Table S3). The *M. m. musculus* predicted genes are four lncRNAs, three snRNA and one snoRNA. There are also two protein-coding genes associated with olfactory receptors. Finally, for *M. m. castaneus* 2621 regions (70.9%) could be mapped to the GRCm39 reference genome. The CAS-specific regions intersect 19 promoters, eight of which are once again “predicted genes” (Supplementary Table S4). Four of the predicted genes are lncRNA, three are protein-coding and one is a snRNA. These regions also contain two “uncharacterized” lncRNA. In total, when taking into consideration the 71 annotated promoters across the three comparisons, half of them, 50.7%, are either predicted or uncharacterized genes or non-protein coding. Of the protein-coding genes, olfactory receptors are the most common, followed by cytochrome c, where two associated genes were identified (Table 4.13).

Table 4.13: Description of total annotated promoters for the three *M. m. spp.* comparisons, with at least 2 entries.

Count	%	Description
32	45.1	Predicted gene
4	5.6	Olfactory receptor
2	2.8	Uncharacterized
2	2.8	RIKEN cDNA
2	2.8	Cytochrome c

Figure 4.12 shows the phylogeny of the 22 genome accession used in the mouse comparison. The majority of the genomes belong to different laboratory mouse strains, with six neighbor genomes. In the phylogeny it can also be seen that *M. m. musculus* and *M. m. molossinus* are closely related, which partly supports the statement that *M. m. molossinus* is a hybrid between *M. m. musculus* and *M. m. castaneus*. The two most distant *Mus* species are *M. spretus* and *M. caroli*, which have diverged from the *M. musculus* approximately 2.7 and 4.5 million years ago, respectively [167].

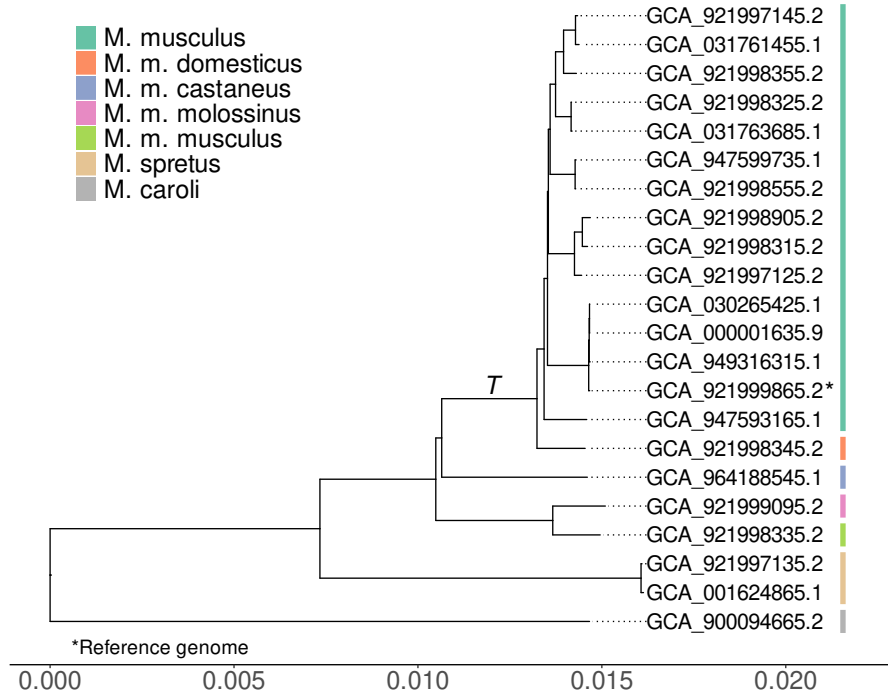


Figure 4.12: Phylogenetic tree of chromosome 19 for the mouse genomes. On the x-axis is the phylogenetic distance in substitutions/site. The tree is rooted with minimal ancestor deviation (MAD). Target clade is marked with *T*.

I then search for regions that are shared across all 16 genome strains and absent from its six neighbors. After the first subtraction step with *fur* I obtain 4,173 regions totalling 4.1 Mb. This is reduced to 93.8 Kb after intersecting the regions across all 16 target genomes, which finally results in just 16 regions totalling 14,865 bp. Figure 4.13 shows the location of these 16 detected regions in the house mouse genome, they are located on ten different chromosomes, with chromosome 10 having six and chromosome 16 two. The remaining eight regions are singletons on their host chromosome.

While these regions do not intersect any promoter, four regions intersect four gene transcripts. These are Akap7 (A kinase anchor protein), 4930447K03Rik (RIKEN cDNA), NCald (neurocalcin delta) and Tiam1 (T cell lymphoma invasion and metastasis 1).

4.3.5 Runtime and resources

The resources required for constructing a Fur index depend on the number of genomes that are being indexed, and also the number of threads. For most of the comparisons the database was constructed with five threads. Table 4.14 lists the runtime and memory required for constructing each Fur index. For a single target and neighbor this required

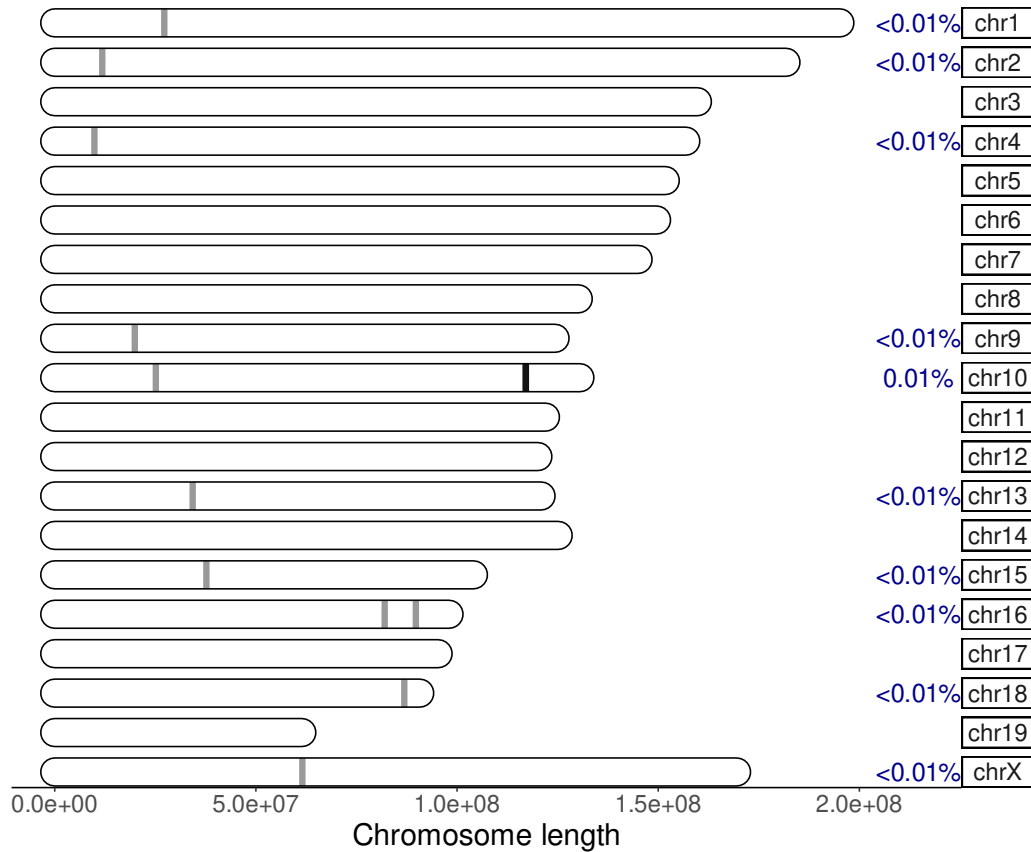


Figure 4.13: Distribution of the lineage-specific regions detected in the house mouse comparison. Regions are shown in grey. In blue, the proportion of the chromosome that is strain-specific.

between 2 h 53 mins and 3 h 45 mins, and between 59.0 and 72.8 GB of memory. As the number of sequences increases, so does the memory requirement. Indexing the four chimpanzees genomes, instead of only one, increased the elapsed time from 3 h 30 min to 4 h 18 min and the memory from 69.0 GB to 207.6 GB. To construct the Fur index for the entire Homininae dataset, including the Gorillas, I reduced the number of threads to four, explaining the reduction in memory at 153.5 Mb. This reduction in threads contributed to the increase in runtime to 8 h 12 min. It can also be seen that in the Chimpanzee-T2T comparison, despite including masking information in the redesigned *Fur* neither the runtime nor the memory escalated, as I had hoped.

Since the memory requirements for the construction of the Fur database with *makeFurDb* are proportional to the size of the longest sequence, the memory consumption listed in Table 4.14 might seem incongruent. However, this is due to the effect of the threading applied in *makeFurDb*. When using multiple threads, the neighbor sequences are distributed according to their name and not their length, thus *makeFurDb* is most likely indexing the

Table 4.14: Runtime and memory required for constructing the Fur database with *makeFurDb* using 5 threads, for each comparison.

Targets	Neighbors	User time (s)	Elapsed Time	Memory (GB)
Haplotype 0	Haplotype 1	10,075.2	2:53:38	59.0
GRCh38	T2T	11,432.7	3:23:44	70.6
GRCh38	Chimpanzee	12,533.7	3:38:36	69.2
T2T	GRCh38	13,097.6	3:45:30	63.1
T2T	Chimpanzee	11,904.2	3:32:58	69.0
Chimpanzee	GRCh38	11,794.1	3:23:20	70.7
Chimpanzee ^a	T2T	11,505.7	3:18:55	73.5
Humans	Chimpanzee	11,805.2	3:30:34	69.0
Humans	<i>Pan sp.</i>	55,795.2	4:18:36	207.6
Humans	<i>Pan sp.</i> +Gorilla ^b	79,712.0	8:12:10	153.5
DOM	CAS+MUS	20,499.0	3:07:00	95.2
CAS	MUS+DOM	22,025.2	3:16:40	97.8
MUS	CAS+DOM	20,461.2	3:07:30	89.5
Strains	<i>Mus sp.</i>	67,282.9	6:42:15	129.2

^a Masked information included in the BLAST database.

^b With 4 threads

longest sequences, usually chromosome 1, of every neighbor simultaneously. This explains why in the *Pan spp.*+Gorilla comparison it was necessary to reduce the number of threads.

In contrast, the analysis with *fur*, tends to use less memory (Table 4.15). The runtime required for *fur*, however, varies depending on the size of the regions that remain after the intersection step. The longer the intersected regions, the slower the second subtraction step with *blastn*. For example, the first two *fur* steps in the Chimpanzee and GRCh38 comparison required 30.2 seconds, while the second subtraction step required 14 h 27 mins.

It is also possible to see the drastic reduction in both runtime and memory requirements when the masking option of *fur* is applied. To compare the chimpanzee genome against the GRCh38 reference genome without masking, *fur* required 6 h 35 mins and 50.4 GB of memory. However in the Chimpanzee-T2T comparison, where repetitive regions were masked in the database and the maximum number of high scoring pairs was set to 1, the runtime was reduced to 23 mins and 45 seconds, and a total memory usage of 6.9 GB. This is a 95.7% reduction in runtime and a 86.3% reduction in memory consumption.

Table 4.15: Runtime and memory required for running *fur* with 25 threads. Memory used is in GB.

Targets	Neighbors	User time (s)	Elapsed Time	Memory
Haplotype 0	Haplotype 1	535.8	00:04:55	6.5
GRCh38	T2T	1,093.2	00:06:51	6.4
GRCh38	Chimpanzee	47,904.6	03:41:31	42.8
T2T	GRCh38	2,260.6	00:17:21	14.7
T2T	Chimpanzee	61,127.1	04:52:20	47.0
Chimpanzee	GRCh38	55,285.7	06:35:00	50.4
Chimpanzee ^a	T2T	2,386.6	00:23:45	6.9
Humans	Chimpanzee	51,093.8	03:05:51	70.2
Humans	<i>Pan sp.</i>	42,567.9	02:33:50	69.4
Humans	<i>Pan sp.</i> +Gorilla	18,337.7	00:54:03	67.9
DOM	CAS+MUS	22,090.0	02:20:59	14.3
CAS	MUS+DOM	10,211.2	00:46:02	12.9
MUS	CAS+DOM	3,460.0	00:27:43	10.5
Strains	<i>Mus sp.</i>	32,288.5	00:48:53	89.1

^a Masked algorithm applied during the 2nd subtraction step of *fur*.

4.4 Discussion

Comparative genomics started during the late 90s, with the first two published genomes from free-living organisms, *Haemophilus influenzae* Rd and *Mycoplasma genitalium* G37 [26, 168]. This comparison focused on comparing known genes between both organisms, to gain insight into physiological differences between two highly distinct prokaryotes.

In 2002, the first two mammalian whole genomes were compared, the mouse and the human genomes. Due to the large evolutionary distance between both species, the study focused on large-scale syntenic blocks and orthologous genes [9]. The release of the chimpanzee genome three years later meant that now closely related species could be compared [2]. Thus, the focus changed to detecting differences between the human and chimpanzee genomes that could explain phenotypic differences between both species.

In this chapter I explored the search for lineage-specific insertions using the program *Fur*, which finds regions present in all members of a sample of target genomes that are absent from a set of - preferably closely related - neighbor genomes. Although I ultimately aim to for lineage-specific regions, this method can be applied to any two sets of genomes, as was exemplified by my search for assembly-specific and haplotype-specific insertions.

I first started by comparing the T2T assembly to the GRCh38 human reference genome. The aim here was to identify regions in the T2T assembly that were absent in the GRCh38 reference. From this, I identified 5.3 Mb of T2T-specific regions that are truly absent from the GRCh38 assembly, and not merely collapsed or badly aligned (Figure 4.5). A large majority of these regions, 86.5%, intersect a human satellite 3 (HSat3), corresponding to 56.1% of all described HSat3 regions in the T2T assembly. HSat3 satellites are derived from a tandem repeat of “CATTC” and occupy 1.6% of the T2T genome, being the second most common type of satellite repeats [169]. These regions, because they are composed of large pericentromeric tandem repeats, are difficult to assemble and thus were missing from the human reference genome until the release of the T2T assembly [170]. This explains why most of the T2T-specific regions are pericentromeric repeats. The largest T2T-specific insertion that does not intersect a pericentromeric region is 19.9 Kb long and is located at chr1:247,833,739-247,853,588, intersecting a non-colinear block with the GRCh38 assembly. Excluding insertions that intersect non-colinear blocks leaves 11.2% of T2T-specific regions located outside hard-to-assemble regions. These T2T-specific regions intersect 317 genes, a fifth of which (20.5%) are RNA coding genes, mostly “uncharacterized”.

4.4.1 Haplotype-specific insertions

In the haplotype comparison, the number of haplotype-specific regions comprises of almost 1 Mb. The resolution could be increased by, for example, decreasing the minimum required length before a region is considered to be unique. These regions intersect “uncharacterized” lncRNAs, and perhaps more interestingly, the immunoglobulin λ locus on chromosome 22. The haploid-specific regions only intersect a small portion of the immunoglobulin λ locus, 3.6% of its length. Still, it is remarkable that structural variation at an immunoglobulin locus could be identified with this method when comparing two highly similar human haplotypes.

4.4.2 Human-specific insertions

In the first published human and chimpanzee comparison, it was estimated that the human genome contained between 40-45 Mb of human-specific insertions [2]. In our analysis, I detected between 15.8 and 16.0 Mb of human-specific insertions, in a 1-to-1 comparison to the common chimpanzee reference genome. This is a 60% reduction in the previously estimated human-specific regions. If I take into account the variability present in other human and chimpanzee genomes, then this value is reduced to 6.1 Mb, a reduction of

84.8%. Scally *et al.* [171] estimated that 0.6 Mb of non-repetitive regions in the human genome were absent from both chimpanzees and gorillas genome, but were present in *orangutangs* (80% of the regions) or other primates (14% the regions), with only 6% of the regions being truly human-specific. In the *Fur* results for the Hominae comparison I detect a total of 0.8 Mb of human-specific regions, slightly more than had been suggested in by Scally *et al.* [171]. However, almost half of the human-specific regions, 49.3% did not have a match to any sequence in *nt*. Of the remaining half that did, only a quarter, 26.2%, had a match to a primate. With orangutangs and macaques having the most hits, corresponding to 10% and 6% of all regions, respectively. These results contrast with the ones from the previous study, since now not only do I detect a slightly higher number of human-specific nucleotides, but they are also less likely to be found in other primates.

Although no enriched biological functions were found, both the GRCh38-specific and the T2T-specific regions intersect the three immunoglobulin loci: the light κ and λ chains and the heavy chains which are located in distinct chromosomes, chr2p11, chr22q11 and chr14q32 respectively. The immunoglobulins, together with the major histocompatibility complex, which also intersected the GRCh38-specific regions, form part of the adaptive immune system [172]. The adaptive immune system, which first appeared in the evolution of jawed vertebrates, can recognize and remember pathogens, thus speeding up the mobilization of the immune response [173]. Immunoglobulins (Igs) are a class of proteins that include both antibodies and cell-surface antigen receptors [66]. Igs molecules are composed by four peptide chains of two identical light chains and two identical heavy chains, the latter two chains determine the function of the antibody. Each chain, in turn, is composed by a variable and a constant amino acid sequence. The variability of the amino acid sequences is what gives rise to such a large diversity of antigen binding sites [66]. It is then understandable that a higher diversity and variability in the immunoglobulin locus can be advantageous. In fact, the immunoglobulin locus are highly instable in the hominids, with evidence for gene duplications and pseudogenes in several antibody classes [174]. This variation explains why the innate immune system was associated with the GRCh38 and T2T-specific regions but not the Human-specific regions. First, the evolutionary instability of these loci, means these regions are variable between hominids species, hence they can be detected in a 1-to-1 comparisons. Their high variability between individuals also means that these regions are not ubiquitous across all human genomes and thus are not human-specific

In the T2T-specific regions, no association could be found to the major histocompatibility complex. Instead, these regions contain keratin-associated proteins, genes which

are among some of the most rapidly diverging gene clusters in humans and chimpanzees [2]. Under the assumption that the T2T comparison is more reliable than the GRCh38 comparison due its lack of assembly gaps, the signal for the major histocompatibility complex in GRCh38 could be an artefact from the incomplete genome. This could be, for example, due to the variability of the locus which can make it harder to assemble [41]. Therefore, the conclusion from the T2T comparison is that human-specific regions are associated with immunoglobulins, olfactory receptors and keratin-associated proteins among the protein-coding genes.

Another observation was that several T2T human-specific regions were located in satellite or centromeric regions. Since both the T2T assembly and the chimpanzee reference genome contain no gaps in the reference, it might be expected that these regions can also be found in the chimpanzee genome. This is not the case because the centromere content of humans and chimpanzees is distinct, with chimpanzee centromeres being typically composed by single α -satellites High-Order Repeats (HORs) that are highly similar [43], while in humans centromeres are composed of α -satellites HORs of distinct k-mers repeats [175]. Thus, the human-specific regions that are located in highly repetitive regions identify α -satellite regions that are indeed human-specific.

As mentioned above, after including the remaining human and ape genomes in the comparison, the amount of human-specific regions drastically decreases from 15.8 Mb to 0.8 Mb, a reduction of 95.1%. Naturally, this also reduced the number of annotated promoters associated with these regions, and no biological function could be detected for these regions.

Similarly, although chimpanzee-specific regions do contain promoters associated with olfactory receptors, among others like metabolic functions, no enriched biological function was associated with these regions, regardless of whether the chimpanzee genome is compared to the GRCh38 or the T2T assembly. Instead, these regions also contain a high number of uncharacterized lncRNAs perhaps hinting at regulatory differences.

4.4.3 House mouse subspecies complex

In the subspecies comparison the total amount of subspecies-specific region varied between 1.9 Mb for *M. m. musculus* and 5.4 Mb for *M. m. castaneus*. As a fraction of genome length, the amount of subspecies-specific regions for *M. m. castaneus* is practically double to that of the other two subspecies, reaching 0.2% of the genome. while for *M. m. domesticus* and *M. m. musculus* its 0.09% and 0.08% respectively. Since the results

for both *M. m. musculus* and *M. m. domesticus* are similar, this may indeed reflect the underlying biology rather than an artefact of the genome length.

In this comparison, just like in the primates comparison, most of the detected regions intersect promoters whose gene is either “predicted”, with the possibility of encoding either RNA or protein-coding genes, or “uncharacterized”. The few protein-coding genes associated with these regions are also olfactory receptors showing their importance for individual identification among mammals [176].

In the second comparison, only 16 regions with 14.9 Kb could be found that are ubiquitous across all 15 laboratory strains and *M. m. domesticus*, but absent from the wild mouse species. The drastic reduction in regions after the intersection step, a 97.7% reduction from 4.1 Mb to 93.8 Kb, may reflect the variation present in the laboratory strains. Simultaneously, the fact that these species are very close, phylogenetically speaking, also reduces the likelihood of finding sufficiently diverged regions that can be detected by *Fur*.

4.4.4 lncRNAs and their function

Regardless of the comparison performed, one observation that remains is the high number of RNA genes associated with the lineage-specific regions in humans and mouse. Although some of these RNA genes have been classified, most of them remain uncharacterized as non-coding RNAs (ncRNAs). For a long time, it was assumed that RNA was merely an intermediate between DNA and protein-coding genes, with a minority of RNA products facilitating this function [177]. It is now known that there is a wide range of RNA molecules that perform diverse functions across the genome from small RNA molecules like miRNAs to lncRNAs, which are defined as ncRNAs that are at least 200 bp long [178–180]. Although ncRNAs are ubiquitous in large eukaryotic genomes, their exact function is often unknown. Furthermore, research into their function is often hampered by their wide definition. lncRNAs are arbitrarily defined as non-protein coding transcripts that are at least 200 bp, accordingly they have no shared function or composition [177]. lncRNAs and lincRNAs are not only widely expressed across the genome, they are also involved in regulatory mechanisms [181–183]. In fact, their dysregulation has been shown to be associated with several human diseases like tumors and even Alzheimer’s disease due to their interference in transcription regulation or mRNA stability among others [179].

The mounting evidence for the central role of lncRNA in the genome maintenance and regulation shows the need to better understand and classify these transcripts. The total number of functional lncRNAs in the human genome is largely unknown, because

lncRNA may not necessarily be expressed, may be mis-annotated or have incomplete open reading frames [177, 184]. Protocols like ChIRP-seq (chromatin isolation by RNA purification) helps establish the relationship between lncRNAs and their target regions [185], while databases like LNCinpedia [186] or RNAcentral [187], among many others (see [188]), store the available information. Experiments, while essential to properly annotate and describe lncRNAs, lag behind the massive number of lncRNA that are being annotated *in silico*, thus it is necessary to rely on predictive tools to search and describe lncRNAs, according to their characteristics. Predicting function, however, is difficult.

Tools for predicting the function of lncRNAs are based on four kinds of evidence: sequence alignment, gene co-expressions, lncRNA interactions and composite features such as chromatin states or expression patterns [188]. Since lncRNAs tend to be conserved neither in their sequence motifs nor their secondary structures, many tools favor the prediction of their function based on interactions between miRNA, mRNA and protein-coding genes. Currently most tools for automated annotation of lncRNAs rely on machine learning which requires accurate transcriptome data for training. Therefore, *de novo* annotation is difficult unless the training set is similar to what is being annotated [189].

Despite the biological importance of lncRNAs, and the efforts that have been made to identify them, it is still hard to predict their function. Because the study of lncRNAs is still in its infancy and empirical studies tend to focus on lncRNAs that are potentially involved in diseases or other traits of interest, it is perhaps not surprising that lineage-specific regions in humans and mouse contain a high number of uncharacterized lncRNAs.

4.4.5 *Fur* and unique satellite regions

Fur was originally designed to search for diagnostic markers in bacteria, whose genomes tend to be both shorter and less repetitive than eukaryotic organisms. If *Fur* is to be used to search for lineage-specific regions in mammals, or other organisms with repetitive genomes, it needs to be able to handle large repetitive regions, specially with the advent of telomere-to-telomere genomes.

To do so, *Fur* was modified such that now it can handle and process the masking information present in the neighbor genomes. If no masking information is present, or if masking is not requested in the second subtraction step, then *fur* behaves exactly as before. If, the masking information is applied, then target-specific regions will be aligned to unmasked regions in the BLAST database. While this does create the danger of picking neighbor repeats, this effect seems to be minimal. For example, in the Chimpanzee-GRCh38 un-

masked comparison a total of 164.8 Mb were considered to be species-specific, while in the Chimpanzee-T2T masked comparison it was 165.2 Mb. This increase corresponds to merely 0.01% of the chimpanzee genome.

4.5 Summary

So far, *Fur* had only been applied to search for diagnostic markers in prokaryotic genomes, as was shown in Chapter 3. Here I extend its application domain to the 1000 times larger genomes of mammals. I search for species-specific regions in two mammals: humans and house mice. For this I have employed *Fur* to find regions shared across all targets and absent from all neighbors.

The high density of lncRNAs in the lineage-specific regions may reflect their importance in regulatory mechanisms that so far are still relatively unknown. Additionally, in the case of the human-specific regions, they can also be associated to the human accelerated regions that have been associated with developmental enhancers.

I have also shown the importance of taking into account individual variation in the search for species-specific regions. By restricting the comparison to a pair of genomes, it becomes possible to detect highly variable divergent regions like immunoglobulins. In turn, by including individual variation in these types of comparisons it becomes possible to decouple intra from inter-species variation.

4.6 Supplementary material

Table S1: List of all annotated promoters in the Homininae comparison. Promoters are listed in alphabetical order according to their gene symbol.

Symbol	Description
AADAC	arylacetamide deacetylase
CEACAM21	CEA cell adhesion molecule 21
COG3	component of oligomeric golgi complex 3
FAM218A	family with sequence similarity 218 member A
HCP5	HLA complex P5
KRTAP13-3	keratin associated protein 13-3
KRTAP13-4	keratin associated protein 13-4
LCN10	lipocalin 10
LINC01480	long intergenic non-protein coding RNA 1480
LINC01815	long intergenic non-protein coding RNA 1815
LOC100128593	uncharacterized LOC100128593
LOC105370732	uncharacterized LOC105370732
LOC105372802	uncharacterized LOC105372802
LOC105377535	uncharacterized LOC105377535
LOC124902200	uncharacterized LOC124902200
MIR6808	microRNA 6808
MIR8074	microRNA 8074
OR6C76	olfactory receptor family 6 subfamily C member 76
OR7G2	olfactory receptor family 7 subfamily G member 2
SMARCD2	SWI/SNF related, matrix associated, actin dependent regulator of chromatin, subfamily d, member 2

Table S2: List of all annotated promoters in the DOM comparison. Promoters are listed in alphabetical order according to their gene symbol.

Symbol	Description
1700028I16Rik	RIKEN cDNA 1700028I16 gene
Agxt2	alanine-glyoxylate aminotransferase 2
Ahrr	aryl-hydrocarbon receptor repressor
Ankrd31	ankyrin repeat domain 31
Ccdc201	coiled coil domain 201
Cmah	cytidine monophospho-N-acetylneuraminic acid hydroxylase
Cox5a	cytochrome c oxidase subunit 5A
Fbxl17	F-box and leucine-rich repeat protein 17
Gm15731	predicted gene 15731
Gm24844	predicted gene, 24844
Gm26972	predicted gene, 26972
Gm32874	predicted gene, 32874
Gm33869	predicted gene, 33869
Gm35113	predicted gene, 35113
Gm35176	predicted gene, 35176
Gm38568	predicted gene, 38568
Gm39258	predicted gene, 39258
Gm40126	predicted gene, 40126
Gm41269	predicted gene, 41269
Gm51635	predicted gene, 51635
Gm52096	predicted gene, 52096
Gm53632	predicted gene, 53632
Gm54135	predicted gene, 54135
Gm6293	predicted pseudogene 6293
Gnb5	guanine nucleotide binding protein (G protein), beta 5
Hspa13	heat shock protein 70 family, member 13
Mir6377	microRNA 6377
Nfxl1	nuclear transcription factor, X-box binding-like 1
Or6c2	olfactory receptor family 6 subfamily C member 2
Pglyrp1	peptidoglycan recognition protein 1
Saxo4	stabilizer of axonemal microtubules 4
Taar7a	trace amine-associated receptor 7A
Ttc231	tetratricopeptide repeat domain 23-like
Vmn1r237	vomeroneasal 1 receptor 237
Zfp459	zinc finger protein 459

Table S3: List of all annotated promoters in the MUS comparison. Promoters are listed in alphabetical order according to their gene symbol.

Symbol	Description
Cbfb	core binding factor beta
Cyct	cytochrome c, testis
Gm20319	predicted gene, 20319
Gm23960	predicted gene, 23960
Gm24577	predicted gene, 24577
Gm25796	predicted gene, 25796
Gm27510	predicted gene, 27510
Gm32846	predicted gene, 32846
Gm35131	predicted gene, 35131
Gm39010	predicted gene, 39010
Helz	helicase with zinc finger domain
Mak16	MAK16 homolog
Or2ag16	olfactory receptor family 2 subfamily AG member 16
Or2z2	olfactory receptor family 2 subfamily Z member 2
Plekhl1	pleckstrin homology domain containing, family M (with RUN domain) member 1
Tmem156	transmembrane protein 156
Trav14-1	T cell receptor alpha variable 14-1

Table S4: List of all annotated promoters in the CAS comparison. Promoters are listed in alphabetical order according to their gene symbol.

Symbol	Description
1500012K07Rik	RIKEN cDNA 1500012K07 gene
Anxa2r2	annexin A2 receptor 2
Cndp1	carnosine dipeptidase 1
Gm11963	predicted gene 11963
Gm20822	predicted gene, 20822
Gm21810	predicted gene, 21810
Gm25270	predicted gene, 25270
Gm33130	predicted gene, 33130
Gm38656	predicted gene, 38656
Gm51701	predicted gene, 51701
Gm52115	predicted gene, 52115
Igkv6-32	immunoglobulin kappa variable 6-32
LOC118568478	uncharacterized LOC118568478
LOC118568479	uncharacterized LOC118568479
Miip	migration and invasion inhibitory protein
Or56b1	olfactory receptor family 56 subfamily B member 1
Pafah2	platelet-activating factor acetylhydrolase 2
Rbmyf8	RNA binding motif protein Y-linked family member 8
Upk1a	uroplakin 1A

Chapter 5

General discussion and future directions

Beatriz Vieira Mourato¹

Affiliations:

¹Max Planck Institute for Evolutionary Biology, RG Bioinformatics, Plön, Germany

This thesis is centered on identifying and annotating unique regions in genome sequences. Each of its three central chapters explores a distinct approach to finding unique genomic regions and how they can be associated with biological function. In Chapter 2 unique regions are shown to be highly enriched for developmental genes in mammalian genomes. In Chapter 3 unique regions are detected from target genomes compared to their phylogenetic neighbors to identify diagnostic markers of bacteria. Finally, in Chapter 4, I applied the same methods to detect lineage-specific regions in human and mouse.

The methods in these three chapters rely on exact matching, rather than classical alignments, to detect unique regions. In comparative genomics, methods based on exact matching are known as “alignment-free” sequence comparison [190, 191]. In Chapter 1 I explained that the alignment-free methods applied in this thesis can be regarded as the results of a successive simplification and specialization of classical alignments, which are optimal but slow, to exact matching which is less accurate but fast. Figure 2.9, which shows the difference in sensitivity between *RepeatMasker* and *Macle* is an example of the effects of this simplification. Heuristic aligners like *RepeatMasker* and *Blast* perform seed and extend algorithms, which gives them adequate sensitivity and speed. In contrast, programs like *MUMmer* and *Macle* essentially just perform the seeding step without the extension step, which makes them very fast but less sensitive.

The two programs I used to detect unique regions, *Macle* and *Fur*, search for maximal matches. In maximal matches homology is detected via the length of the match, where if the length of the match is indistinguishable from a random sequence, no homologs exist, and the region is unique. In Chapter 1, I explained how maximal matches can be looked up from suffix arrays and how to derive the null distribution of their lengths [19].

While calculating suffix arrays, and hence maximal matches, is fast, it consumes memory proportional to the length of the input string. Running *Macle* on single genomes as was done in Chapter 2, already required more than 100 GB of RAM. To run *Fur* on large samples of bacterial genomes or on multiple mammalian genomes as I do in Chapters 3 and 4, it was necessary to develop a more memory-efficient algorithm than the one implemented in the *Fur* version available at the beginning of my thesis [21].

In the following I summarize the findings from each of the three central chapters of this thesis. I also discuss the limitations of each chapter and how they might be overcome in future work.

5.1 Unique regions in mammalian genomes

In Chapter 2, I searched for highly complex or unique regions across 18 mammalian species representing nine mammalian orders. To achieve this, I employed the program *Macle* [8], which calculates the match complexity of a sequence to identify genomic regions present only once within a given genome.

Vertebrates have highly repetitive genomes that have not only undergone two rounds of whole genome duplication [192], which gave rise to the four *Hox* clusters, but also contain a high percentage of transposable elements like retrotransposons [15, 59, 60]. Thus, when searching for unique genomic regions within mammals, it is also necessary to define whether duplicated region like the *Hox* clusters, can be considered unique. In Chapter 2, I established that *Macle* can detect repeats that have occurred up until the divergence of placental mammals. Therefore, the identified unique genomic regions are free of any recent transposition or duplication event but may include more distant duplications like the *Hox* clusters.

5.1.1 Large-scale survey of mammalian genomes

The 18 species mammalian species were selected based on two criteria. First, only families with at least two available genomes were considered. Previously, the search for unique regions was restricted to mice and humans [8, 16], thus it was unknown whether similar patterns would be observed across the mammalian clade. Therefore, I decided to only look at families that had at least two genomes, to check the consistency of the results obtained. Moreover, to make sure that every mammalian order was equally represented, a restriction of two genomes per order was imposed. This is because available genomes are highly biased towards large-bodied and high-latitude species [73]. By end of 2023, 78 carnivore genomes had been assembled to either chromosome or complete level, while only 11 Afrotheria genomes had been assembled to the same level. To note that Afrotheria is a superorder, whereas Carnivora is a family from the Laurasiatheria superorder.

One of the main obstacles to large-scale genomic studies of mammals is the lack of available genome sequences, partly due to the difficulty in acquiring high-quality samples from wild animals. To address this, initiatives like the Zoonomia project [193] or the Vertebrates Genomes project [194] were launched to capture the diversity across mammalian and vertebrates species. As of 2nd September 2024, there were 3,459 mammalian genomes, 591 of which were assembled to either the chromosome or the complete level. From these 591 genomes, 122 alone were added since the beginning of 2024.

As the number of high-quality genomes available quickly increases, fine-grained large-scale comparative genomic studies have become more feasible. Thus, in the near future it should be possible to search for unique genomic regions across all mammalian orders. Additionally, as I have shown in Chapter 2 the association between unique regions and developmental genes is present across all mammals, therefore I can now investigate whether the biological functions associated with these regions are conserved among phylogenetic groups, or if they reflect changes in lifestyle, like locomotion.

5.1.2 Genome annotation

Another challenge in comparative genomics is the lack of available annotated genomes, particularly for non-model organisms. For example, of the 591 available mammalian genomes mentioned above, only 186 are annotated. In other words, 94.6% of the available mammalian genomes, are either poorly assembled, below chromosome level, or not annotated. While there are several tools for performing *de novo* annotation these must be carefully curated for each organism since there is no annotation pipeline that can be blindly applied to every organism while simultaneously obtaining reliable annotations.

Genome annotation by orthology, as done in Chapter 2, is considered more accurate than annotation by homology. Still, if the aim is to search for differences in gene functionality across distinct mammalian orders, or to search for lineage-specific regions as done in Chapter 4, then high quality species-specific genome annotations are essential.

5.1.3 Memory efficient *Macle*

Besides expanding the scope of my analysis to include more mammalian genomes, the tool I used to look up unique regions, *Macle*, can be improved. Currently, when calculating the macle index, *Macle* converts the entire genome into a suffix array, making the memory requirements proportional to the size of the genome. As it stands, *Macle* required between 139.8 to 238.1 GB of memory to index individual mammalian genomes. However, it may not be necessary to convert the entire genome at once. Following the logic applied in Chapter 3 to make *Fur* more memory-efficient, *Macle* can be modified to iteratively index each chromosome or contig at a time, reducing the memory requirements to that of the longest contig in the genome. A multi-threading algorithm can then be applied to the iterative *Macle* decreasing the runtime, albeit at an increase in memory consumption. This should bring the memory requirements of *Macle* closer to those of *Fur*, which required between 59.0 to 73.5 GB of memory to index a single mammalian neighbor genome in Chapter 4.

Unlike *Fur*, which compares a query to a subject sequence, *Macle* calculates the repetitiveness of a single sequence to itself. Thus, given a single genome with multiple contigs, an iterative version of *Macle* needs to 1) calculate the maximal match length of a contig to itself and 2) calculate the maximal match length between all other contigs.

Figure 5.1 illustrates how an iterative version of *Macle* may be implemented, where for each contig one index and two match length arrays are calculated, the intra (Figure 5.1A) and the inter (Figure 5.1B) match length arrays. The suffix array of Contig1 is illustrated as a suffix tree in Figure 5.1C, where the internal nodes of the tree are annotated with the length of the path label from the root to that node. The path length, or “string depth” of an internal node, v , that is a direct parent of two more leaves, denotes the match lengths for these leaves positions. Starting at leaf 1, the corresponding parental node has a string depth of 1. The match length of 1 is then written down in the intra match array at position 1. In this way, I then continue to fill out the intra match length arrays by iterating over all remaining leaves, 2,3,...,7. As a further example, Figure 5.1 shows how to find the lengths of the maximal matches starting at positions 4 and 5. This length is 2, the string depth of the parent, and is shown in blue in the intra match length array. This length corresponds to the string AA, shown in blue in the suffix tree.

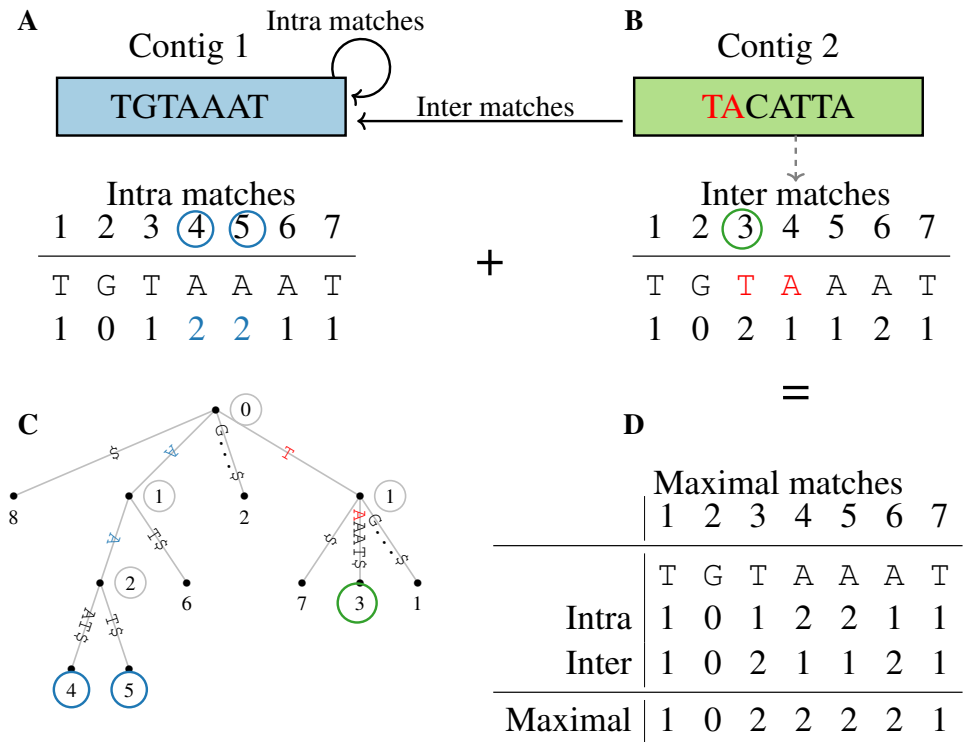


Figure 5.1: Scheme of how to calculate maximal matches lengths in the iterative version of *Macle*.

To get the inter matches, we “stream” Contig2 against the suffix tree. Starting at the first position of Contig2, two characters are matched, TA, marked in red in Figure 5.1B, into the suffix tree, also marked in red in Figure 5.1C. The match TA ends on the branch to leaf 3, circled in green, which means that TA can be found at the third position of Contig1. Thus we note a length of 2 at the third position of the inter array, also circled in green. Then we repeat the streaming starting at the second position of Contig2, A, and find matches with a length of 1 at positions 4,5 and 6, and so on. Whenever an entry greater than the previous one is found, the greater one is stored.

In other words, Intra match lengths are found by iterating over the leaves, and looking up the parent string depth while inter match lengths are found by matching from the root.

The described streaming algorithm resumes matching at every position of Contig2 and continues until the first mismatch. In highly similar genomes this would lead to a quadratic runtime. However, the algorithm is closely related to the classical algorithm of matching statistics, which runs in linear time [22, p.132ff].

Finally, the intra and inter match length arrays can be summarized into a maximal match length array by extracting the largest match length for each position i of the sequence.

This new indexing algorithm for *Macle* would require memory proportional to the longest contig, rather than proportional to the combined lengths of all contigs. Still, even if each inter-contig comparison runs in linear time, the streaming aspect remains quadratic in the number of contigs. So smaller contigs should be aggregated into larger contigs to reduce this contig-dependent quadratic behaviour.

5.2 Markers in prokaryotes

In Chapter 3 I developed a pipeline, *Mapro*, for automatic identification of diagnostic markers in prokaryote genomes. The pipeline links three different packages: *Neighbors*, *Fur* and *Prim*. The main package is *Fur*, while *Neighbors* curates the input data and *Prim* designs and evaluates primers based on the marker regions given by *Fur*.

Version 3.2 of *Fur* implemented the *Macle* algorithm and indexed the entire neighborhood. Hence, its memory requirements were proportional to the size of the neighborhood. If I consider a 5 Mb *Escherichia coli* genome, a neighborhood composed of 100 *E.coli* genomes would be 500 Mb long, larger than chromosome 1 in Humans. To expand the applicability of *Fur* it was necessary to reduce its memory requirements so that it could handle both large neighborhoods (Chapter 3) and large genomes (Chapter 4). Version 4.2

of *Fur* iterates over the neighbor sequences such that its memory requirements is now proportional to the size of the longest sequence of the neighborhood. Additionally, since the new algorithm does not calculate intra matches, marker regions no longer need to be unique within the target genome.

5.2.1 Navigating taxonomic databases

Pipelines that rely exclusively on taxonomical information are often criticized due to the presence of misclassified sequences or organisms in taxonomic databases. This, together with the manual curation of input data, was another limitation of *Fur*. Given that *Fur* is highly sensitive to the composition of the neighborhood, the package *Neighbors* was developed. *Neighbors* relies on the taxonomical and genome information available from NCBI databases to list all the available target and neighbor genomes for a given organism. This ensures that all available target and neighbor genomes can be taken as input for *Fur*. However, if taxonomic information is not reliable, how do I accurately distinguish between target and neighbor genomes? *Neighbors* solves this through the program *fintac*. *fintac* finds the phylogenetic clade of the dataset that maximizes the number of taxonomic targets while minimizing the number of taxonomical neighbor genomes. Thus, while the taxonomy of the organism is still important, misclassified sequences will be grouped according to their phylogeny, not taxonomy.

Prokaryotic organisms are considered different species if their ANI score is below 96% [124]. In Chapter 3 I briefly compared the tool *fastANI*, which calculates the ANI score, with the phylogenetic groups obtained from *phylonium* and the program *fintac*. *phylonium* and *fintac* were highly accurate and 400x faster requiring half the memory for the same database of 1201 sequences. Thus, in the future it might also be interesting to quantify the proportion of bacterial genomes that are misclassified in the database, and whether a given taxonomical group has a higher rate of misclassification or whether this is a general observation that can be applied to the entire prokaryote taxonomical database of the NCBI.

Another difficulty in navigating prokaryotic taxonomic databases is the inconsistency in taxonomical ranks across organisms. In the example given in Chapter 3, Figure 3.3, the strain *E.coli* O157:H7 is shown to have an extra taxonomical rank “serotype” between “species” and “strain”. However, another strain from the same species *E.coli* IAI39, does not have the taxonomical ranking “serotype”. I have solved this issue in the *Mapro* pipeline by climbing two taxonomic ranks before obtaining the full list of available genomes associated with a given organism. This approach ensures that the search

for diagnostic markers includes all strains of a specific species and not merely a subset. It would be interesting, however, to quantify what are the taxonomical units being compared by *Mapro*, and whether there is a better way to overcome this inconsistency in bacterial taxonomy.

5.2.2 In-depth quantification of marker regions

For the functional annotation of the marker regions detected in 120 bacteria, I restricted the analysis to the top 10 organisms with the highest number of target genomes for which primers could be designed. Of these 10 organisms, only three were represented in the GO databases. Ultimately, the functional analysis could only be performed for *Pseudomonas aeruginosa* since the annotated genes for the other two organisms were not present in the GO database. This was a relatively straightforward analysis that can be expanded on the following ways.

First, it would be informative to have an overview of the annotated genes that were found in the marker regions for the other 110 bacteria. Secondly, the lack of entries in the GO database for certain organisms could be overcome by associating known functions from orthologous genes as was done in Chapter 2. By taking these two steps I could gain a broader insight into the genes, and their respective functions, located in marker regions. Ultimately, this could result in a list of distinctive genes - in pathogens, these might include drug targets - for each organism.

5.3 Lineage-specific regions in mammalian genomes

In Chapter 4 I used *Fur* to search for lineage-specific regions in primates and mouse genomes that might give insight into species-specific genome regions.

The chapter starts by comparing the telomere to telomere (T2T) assembly of the human genome to the GRCh38 reference genome. As the first fully complete human assembly, the T2T assembly is used as the target representative for all primate comparisons based on the assumption that the obtained T2T-specific regions are a superset of the GRCh38-specific regions.

Instead of using exclusively the T2T assembly, I compared both the GRCh38 and the T2T assembly. This is because, despite its quality, the T2T assembly is relatively new and is not a reference genome. Reference genomes are widely used and contain a large amount of associated data from numerous studies. Therefore, moving from one reference genome

to another is not simple. The previous human reference genome, hg19, was replaced by GRCh38 in 2013, more than a decade ago. Despite this some, clinical laboratories continue to use the hg19 reference genome. This is because of existing pipelines that would not only need to be revalidated, but the existing data itself would also need to be curated [195].

Thus, although adapting and using improved assemblies is beneficial due to the better resolution of problematic regions, older reference genomes, which contain a high amount of associated data, cannot be immediately discarded. This is why I replicated all comparisons with T2T in GRCh38, even though the results were very similar.

5.3.1 Further assessments on the masking step of *Fur*

The search for chimpanzee-specific regions was performed using two distinct *Fur* versions: version 4.2 and version 4.3. The chimpanzee-GRCh38 comparison used *Fur* version 4.2 developed in Chapter 3 as when comparing the Chimpanzee reference genome and the T2T assembly, it was necessary to use a masked Blast database in the second subtraction step. Although the results from both comparison were very similar, no control comparison was performed. Such control can be performed by comparing the results from the chimpanzee-GRCh38 comparison with and without masking the Blast database. This will help to fully assess the impact of the masking used in the second subtraction and to further evaluate the addition of the masking option to *Fur*. Additionally, this also allows to determine whether there is any shared repetitive region that was dropped in the unmasked comparison, but was considered target-specific in the masked comparison.

5.3.2 Individually variable regions

Another point that needs to be addressed in my analysis of lineage-specific regions in mammals is the difference between individual and population level variation. *Fur* searches for genomic regions that are shared across a sample of genomes but absent in the neighbors. However, it cannot identify genomic regions that are absent in the neighbors but variable within the targets. Figure 5.2 illustrates this distinction between marker regions, in red, and variable target-specific regions in blue, where markers are a subset of all existing individual variation. Naturally, if the target set of *Fur* consists of a single genome, then the target-specific regions reflect this individual variation. For example, when the GRCh38 or the T2T human assembly were compared to the common chimpanzee, several promoters associated with the adaptive immune system could be annotated. However, as the number of targets genomes increased, these promoters could no longer be identi-

fied, because currently *Fur* only identifies target-specific regions that are present in all the target genomes. Species-specific regions that vary between target genomes are excluded.

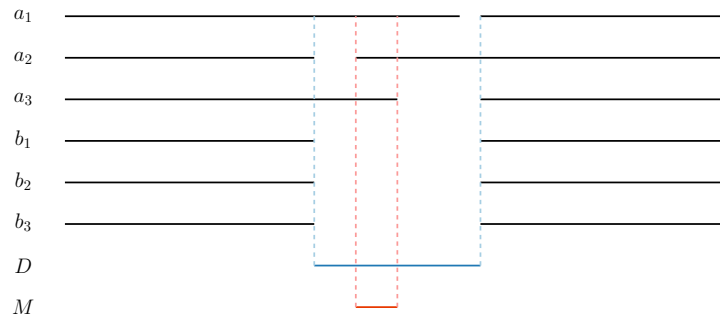


Figure 5.2: Multiple sequence alignment of A and B illustrating their difference, D , as opposed to the diagnostic marker, M , of A .

When comparing the 11 human genomes against the common chimpanzee reference genome, 83.9% of the human-specific regions were lost during the intersection step. This means that more than fourth-fifths of the human-specific regions found in the T2T genome were not shared across all the other individuals. In the mouse comparison, only 2.3% of the unique regions identified in the GRCm39 mouse reference genome were shared across all other laboratory strains and *M. m. domesticus*. One might think that all that is needed is to remove the intersection step of *Fur*, however this is not the case. The target specific regions detected by *Fur* are all mapped to a representative sequence, meaning that even if the intersection step were to be removed, only variable regions present in the target representative would be found. If the goal is to search for variable regions in all the targets, it is necessary to develop a tool that can 1) search for target-specific regions across all target genomes and 2) does not perform the intersection step. Resulting in a list of target-specific variable regions for each target genome.

Since most of the target genomes will not be annotated, these regions can be annotated through homology searches in databases like *nt*. The mapping of these regions to a single reference genome should be avoided, as that can lead to the filtering of variable regions that are absent from the reference.

5.3.3 Characterization of lncRNA

The species-specific regions detected in Chapter 4, whether for human, chimpanzee or mouse, were highly associated with RNA molecules. However the function of these molecules is largely unknown. These molecules can be characterized with the aid of empirical studies such as CRISPR-CAS or site-directed mutagenesis, as well by associat-

ing these regions with epigenetic markers published, for example, as part of the ENCODE project [196].

5.3.4 Lineage-specific accelerated regions (linARS)

In 2006, it was found that several conserved mammalian regions in the human genome show significantly accelerated substitution rates in comparison to chimpanzees [197, 198]. These human accelerated regions, or HARs, were mostly located in either non-coding regions or intronic regions, with only 1.5% overlapping coding genes and possibly playing a role in transcription regulation [198]. Further studies showed that these regions are mostly located in non-coding regions or near genes involved in developmental and regulatory processes. Specifically, 30% of these regions were associated with developmental enhancers [199].

HARs are not restricted to humans and have also been identified in a wide range of primates, where they are called lineage-specific accelerated regions (linARs) [200, 201]. Regardless of the name, the same conclusions hold as linARs are located near regulatory DNAs associated with development [200], while Homininae linARs were significantly associated with sensory perception [201].

It might thus be interesting to explore the association between the lineage-specific regions described in Chapter 4 and mutation rates. From there, I can evaluate similarities, or dissimilarities, between these two types of genomic regions, specially when it comes to associations with regulatory regions and olfactory receptors.

5.4 Further improvements on *Fur*

In Chapter 3 and 4 of this thesis I explored different applications of *Fur*. To do so, it was necessary to improve several aspects of the program.

In Chapter 3, version 4.2 of *Fur*, was developed with memory requirements proportional to the size of the longest neighbor sequence. This allowed *Fur* to search for target-specific regions in large genomes and/or large neighborhoods. In Chapter 4, version 4.3 of *Fur* was developed to include the masking information.

There are, however, other issues that were not addressed in this thesis, and should be the focus of future work on *Fur*. The first issue is related to the identification of target-specific regions during the first subtraction step. An interval is considered to be target-specific if the number of matches in a window length w , divided by a quantile of the match length

probability distribution q is greater than a threshold t as shown in the equation

$$t = w/q + 1$$

Generally speaking, the intervals identified with *Fur* are highly accurate. However, there is an exception. When comparing single target and neighbor sequences, *Fur* tends to underestimate the amount of marker material obtained for shorter sequences (less than <1 Mb long). Because by default *Fur* uses the 0.1 quantile only windows with consistently short matches are considered unique. Since *Fur* was initially designed to search for diagnostic markers, a more conservative approach was preferred when it came to the identification of marker regions. However, as the applications of *Fur* expand beyond the identification of marker regions, it might be time to reconsider some parameters. In this case, the missing marker material can be recovered by applying a higher quantile as shown in Figure 5.3. Thus, it is necessary to rethink the importance of the applied quantile and evaluate whether there is an ideal quantile that should be applied instead of the current 0.1 quantile.

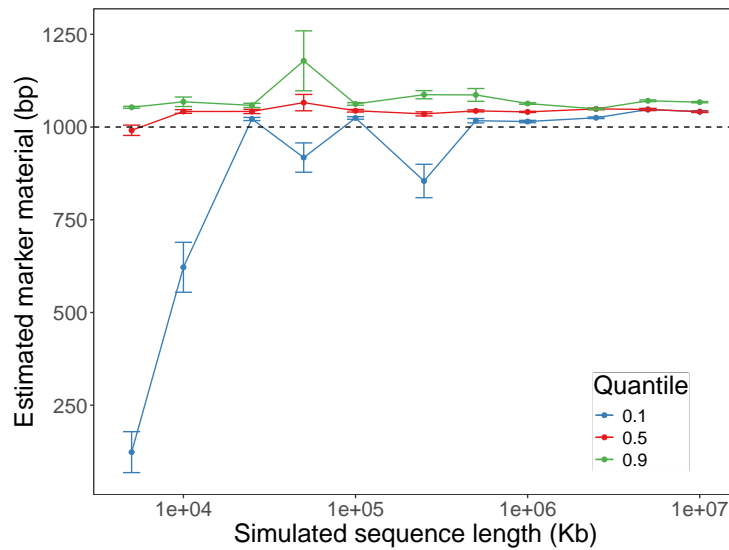


Figure 5.3: Obtained marker material after the first subtraction step with *Fur* for a single target and neighbor sequence for three different quantile values. Mean \pm SEM of 10 simulations are shown.

Another topic that can be further expanded on is the suffix-array algorithm implemented in version 4.2 of *Fur*. Just like suffix-arrays, Burrows-Wheeler indexes are fast and have reduced memory consumption [202, Ch9][203, 204]. Therefore, it would be beneficial to compare the efficiency of suffix arrays and Burrows-Wheeler indexes.

One example of a Burrows-Wheeler index is *MONI* [204]. Compared to other aligners, like BWA-MEM [205], *MONI* was shown to be simultaneously faster and consume less memory. Indeed, when compared to *makeFurDb*, *MONI* does requires less memory (Figure 5.4A), however it is slower (Figure 5.4B). This is a classic case where there was a trade-off between runtime and memory usage. Since memory is often a more limiting resource than time, it might still be beneficial to implement a Burrows-Wheeler index into *Fur*.

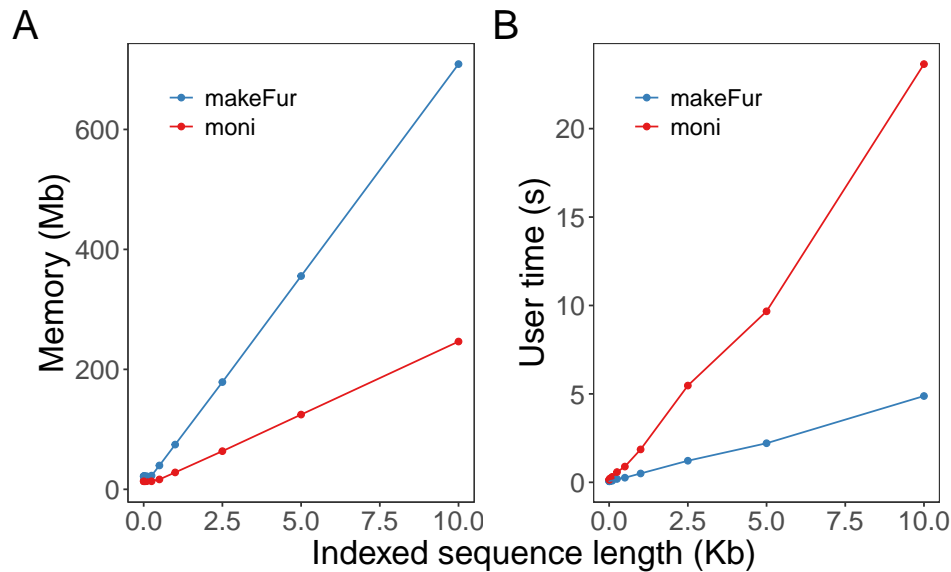


Figure 5.4: Memory (A) and User time (B) required for indexing a single target sequence with *makeFurDb* and *moni*. The memory of *moni* corresponds to the maximum observed for indexing and computing the matching statistics while the user time is the sum of these two steps.

5.5 General conclusion

This thesis aimed to explore the identification and role of unique genomic regions in large genomic datasets while simultaneously assessing existing software for finding unique genomic regions.

Throughout this thesis, I demonstrate that unique genomic regions can give insight into developmental factors (Chapter 2), phenotypes (Chapter 3), and regulatory regions (Chapter 4). Additionally, I have made available all the information regarding the identified unique regions, so that further work can be done to identify and annotate these regions, especially in non-model organisms.

Finally, to search for unique regions on such a large scale and in such highly diverse datasets, like bacteria and mammals, it was necessary to develop, test and adapt currently existing software tools like *Fur*. While several issues have been addressed and overcome throughout this thesis, there is still room for improvement of existing tools.

Bibliography

- [1] Bernhard Haubold and Thomas Wiehe. *Introduction to computational biology: an evolutionary approach. An Evolutionary Approach*. SpringerLink. Basel: Springer Science & Business Media, 2006. 328170 pp. ISBN: 978-3-7643-6700-8.
- [2] The Chimpanzee Sequencing and Analysis Consortium. “Initial sequence of the chimpanzee genome and comparison with the human genome”. In: *Nature* 437.7055 (2005), pp. 69–87. DOI: 10.1038/nature04072.
- [3] Adrian Bird et al. “A fraction of the mouse genome that is derived from islands of nonmethylated, CpG-rich DNA”. In: *Cell* 40.1 (1985), pp. 91–99. ISSN: 0092-8674. DOI: 10.1016/0092-8674(85)90312-5.
- [4] Adrian P Bird. “CpG-rich islands and the function of DNA methylation”. In: *Nature* 321.6067 (1986), pp. 209–213. ISSN: 0028-0836. DOI: 10.1038/321209a0.
- [5] ENCODE Project Consortium et al. “An integrated encyclopedia of DNA elements in the human genome”. In: *Nature* 489.7414 (2012), p. 57. DOI: 10.1038/nature11247.
- [6] Elisabeth Wachter et al. “Synthetic CpG islands reveal DNA sequence determinants of chromatin structure”. In: *Elife* 3 (2014), e03397. DOI: 10.7554/elife.03397.
- [7] Navin Elango and Soojin V Yi. “Functional relevance of CpG island length for regulation of gene expression”. In: *Genetics* 187.4 (2011), pp. 1077–1083. DOI: 10.1534/genetics.110.126094.
- [8] Anton Pirogov et al. “High-complexity regions in mammalian genomes are enriched for developmental genes”. In: *Bioinformatics* 35.11 (2019), pp. 1813–1819. ISSN: 1460-2059. DOI: 10.1093/bioinformatics/bty922.
- [9] Mouse Genome Sequencing Consortium. “Initial sequencing and comparative analysis of the mouse genome”. In: *Nature* 420.6915 (Dec. 2002), pp. 520–562. ISSN: 1476-4687. DOI: 10.1038/nature01262.
- [10] Stephen F. Altschul et al. “Basic local alignment search tool”. In: *Journal of Molecular Biology* 215.3 (1990), pp. 403–410. ISSN: 0022-2836. DOI: 10.1016/S0022-2836(05)80360-2.
- [11] Christiam Camacho et al. “BLAST+: Architecture and applications”. In: *BMC Bioinformatics* 10.1 (Dec. 2009), p. 421. ISSN: 1471-2105. DOI: 10.1186/1471-2105-10-421.
- [12] Bernhard Haubold and Angelika Börsch-Haubold. *Bioinformatics for Evolutionary Biologists: A problems approach*. Second edition. Literaturangaben: Seite 401-403. Cham: Springer, Dec. 21, 2022. 411 pp. ISBN: 978-3-031-20413-5. DOI: 10.1007/978-3-319-67395-0.
- [13] Bernhard Haubold and Thomas Wiehe. “How repetitive are genomes?” In: *BMC Bioinformatics* 7.1 (2006), p. 541. ISSN: 1471-2105. DOI: 10.1186/1471-2105-7-541.

- [14] Guillaume Marçais et al. “MUMmer4: A fast and versatile genome alignment system”. In: *PLoS computational biology* 14.1 (2018), e1005944. DOI: 10.1371/journal.pcbi.1005944.
- [15] International Human Genome Sequencing Consortium. “Initial sequencing and analysis of the human genome”. In: *Nature* 409 (6822 2001). ISSN: 0028-0836. DOI: 10.1038/35057062.
- [16] Cas Simons et al. “Transposon-free regions in mammalian genomes”. In: *Genome Research* 16.2 (Feb. 2006), pp. 164–172. DOI: 10.1101/gr.4624306.
- [17] Cas Simons et al. “Maintenance of transposon-free regions throughout vertebrate evolution”. In: *BMC Genomics* 8.1 (2007), p. 470. ISSN: 1471-2164. DOI: 10.1186/1471-2164-8-470.
- [18] Christopher Pockrandt et al. “GenMap: ultra-fast computation of genome mappability”. In: *Bioinformatics* 36.12 (2020), pp. 3687–3692. DOI: 10.1093/bioinformatics/btaa222.
- [19] Bernhard Haubold et al. “Estimating Mutation Distances from Unaligned Genomes”. In: *Journal of Computational Biology* 16 (2009), pp. 1487–1500. DOI: 10.1089/cmb.2009.0106.
- [20] Pavel Beran et al. “KEC: unique sequence search by K-mer exclusion”. In: *Bioinformatics* 37.19 (2021), pp. 3349–3350. DOI: 10.1093/bioinformatics/btab196.
- [21] Bernhard Haubold et al. “Fur: Find Unique Genomic Regions for Diagnostic PCR”. In: *Bioinformatics* (2021), pp. 1–8. ISSN: 1367-4803. DOI: 10.1093/bioinformatics/btab059.
- [22] Dan Gusfield. *Algorithms on Strings, Trees and Sequences: computer science and computational biology*. 12. print. Cambridge: Cambridge University Press, 1997. 534 pp. ISBN: 978-0-521-58519-4. DOI: 10.1017/cbo9780511574931.
- [23] Enno Ohlebusch. *Bioinformatics Algorithms: Sequence analysis, genome rearrangements, and phylogenetic reconstruction*. Ulm: Oldenbusch Verlag, 2013. 604 pp. ISBN: 978-3-00-041316-2.
- [24] Frederick Sanger et al. “The nucleotide sequence of bacteriophage ϕ X174”. In: *Journal of molecular biology* 125.2 (1978), pp. 225–246. DOI: 10.1016/0022-2836(78)90346-7.
- [25] Frederick Sanger et al. “Nucleotide sequence of bacteriophage λ DNA”. In: *Journal of molecular biology* 162.4 (1982), pp. 729–773. DOI: 10.1016/0022-2836(82)90546-0.
- [26] Robert D Fleischmann et al. “Whole-genome random sequencing and assembly of Haemophilus influenzae Rd”. In: *Science* 269.5223 (1995), pp. 496–512. DOI: 10.1126/science.7542800.
- [27] André Goffeau et al. “Life with 6000 genes”. In: *Science* 274.5287 (1996), pp. 546–567. DOI: 10.1126/science.274.5287.546.
- [28] C. elegans Sequencing Consortium*. “Genome sequence of the nematode C. elegans: a platform for investigating biology”. In: *Science* 282.5396 (1998), pp. 2012–2018. DOI: 10.1126/science.282.5396.2012.
- [29] Arabidopsis Genome Initiative genomeanalysis. “Analysis of the genome sequence of the flowering plant Arabidopsis thaliana”. In: *Nature* 408.6814 (2000), pp. 796–815. DOI: 10.1038/35048692.
- [30] Mark D Adams et al. “The genome sequence of Drosophila melanogaster”. In: *Science* 287.5461 (2000), pp. 2185–2195. DOI: 10.1126/science.287.5461.2185.
- [31] J Craig Venter et al. “The sequence of the human genome”. In: *Science* 291.5507 (2001), pp. 1304–1351. DOI: 10.1373/clinchem.2014.237016.
- [32] Francis S Collins et al. “A vision for the future of genomics research”. In: *Nature* 422.6934 (2003), pp. 835–847. DOI: 10.1038/nature01626.
- [33] Richard A Gibbs. “The human genome project changed everything”. In: *Nature Reviews Genetics* 21.10 (2020), pp. 575–576. DOI: 10.1038/s41576-020-0275-3.

- [34] Kerstin Lindblad-Toh et al. “Genome sequence, comparative analysis and haplotype structure of the domestic dog”. In: *Nature* 438.7069 (2005), pp. 803–819. DOI: 10.1038/nature04338.
- [35] Rat Genome Sequencing Project Consortium. “Genome sequence of the Brown Norway rat yields insights into mammalian evolution”. In: *Nature* 428.6982 (2004), pp. 493–521. DOI: 10.1038/nature02426.
- [36] Eric S Lander. “Initial impact of the sequencing of the human genome”. In: *Nature* 470.7333 (2011), pp. 187–197. DOI: 10.1038/nature09792.
- [37] National Center for Biotechnology Information Bethesda (MD): National Library of Medicine (US). *National Center for Biotechnology Information (NCBI)[Internet]*. 2024. URL: <https://www.ncbi.nlm.nih.gov/genbank/statistics/>.
- [38] Miten Jain et al. “Nanopore sequencing and assembly of a human genome with ultra-long reads”. In: *Nature biotechnology* 36.4 (2018), pp. 338–345. DOI: 10.1038/nbt.4060.
- [39] Miten Jain et al. “Linear assembly of a human centromere on the Y chromosome”. In: *Nature biotechnology* 36.4 (2018), pp. 321–323. DOI: 10.1038/nbt.4109.
- [40] Karen H. Miga et al. “Telomere-to-telomere assembly of a complete human X chromosome”. In: *Nature* 585.7823 (2020), pp. 79–84. ISSN: 1476-4687. DOI: 10.1038/s41586-020-2547-7.
- [41] Sergey Nurk et al. “The complete sequence of a human genome”. In: *Science* 376.6588 (2022), pp. 44–53. DOI: 10.1126/science.abj6987.
- [42] Kateryna D Makova et al. “The complete sequence and comparative analysis of ape sex chromosomes”. In: *Nature* (2024), pp. 1–11. DOI: 10.1101/2023.11.30.569198.
- [43] DongAhn Yoo et al. “Complete sequencing of ape genomes”. In: *bioRxiv* (2024), pp. 2024–07. DOI: 10.1101/2024.07.31.605654..
- [44] Brandi L Cantarel et al. “MAKER: an easy-to-use annotation pipeline designed for emerging model organism genomes”. In: *Genome research* 18.1 (2008), pp. 188–196. DOI: 10.1101/gr.6743907.
- [45] Peter W Harrison et al. “Ensembl 2024”. In: *Nucleic acids research* 52.D1 (2024), pp. D891–D899. DOI: 10.1093/nar/gkad1049.
- [46] Mark Yandell and Daniel Ence. “A beginner’s guide to eukaryotic genome annotation”. In: *Nature Reviews Genetics* 13.5 (2012), pp. 329–342. DOI: 10.1038/nrg3174.
- [47] Nancy Manchanda et al. “GenomeQC: a quality assessment tool for genome assemblies and gene structure annotations”. In: *BMC genomics* 21.1 (2020), pp. 1–9. DOI: 10.1186/s12864-020-6568-2.
- [48] Felipe A Simão et al. “BUSCO: assessing genome assembly and annotation completeness with single-copy orthologs”. In: *Bioinformatics* 31.19 (2015), pp. 3210–3212. DOI: 10.1093/bioinformatics/btv351.
- [49] April A Jauhal and Richard D Newcomb. “Assessing genome assembly quality prior to downstream analysis: N50 versus BUSCO”. In: *Molecular Ecology Resources* 21.5 (2021), pp. 1416–1421. DOI: 10.1111/1755-0998.13364.
- [50] Girum Fitihamlak Ejigu and Jaehee Jung. “Review on the computational genome annotation of sequences obtained by next-generation sequencing”. In: *Biology* 9.9 (2020), p. 295. DOI: 10.3390/biology9090295.
- [51] Jaime Huerta-Cepas et al. “eggNOG 5.0: a hierarchical, functionally and phylogenetically annotated orthology resource based on 5090 organisms and 2502 viruses”. In: *Nucleic Acids Research* 47.D1 (Nov. 2018), pp. D309–D314. ISSN: 0305-1048. DOI: 10.1093/nar/gky1085. eprint:

<https://academic.oup.com/nar/article-pdf/47/D1/D309/27437484/gky1085.pdf>.

- [52] Michael Ashburner et al. "Gene ontology: Tool for the unification of biology". In: *Nature Genetics* 25.1 (2000), pp. 25–29. ISSN: 1061-4036. DOI: 10.1038/75556.
- [53] TGO Consortium et al. "The gene ontology knowledgebase in 2023". In: *Genetics* 224.1 (2023). DOI: 10.1093/genetics/iyad031.
- [54] Beatriz Vieira Mourato and Bernhard Haubold. "Detection and annotation of unique regions in mammalian genomes". In: *G3* accepted (2024). DOI: 10.1101/2024.10.11.617789.
- [55] Alan M Weiner. "SINEs and LINEs: the art of biting the hand that feeds you". In: *Current opinion in cell biology* 14.3 (2002), pp. 343–350. DOI: 10.1016/s0955-0674(02)00338-1.
- [56] Astrid Böhne et al. "Transposable elements as drivers of genomic and biological diversity in vertebrates". In: *Chromosome research* 16 (2008), pp. 203–215. DOI: 10.1007/s10577-007-1202-6.
- [57] Zoltán Ivics et al. "Transposon-mediated genome manipulation in vertebrates". In: *Nature methods* 6.6 (2009), pp. 415–422. DOI: 10.1038/nmeth.1332.
- [58] Cédric Feschotte and Ellen J Pritham. "DNA transposons and the evolution of eukaryotic genomes". In: *Annu. Rev. Genet.* 41 (2007), pp. 331–368. DOI: 10.1146/annurev.genet.40.110405.090448.
- [59] Margaret G Kidwell and Damon R Lisch. "Perspective: transposable elements, parasitic DNA, and genome evolution". In: *Evolution* 55.1 (2001), pp. 1–24. DOI: 10.1554/0014-3820(2001)0550001:ptepda2.0.co;2.
- [60] C Miskey et al. "Review DNA transposons in vertebrate functional genomics". In: *Cellular and molecular life sciences* 62 (2005), pp. 629–641. DOI: 10.1007/s00018-004-4232-7.
- [61] James R Lupski. "Retrotransposition and structural variation in the human genome". In: *Cell* 141.7 (2010), pp. 1110–1112. DOI: 10.1016/j.cell.2010.06.014.
- [62] John F McDonald. "Transposable elements, gene silencing and macroevolution". In: *Trends in Ecology & Evolution* 13.3 (1998), pp. 94–95. DOI: 10.1016/s0169-5347(97)01282-2.
- [63] Roy J Britten. "DNA sequence insertion and evolutionary variation in gene regulation." In: *Proceedings of the National Academy of Sciences* 93.18 (1996), pp. 9374–9377. DOI: 10.1073/pnas.93.18.9374.
- [64] Anna D Senft and Todd S Macfarlan. "Transposable elements shape the evolution of mammalian development". In: *Nature Reviews Genetics* 22.11 (2021), pp. 691–711. ISSN: 1471-0064. DOI: 10.1038/s41576-021-00385-1.
- [65] Simone Hoegg and Axel Meyer. "Hox clusters as models for vertebrate genome evolution". In: *Trends in Genetics* 21.8 (2005), pp. 421–424. ISSN: 0168-9525. DOI: 10.1016/j.tig.2005.06.004.
- [66] Bruce Alberts et al. *Molecular biology of the cell*. Ed. by Rebecca Heald et al. 7th edition. "Not for sale in the United States or Canada" (Cover). New York, NY: W. W.Norton & Company, 2022. 14043770 pp. ISBN: 9780393884821. DOI: 10.1201/9781315735368.
- [67] Arian Smith, Robert Hubley, and P Green. *RepeatMasker Open-4.0*. 2013.
- [68] Elgion LS Loreto et al. "The good, the bad and the ugly of transposable elements annotation tools". In: *Genetics and Molecular Biology* 46.3 Suppl 1 (2023). DOI: 10.1590/1678-4685-gmb-2023-0138.

- [69] Jessica Storer et al. “The Dfam community resource of transposable element families, sequence models, and genome annotations”. In: *Mobile DNA* 12.1 (2021), pp. 1–14. DOI: 10.1186/s13100-020-00230-y.
- [70] Austin B Osmanski et al. “Insights into mammalian TE diversity through the curation of 248 genome assemblies”. In: *Science* 380.6643 (2023), eabn1430. DOI: 10.1126/science.abn1430.
- [71] Abraham Lempel and Jacob Ziv. “On the complexity of finite sequences”. In: *IEEE Transactions on information theory* 22.1 (1976), pp. 75–81. DOI: 10.1109/tit.1976.1055501.
- [72] Linda Odenthal-Hesse et al. “hotspot: software to support sperm-typing for investigating recombination hotspots”. In: *Bioinformatics* 32.16 (2016), pp. 2554–2555. DOI: 10.1093/bioinformatics/btw195.
- [73] Nathan S Upham and Michael J Landis. “Genomics expands the mammalverse”. In: *Science* 380.6643 (2023), pp. 358–359. DOI: 10.1126/science.add2209.
- [74] Annabel Whibley, Joanna L Kelley, and Shawn R Narum. “The changing face of genome assemblies: Guidance on achieving high-quality reference genomes”. In: *Molecular Ecology Resources* 21.3 (Apr. 2021), pp. 641–652. ISSN: 1755-098X. DOI: 10.1111/1755-0998.13312.
- [75] Amanda Warr et al. “An improved pig reference genome sequence to enable pig genetics and genomics research”. In: *Gigascience* 9.6 (2020), giaa051. DOI: 10.1093/gigascience/giaa051.
- [76] Joan K Lunney et al. “Importance of the pig as a human biomedical model”. In: *Science translational medicine* 13.621 (2021), eabd5758. DOI: 10.1126/scitranslmed.abd5758.
- [77] Lincoln Stein. “Genome annotation: from sequence to biology”. In: *Nature reviews genetics* 2.7 (2001), pp. 493–503. DOI: 10.1038/35080529.
- [78] The Genome Reference Consortium. *Major update to the mouse reference assembly*. June 2020.
- [79] Jaime Huerta-Cepas et al. “Fast genome-wide functional annotation through orthology assignment by eggNOG-mapper”. In: *Molecular biology and evolution* 34.8 (2017), pp. 2115–2122. DOI: 10.1093/molbev/msx148.
- [80] Carlos P Cantalapiedra et al. “eggNOG-mapper v2: functional annotation, orthology assignments, and domain prediction at the metagenomic scale”. In: *Molecular biology and evolution* 38.12 (2021), pp. 5825–5829. DOI: 10.1093/molbev/msab293.
- [81] National Center for Biotechnology Information Bethesda (MD): National Library of Medicine (US). *National Center for Biotechnology Information (NCBI)[Internet]*. 1988. URL: <https://www.ncbi.nlm.nih.gov/>.
- [82] Eric W Sayers et al. “Database resources of the national center for biotechnology information”. In: *Nucleic acids research* 50.D1 (2022), p. D20. DOI: 10.1093/nar/gkab1112.
- [83] Nuala A. O’Leary et al. “Exploring and retrieving sequence and metadata for species across the tree of life with NCBI Datasets”. In: *Scientific Data* 11.1 (July 2024). ISSN: 2052-4463. DOI: 10.1038/s41597-024-03571-y.
- [84] National Center for Biotechnology Information. *NCBI Datasets command-line tools*. URL: <https://www.ncbi.nlm.nih.gov/datasets/docs/v2/download-and-install/>.
- [85] The Genome Reference Consortium. *Fourteenth patch release for the GRCh38 reference assembly*. May 2022.
- [86] Kazutaka Katoh et al. “MAFFT: a novel method for rapid multiple sequence alignment based on fast Fourier transform”. In: *Nucleic acids research* 30.14 (2002), pp. 3059–3066. DOI: 10.1093/nar/gkf436.

- [87] John Rozewicki et al. “MAFFT-DASH: integrated protein sequence and structural alignment”. In: *Nucleic acids research* 47.W1 (2019), W5–W10. DOI: 10.1093/nar/gkz342.
- [88] Subha Kalyanamoothy et al. “ModelFinder: fast model selection for accurate phylogenetic estimates”. In: *Nature methods* 14.6 (2017), pp. 587–589. DOI: 10.1038/nmeth.4285.
- [89] Bui Quang Minh et al. “IQ-TREE 2: new models and efficient methods for phylogenetic inference in the genomic era”. In: *Molecular biology and evolution* 37.5 (2020), pp. 1530–1534. DOI: 10.1093/molbev/msaa015.
- [90] Diep Thi Hoang et al. “UFBoot2: improving the ultrafast bootstrap approximation”. In: *Molecular biology and evolution* 35.2 (2018), pp. 518–522. DOI: 10.1093/molbev/msx281.
- [91] Fernando Domingues Kümmel Tria, Giddy Landan, and Tal Dagan. “Phylogenetic rooting using minimal ancestor deviation”. In: *Nature ecology & evolution* 1.7 (2017), p. 0193. DOI: 10.1038/s41559-017-0193.
- [92] Guangchuan Yu. “Using ggtree to Visualize Data on Tree-Like Structures”. In: *Current Protocols in Bioinformatics* 69.1 (2020), e96. DOI: 10.1002/cpbi.96.
- [93] Stefan Götz et al. “High-throughput functional annotation and data mining with the Blast2GO suite”. In: *Nucleic acids research* 36.10 (2008), pp. 3420–3435. DOI: 10.1093/nar/gkn176.
- [94] Ross Overbeek et al. “The SEED and the Rapid Annotation of microbial genomes using Subsystems Technology (RAST)”. In: *Nucleic acids research* 42.D1 (2014), pp. D206–D214. DOI: 10.1093/nar/gkt1226.
- [95] Robert D Finn et al. “Pfam: the protein families database”. In: *Nucleic acids research* 42.D1 (2014), pp. D222–D230. DOI: 10.1093/nar/gkt1223.
- [96] Philip Jones et al. “InterProScan 5: genome-scale protein function classification”. In: *Bioinformatics* 30.9 (2014), pp. 1236–1240. DOI: 10.1093/bioinformatics/btu031.
- [97] Roman L Tatusov, Eugene V Koonin, and David J Lipman. “A genomic perspective on protein families”. In: *Science* 278.5338 (1997), pp. 631–637. DOI: 10.1126/science.278.5338.631.
- [98] Benjamin Buchfink, Klaus Reuter, and Hajk-Georg Drost. “Sensitive protein alignments at tree-of-life scale using DIAMOND”. In: *Nature methods* 18.4 (2021), pp. 366–368. DOI: 10.1038/s41592-021-01101-x.
- [99] Bernhard Haubold. *gin: Analyze Genome Intervals*. <https://github.com/EvolBioInf/gin>.
- [100] Mark S Springer et al. “Molecules consolidate the placental mammal tree”. In: *Trends in ecology & evolution* 19.8 (2004), pp. 430–438. DOI: 10.1016/s0169-5347(04)00142-9.
- [101] William J Murphy et al. “Using genomic data to unravel the root of the placental mammal phylogeny”. In: *Genome research* 17.4 (2007), pp. 413–421. DOI: 10.1101/gr.5918807.
- [102] Nicole M Foley et al. “A genomic timescale for placental mammal evolution”. In: *Science* 380.6643 (2023), eabl8189. DOI: 10.1126/science.abl8189.
- [103] Ulfur Arnason et al. “Mammalian mitogenomic relationships and the root of the eutherian tree”. In: *Proceedings of the National Academy of Sciences* 99.12 (2002), pp. 8151–8156. DOI: 10.1073/pnas.102164299.
- [104] Peter WH Holland et al. “Gene duplications and the origins of vertebrate development”. In: *Development* 1994.Supplement (1994), pp. 125–133. DOI: 10.1242/dev.1994.supplement.125.

- [105] Edward Málaga-Trillo and Axel Meyer. “Genome duplications and accelerated evolution of Hox genes and cluster architecture in teleost fishes”. In: *American Zoologist* 41.3 (2001), pp. 676–686. DOI: 10.1093/icb/41.3.676.
- [106] Aylwyn Scally. “The mutation rate in human evolution and demographic inference”. In: *Current opinion in genetics & development* 41 (2016), pp. 36–43. DOI: 10.1016/j.gde.2016.07.008.
- [107] Heui-Soo Kim, Mee Sun Ock, and Hee-Jae Cha. “Interactions between human endogenous and exogenous retroviruses”. In: *Genes & Genomics* 39 (2017), pp. 923–927. DOI: 10.1007/s13258-017-0568-x.
- [108] Antonito T Panganiban and Howard M Temin. “The retrovirus pol gene encodes a product required for DNA integration: identification of a retrovirus int locus.” In: *Proceedings of the National Academy of Sciences* 81.24 (1984), pp. 7885–7889. DOI: 10.1073/pnas.81.24.7885.
- [109] Xingyu Liao et al. “Repetitive DNA sequence detection and its role in the human genome”. In: *Communications Biology* 6.1 (2023), p. 954. DOI: 10.1038/s42003-023-05322-y.
- [110] J. A. Bedell, I. Korf, and W. Gish. “MaskerAid: a performance enhancement to RepeatMasker”. In: *Bioinformatics* 16 (2000), pp. 1040–1. DOI: 10.1093/bioinformatics/16.11.1040.
- [111] Maxime Lasseron. “Enigmatic teeth from the Jurassic–Cretaceous transition of Morocco: The latest known non-mammaliaform cynodonts (Synapsida, Cynodontia) from Africa?” In: *Comptes Rendus Palevol* 18.7 (2019), pp. 897–907. DOI: 10.1016/j.crpv.2019.05.002.
- [112] Laurence Bénit, Alexandra Calteau, and Thierry Heidmann. “Characterization of the low-copy HERV-Fc family: evidence for recent integrations in primates of elements with coding envelope genes”. In: *Virology* 312.1 (2003), pp. 159–168. DOI: 10.1016/s0042-6822(03)00163-6.
- [113] Guilherme S do Olival et al. “Genomic analysis of ERVWE2 locus in patients with multiple sclerosis: absence of genetic association but potential role of human endogenous retrovirus type W elements in molecular mimicry with myelin antigen”. In: *Frontiers in microbiology* 4 (2013), p. 172. DOI: 10.3389/fmicb.2013.00172.
- [114] eggNOG-mapper. *Too many GO annotations*. <https://github.com/eggnogdb/eggno-mapper/issues/407>. Sept. 2022.
- [115] Beatriz Vieira Mourato et al. “Marker discovery in the large”. In: *Bioinformatics Advances* 4.1 (2024). DOI: 10.1093/bioadv/vbae113.
- [116] John SantaLucia et al. “Appendix Q: Recommendations for Developing Molecular Assays for Microbial Pathogen Detection Using Modern In Silico Approaches.” In: *Journal of AOAC International* 103.4 (July 1, 2020), pp. 882–899. DOI: 10.1093/jaoacint/qsaa045.
- [117] Shaista Karim et al. “Development of the Automated Primer Design Workflow Uniqprimer and Diagnostic Primers for the Broad-Host-Range Plant Pathogen *Dickeya dianthicola*.” In: *Plant disease* 103.11 (Aug. 20, 2019), pp. 2893–2902. DOI: 10.1094/pdis-10-18-1819-re.
- [118] Stefan Kurtz et al. “Versatile and open software for comparing large genomes”. In: *Genome biology* 5 (2004), pp. 1–9. DOI: 10.1186/gb-2004-5-2-r12.
- [119] Andreas Untergasser et al. “Primer3—new capabilities and interfaces”. In: *Nucleic acids research* 40.15 (2012), e115–e115. DOI: 10.1093/nar/gks596.
- [120] Pavel Beran et al. “Utilization of a new hundred-genomes pipeline to design a rapid duplex LAMP detection assay for *Xanthomonas euvesicatoria* and *X. vesicatoria* in tomato”. In: *Plant Disease* 106 (2023), pp. 1822–1828. DOI: 10.1094/PDIS-05-22-1098-RE.

- [121] Zhenan Wang et al. “Development of loop-mediated isothermal amplification assays for the rapid and accurate diagnosis of *Exserohilum turcicum* for field applications”. In: *Plant Disease* (2024), doi.org/10.1094/PDIS-10-23-2101-SR. DOI: 10.1094/PDIS-10-23-2101-SR.
- [122] Johannes Fischer and Florian Kurpicz. “Dismantling divsufsort”. In: *arXiv preprint arXiv:1710.01896* (2017). DOI: 10.48550/arXiv.1710.01896.
- [123] Scott Federhen et al. *Meeting report: GenBank microbial genomic taxonomy workshop (12-13 May, 2015)*. 2016. DOI: 10.1186/s40793-016-0134-1.
- [124] Stacy Ciufo et al. “Using average nucleotide identity to improve taxonomic assignments in prokaryotic genomes at the NCBI”. In: *International journal of systematic and evolutionary microbiology* 68.7 (2018), pp. 2386–2392. DOI: 10.1099/ijsem.0.002809.
- [125] Conrad L Schoch et al. “NCBI Taxonomy: a comprehensive update on curation, resources and tools”. In: *Database* 2020 (2020), baaa062. DOI: 10.1093/database/baaa062.
- [126] Konstantinos T Konstantinidis and James M Tiedje. “Genomic insights that advance the species definition for prokaryotes”. In: *Proceedings of the National Academy of Sciences* 102.7 (2005), pp. 2567–2572. DOI: 10.1073/pnas.0409727102.
- [127] Michael Richter and Ramon Rosselló-Móra. “Shifting the genomic gold standard for the prokaryotic species definition”. In: *Proceedings of the National Academy of Sciences* 106.45 (2009), pp. 19126–19131. DOI: 10.1073/pnas.0906412106.
- [128] Chirag Jain et al. “High throughput ANI analysis of 90K prokaryotic genomes reveals clear species boundaries”. In: *Nature communications* 9.1 (2018), p. 5114. DOI: 10.1038/s41467-018-07641-9.
- [129] Scott Federhen et al. “Toward richer metadata for microbial sequences: replacing strain-level NCBI taxonomy taxids with BioProject, BioSample and Assembly records”. In: *Standards in genomic sciences* 9 (2014), pp. 1275–1277. DOI: 10.4056/sigs.4851102.
- [130] Fabian Klötzl and Bernhard Haubold. “Phylonium: Fast estimation of evolutionary distances from large samples of similar genomes”. In: *Bioinformatics* 36.7 (Apr. 2020), pp. 2040–2046. ISSN: 1460-2059. DOI: 10.1093/bioinformatics/btz903.
- [131] Bernhard Haubold. *stan: Simulate Targets and Neighbors*. <https://github.com/EvolBioInf/stan>.
- [132] Triinu Koressaar and Mado Remm. “Enhancements and modifications of primer design program Primer3”. In: *Bioinformatics* 23.10 (2007), pp. 1289–1291. DOI: 10.1093/bioinformatics/btm091.
- [133] Paul D Thomas et al. “PANTHER: Making genome-scale phylogenetics accessible to all”. In: *Protein Science* 31.1 (2022), pp. 8–22. DOI: 10.1002/pro.4218.
- [134] Seth Carbon et al. “AmiGO: online access to ontology and annotation data”. In: *Bioinformatics* 25.2 (2009), pp. 288–289. DOI: 10.1093/bioinformatics/btn615.
- [135] Alan Donovan and Brian Kernighan. *The Go Programming Language*. Ed. by Brian W. Kernighan. Addison-Wesley professional computing series. Includes index. - Description based on print version record. New York: Addison-Wesley, 2016. 11 pp. ISBN: 978-0-13-419044-0. DOI: 10.1007/978-3-662-09507-2_22.
- [136] Jinshui Zheng et al. “A taxonomic note on the genus *Lactobacillus*: Description of 23 novel genera, emended description of the genus *Lactobacillus* Beijerinck 1901, and union of *Lactobacillaceae* and *Leuconostocaceae*”. In: *International journal of systematic and evolutionary microbiology* 70.4 (2020), pp. 2782–2858. DOI: 10.1099/ijsem.0.004107.

- [137] *Off-Target Search Tool and Primer Design*. URL: https://www.flyrnai.org/RNAi_primer_design.html (visited on 05/27/2024).
- [138] Stephen P Diggle and Marvin Whiteley. “Microbe Profile: *Pseudomonas aeruginosa*: opportunistic pathogen and lab rat”. In: *Microbiology* 166.1 (2020), pp. 30–33. DOI: 10.1099/mic.0.000860.
- [139] Martin G Lamarche et al. “The phosphate regulon and bacterial virulence: a regulatory network connecting phosphate homeostasis and pathogenesis”. In: *FEMS microbiology reviews* 32.3 (2008), pp. 461–473. DOI: 10.1111/j.1574-6976.2008.00101.x.
- [140] Vanessa Jensen et al. “RhIR expression in *Pseudomonas aeruginosa* is modulated by the *Pseudomonas* quinolone signal via PhoB-dependent and-independent pathways”. In: *Journal of bacteriology* 188.24 (2006), pp. 8601–8606. DOI: 10.1128/JB.01378-06.
- [141] Samuel Mohammed Chekabab, Josée Harel, and Charles M Dozois. “Interplay between genetic regulation of phosphate homeostasis and bacterial virulence”. In: *Virulence* 5.8 (2014), pp. 786–793. DOI: 10.4161/viru.29307.
- [142] Manjeet Bains, Lucia Fernandez, and Robert EW Hancock. “Phosphate starvation promotes swarming motility and cytotoxicity of *Pseudomonas aeruginosa*”. In: *Applied and environmental microbiology* 78.18 (2012), pp. 6762–6768. DOI: 10.1128/aem.01015-12.
- [143] Lucas A Gallarato et al. “Exopolyphosphatase of *Pseudomonas aeruginosa* is essential for the production of virulence factors, and its expression is controlled by NtrC and PhoB acting at two interspaced promoters”. In: *Microbiology* 160.2 (2014), pp. 406–417. DOI: 10.1099/mic.0.074773-0.
- [144] Ana Sánchez-Jiménez, María A Llamas, and Francisco Javier Marcos-Torres. “Transcriptional Regulators Controlling Virulence in *Pseudomonas aeruginosa*”. In: *International Journal of Molecular Sciences* 24.15 (2023), p. 11895. DOI: 10.3390/ijms241511895.
- [145] Shawn Lewenza et al. “The *olsA* gene mediates the synthesis of an ornithine lipid in *Pseudomonas aeruginosa* during growth under phosphate-limiting conditions, but is not involved in antimicrobial peptide susceptibility”. In: *FEMS microbiology letters* 320.2 (2011), pp. 95–102. DOI: 10.1111/j.1574-6968.2011.02295.x.
- [146] Nicholas P Cianciotto. “Type II secretion: a protein secretion system for all seasons”. In: *Trends in microbiology* 13.12 (2005), pp. 581–588. DOI: 10.1016/j.tim.2005.09.005.
- [147] Jeevan Jyot et al. “Type II secretion system of *Pseudomonas aeruginosa*: in vivo evidence of a significant role in death due to lung infection”. In: *Journal of Infectious Diseases* 203.10 (2011), pp. 1369–1377. DOI: 10.1093/infdis/jir045.
- [148] Annette E LaBauve and Matthew J Wargo. “Detection of host-derived sphingosine by *Pseudomonas aeruginosa* is important for survival in the murine lung”. In: *PLoS pathogens* 10.1 (2014), e1003889. DOI: 10.1371/journal.ppat.1003889.
- [149] Yoko Satta, Jan Klein, and Naoyuki Takahata. “DNA archives and our nearest relative: the trichotomy problem revisited”. In: *Molecular Phylogenetics and Evolution* 14.2 (2000), pp. 259–275. DOI: 10.1006/mpev.2000.0704.
- [150] Feng-Chi Chen and Wen-Hsiung Li. “Genomic divergences between humans and other hominoids and the effective population size of the common ancestor of humans and chimpanzees”. In: *The American Journal of Human Genetics* 68.2 (2001), pp. 444–456. DOI: 10.1086/318206.

- [151] Wolfgang Enard and Svante Pääbo. “Comparative primate genomics”. In: *Annu. Rev. Genomics Hum. Genet.* 5.1 (2004), pp. 351–378. DOI: 10.1146/annurev.genom.5.061903.180040.
- [152] Ingo Ebersberger et al. “Mapping human genetic ancestry”. In: *Molecular biology and evolution* 24.10 (2007), pp. 2266–2276. DOI: 10.1093/molbev/msm156.
- [153] Yong Shao et al. “Phylogenomic analyses provide insights into primate evolution”. In: *Science* 380.6648 (2023), pp. 913–924. DOI: 10.1126/science.abn6919.
- [154] Taishan Hu et al. “Next-generation sequencing technologies: An overview”. In: *Human Immunology* 82.11 (2021), pp. 801–811. DOI: 10.1016/j.humimm.2021.02.012.
- [155] Karen H Miga and Ting Wang. “The need for a human pangenome reference sequence”. In: *Annual Review of Genomics and Human Genetics* 22.1 (2021), pp. 81–102. DOI: 10.1146/annurev-genom-120120-081921.
- [156] Wen-Wei Liao et al. “A draft human pangenome reference”. In: *Nature* 617.7960 (2023), pp. 312–324. DOI: 10.1530/ey.20.12.1.
- [157] Amy M Runck, Hideaki Moriyama, and Jay F Storz. “Evolution of duplicated β -globin genes and the structural basis of hemoglobin isoform differentiation in *Mus*”. In: *Molecular biology and evolution* 26.11 (2009), pp. 2521–2532. DOI: 10.1093/molbev/msp165.
- [158] Megan Phifer-Rixey, Bettina Harr, and Jody Hey. “Further resolution of the house mouse (*Mus musculus*) phylogeny by integration over isolation-with-migration histories”. In: *BMC Evolutionary Biology* 20.1 (2020), p. 120. ISSN: 1471-2148. DOI: 10.1186/s12862-020-01666-9.
- [159] Kazumichi Fujiwara et al. “Insights into *Mus musculus* population structure across Eurasia revealed by whole-genome analysis”. In: *Genome biology and evolution* 14.5 (2022), evac068. DOI: 10.1093/gbe/evac068.
- [160] Claire M Wade et al. “The mosaic structure of variation in the laboratory mouse genome”. In: *Nature* 420.6915 (2002), pp. 574–578. DOI: 10.1038/nature01252.
- [161] Thomas M Keane et al. “Mouse genomic variation and its effect on phenotypes and gene regulation”. In: *Nature* 477.7364 (2011), pp. 289–294. DOI: 10.1038/nature10413.
- [162] Hyuna Yang et al. “On the subspecific origin of the laboratory mouse”. In: *Nature genetics* 39.9 (2007), pp. 1100–1107. DOI: 10.1038/ng2087.
- [163] Bernhard Haubold. *biobox: Tools for Bioinformatics*. <https://github.com/EvolBioInf/biobox>.
- [164] Luis R Nassar et al. “The UCSC genome browser database: 2023 update”. In: *Nucleic acids research* 51.D1 (2023), pp. D1188–D1195. DOI: 10.1093/nar/gkac1072.
- [165] Nick Patterson et al. “Genetic evidence for complex speciation of humans and chimpanzees”. In: *Nature* 441.7097 (2006), pp. 1103–1108. DOI: 10.1038/nature04789.
- [166] Asger Hobolth et al. “Incomplete lineage sorting patterns among human, chimpanzee, and orangutan suggest recent orangutan speciation and widespread selection”. In: *Genome research* 21.3 (2011), pp. 349–356. DOI: 10.1101/gr.114751.110.
- [167] Sudhir Kumar et al. “TimeTree: a resource for timelines, timetrees, and divergence times”. In: *Molecular biology and evolution* 34.7 (2017), pp. 1812–1819. DOI: 10.1093/molbev/msx116.
- [168] Claire M Fraser et al. “The minimal gene complement of *Mycoplasma genitalium*”. In: *Science* 270.5235 (1995), pp. 397–404. DOI: 10.1126/science.270.5235.397.
- [169] Nicolas Altemose et al. “Complete genomic and epigenetic maps of human centromeres”. In: *Science* 376.6588 (2022), eabl4178. DOI: 10.1101/2021.07.12.452052.

- [170] Nicolas Altomose. “A classical revival: Human satellite DNAs enter the genomics era”. In: 128 (2022), pp. 2–14. DOI: 10.1016/j.semcd.2022.04.012.
- [171] Aylwyn Scally et al. “Insights into hominid evolution from the gorilla genome sequence”. In: *Nature* 483.7388 (2012), pp. 169–175. ISSN: 1476-4687. DOI: 10.1038/nature10842.
- [172] Jafar Salimian et al. “Evolution of immunoglobulins in vertebrates”. In: *IgY-Technology: Production and Application of Egg Yolk Antibodies: Basic Knowledge for a Successful Practice*. Springer, 2021, pp. 49–58. ISBN: 978-3-030-72686-7. DOI: 10.1007/978-3-030-72688-1_4.
- [173] Max D Cooper and Matthew N Alder. “The evolution of adaptive immune systems”. In: *Cell* 124.4 (2006), pp. 815–822. DOI: 10.1016/j.cell.2006.02.001.
- [174] David N Olivieri and Francisco Gambón Deza. “Immunoglobulin genes in primates”. In: *Molecular immunology* 101 (2018), pp. 353–363. DOI: 10.1016/j.molimm.2018.07.020.
- [175] Glennis A Logsdon et al. “The variation and evolution of complete human centromeres”. In: *Nature* 629.8010 (2024), pp. 136–145. DOI: 10.1038/s41586-024-07278-3.
- [176] Peter A Brennan and Keith M Kendrick. “Mammalian social odours: attraction and individual recognition”. In: *Philosophical Transactions of the Royal Society B: Biological Sciences* 361.1476 (2006), pp. 2061–2078. DOI: 10.1098/rstb.2006.1931.
- [177] John S Mattick et al. “Long non-coding RNAs: definitions, functions, challenges and recommendations”. In: *Nature reviews Molecular cell biology* 24.6 (2023), pp. 430–447. DOI: 10.1038/s41580-022-00566-8.
- [178] Chris P Ponting, Peter L Oliver, and Wolf Reik. “Evolution and functions of long noncoding RNAs”. In: *Cell* 136.4 (2009), pp. 629–641. DOI: 10.1016/j.cell.2009.02.006.
- [179] Xuefei Shi et al. “Long non-coding RNAs: a new frontier in the study of human diseases”. In: *Cancer letters* 339.2 (2013), pp. 159–166. DOI: 10.1016/j.canlet.2013.06.013.
- [180] Alexander F Palazzo and Eliza S Lee. “Non-coding RNA: what is functional and what is junk?” In: *Frontiers in genetics* 6 (2015), p. 2. DOI: 10.3389/fgene.2015.00002.
- [181] Thomas R Cech and Joan A Steitz. “The noncoding RNA revolution—trashing old rules to forge new ones”. In: *Cell* 157.1 (2014), pp. 77–94. DOI: 10.1016/j.cell.2014.03.008.
- [182] Xiang-Dong Fu. “Non-coding RNA: a new frontier in regulatory biology”. In: *National science review* 1.2 (2014), pp. 190–204. DOI: 10.1093/nsr/nwu008.
- [183] Luisa Statello et al. “Gene regulation by long non-coding RNAs and its biological functions”. In: *Nature reviews Molecular cell biology* 22.2 (2021), pp. 96–118. DOI: 10.1038/s41580-020-00315-9.
- [184] Shaohua Lu et al. “A hidden human proteome encoded by ‘non-coding’ genes”. In: *Nucleic acids research* 47.15 (2019), pp. 8111–8125. DOI: 10.1093/nar/gkz646.
- [185] Ci Chu, Jeffrey Quinn, and Howard Y Chang. “Chromatin isolation by RNA purification (ChIRP)”. In: *JoVE (Journal of Visualized Experiments)* 61 (2012), e3912. DOI: 10.3791/3912.
- [186] Pieter-Jan Volders et al. “LNCipedia 5: towards a reference set of human long non-coding RNAs”. In: *Nucleic acids research* 47.D1 (2019), pp. D135–D139. DOI: 10.1093/nar/gky1031.
- [187] RNAcentral Consortium. “RNAcentral: a hub of information for non-coding RNA sequences”. In: *Nucleic Acids Research* 47.D1 (2019), pp. D221–D229. DOI: 10.1093/nar/gky1034.
- [188] Xing Chen et al. “Computational models for lncRNA function prediction and functional similarity calculation”. In: *Briefings in functional genomics* 18.1 (2019), pp. 58–82. DOI: 10.1093/bfpg/ely031.

- [189] Christopher Klapproth et al. “Common features in lncRNA annotation and classification: a survey”. In: *Non-coding RNA* 7.4 (2021), p. 77. DOI: 10.3390/ncrna7040077.
- [190] Bernhard Haubold. “Alignment-free phylogenetics and population genetics”. In: *Briefings in Bioinformatics* 15.3 (2013), pp. 407–418. DOI: 10.1093/bib/bbt083.
- [191] Andrzej Zielezinski et al. “Alignment-free sequence comparison: Benefits, applications, and tools”. In: *Genome Biology* 18.1 (2017), p. 186. ISSN: 1474-760X. DOI: 10.1186/s13059-017-1319-7.
- [192] Paramvir Dehal and Jeffrey L Boore. “Two rounds of whole genome duplication in the ancestral vertebrate”. In: *PLoS biology* 3.10 (2005), e314. DOI: 10.1371/journal.pbio.0030314.
- [193] Zoonomia Consortium. “A comparative genomics multitool for scientific discovery and conservation”. In: *Nature* 587.7833 (2020), pp. 240–245. DOI: 10.1038/s41586-020-2876-6.
- [194] Genome 10K Community of Scientists. “Genome 10K: a proposal to obtain whole-genome sequence for 10 000 vertebrate species”. In: *Journal of Heredity* 100.6 (2009), pp. 659–674. DOI: 10.1093/jhered/esp086.
- [195] Lisa A Lansdon et al. “Factors affecting migration to GRCh38 in laboratories performing clinical next-generation sequencing”. In: *The Journal of Molecular Diagnostics* 23.5 (2021), pp. 651–657. DOI: 10.1016/j.jmoldx.2021.02.003.
- [196] Yupeng He et al. “Spatiotemporal DNA methylome dynamics of the developing mouse fetus”. In: *Nature* 583.7818 (2020), pp. 752–759. ISSN: 1476-4687. DOI: 10.1038/s41586-020-2119-x.
- [197] Katherine S Pollard et al. “An RNA gene expressed during cortical development evolved rapidly in humans”. In: *Nature* 443.7108 (2006), pp. 167–172. DOI: 10.1038/nature05113.
- [198] Katherine S Pollard et al. “Forces shaping the fastest evolving regions in the human genome”. In: *PLoS genetics* 2.10 (2006), e168. DOI: 10.1371/journal.pgen.0020168.eor.
- [199] John A Capra et al. “Many human accelerated regions are developmental enhancers”. In: *Philosophical Transactions of the Royal Society B: Biological Sciences* 368.1632 (2013), p. 20130025. DOI: 10.1098/rstb.2013.0025.
- [200] Dennis Kostka, Alisha K Holloway, and Katherine S Pollard. “Developmental loci harbor clusters of accelerated regions that evolved independently in ape lineages”. In: *Molecular biology and evolution* 35.8 (2018). We developed a model selection procedure to scan ape genomes in parallel for conserved elements similar to HARs, pp. 2034–2045. DOI: 10.1093/molbev/msy109.
- [201] Xupeng Bi et al. “Lineage-specific accelerated sequences underlying primate evolution”. In: *Science Advances* 9.22 (2023). lineage-specific accelerated regions in Apes, eadc9507. DOI: 10.1126/sciadv.adc9507.
- [202] Veil Mäkinen et al. *Genome-scale algorithm design: Bioinformatics in the era of high-throughput sequencing*. Ed. by Djamal Belazzougui, Fabio Cunial, and Alexandru I. Tomescu. Second edition. Title from publisher’s bibliographic system (viewed on 28 Sep 2023). Cambridge: Cambridge university press, 2015. 1444 pp. ISBN: 978-1-009-34123-3. DOI: 10.1017/9781009341257.
- [203] Fabio Cunial, Olger Denas, and Djamal Belazzougui. “Fast and compact matching statistics analytics”. In: *Bioinformatics* 38.7 (2022), pp. 1838–1845. DOI: 10.1093/bioinformatics/btac064.
- [204] Massimiliano Rossi et al. “MONI: a pangenomic index for finding maximal exact matches”. In: *Journal of Computational Biology* 29.2 (2022), pp. 169–187. DOI: 10.1089/cmb.2021.0290.

- [205] Heng Li. “Aligning sequence reads, clone sequences and assembly contigs with BWA-MEM”. In: *arXiv preprint arXiv:1303.3997* (2013). DOI: 10 . 48550 / arXiv . 1303 . 3997. eprint: 1303.3997.

Data availability

All supplementary material described in this thesis is available at the Open Research Data Repository for the Max Planck Society, EDMOND, under the following DOI: <https://doi.org/10.17617/3.RXJP5A>

List of Publications

1. B. Vieira Mourato et al. (2024). “Marker discovery in the large”. *Bioinformatics*, 1-7. <https://doi.org/10.1093/bioadv/vbae113>.
2. B. Vieira Mourato and B. Haubold. (2024). “Detection and annotation of unique regions in mammalian genomes”, *G3*. <https://doi.org/10.1093/g3journal/jkae257>.
3. B. Vieira Mourato and B. Haubold. “Fast Detection of Unique Genomic Regions”, *Computational and Structural Biotechnology Journal* (submitted).

Author contributions

Chapter 1: General introduction

Section 1.1 will be included in a mini review in Computational and Structural Biotechnology Journal:

- B. Vieira Mourato and B. Haubold. “Fast Detection of Unique Genomic Regions”, *Computational and Structural Biotechnology Journal* (in preparation).

BVM wrote the thesis chapter. BH edited it.

Chapter 2: Highly complex regions in mammalian genomes

Chapter 2 was published in G3:

- B. Vieira Mourato and B. Haubold. (2024). “Detection and annotation of unique regions in mammalian genomes”, *G3*. <https://doi.org/10.1093/g3journal/jkae257>.

BVM and BH were responsible for the conceptualization of the study. BVM carried out the formal analysis and visualization. BVM and BH developed the methodology. BVM was responsible for the writing of the thesis chapter. BVM and BH wrote the manuscript. BH edited the chapter.

Chapter 3: Large-scale marker discovery in prokaryotes

Chapter 3 was published in Bioinformatics Advances:

- B. Vieira Mourato et al. (2024). “Marker discovery in the large”. *Bioinformatics*, 1-7. <https://doi.org/10.1093/bioadv/vbae113>.

BVM and BH were responsible for the conceptualization of the study. BVM, BH, FK, IT and SV developed the methodology. BH developed the software while BVM, BH and

FK designed, tested and delivered the software. BVM, BH, IT and SD carried out the formal analysis and investigation. BVM was responsible for writing the thesis chapter while BVM and BH wrote the manuscript. BH edited the chapter.

Chapter 4: Finding lineage-specific regions in mammal

BVM and BH were responsible for the conceptualization of the study. BH developed the software while BVM and BH designed and tested the software. BVM carried out the formal analysis and visualization. BVM was responsible for writing the thesis chapter. BH edited the chapter.

Chapter 5: General discussion and future directions

BVM wrote the thesis chapter. BH edited it.

Authors given in alphabetical order (after BVM):

BVM: Beatriz Vieira Mourato, **BH:** Bernhard Haubold, **FK:** Fabian Klötzl, **IT:** Ivan Tsers, **SD:** Svenja Denker.

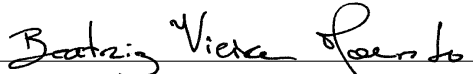
Statement

I hereby declare that:

- i) apart from my supervisor's guidance the content and design of this thesis entitled "Detection and functional annotation of unique genomic regions" is of my own work; The co-authors contributions are listed in the dedicated section;
- ii) this thesis, or parts thereof, has not been submitted to another examining body, nor has it been published or submitted for publication;
- iii) the thesis has been prepared and written with regard to the Rules of Good Scientific Practice of the German Research Foundation;
- iv) I have not attempted and failed to obtain a doctoral degree and no academic degree has ever been withdrawn;

Plön, 16th October 2024

Place, Date


Beatriz Vieira Mourato