Representation and integrated analysis of heterogeneous genomic datasets

Sveinung Gundersen

Thesis for the degree of Philosophiae Doctor (PhD)
Preface

It is now time to close this chapter of my life, and it’s about time, too! Many a late night have been used for hammering out code and testing functionality, or hammering out words that will hopefully convey a few interesting ideas to the reader. I was supposed to make this thesis as short as possible, but the wide nature of the research area lead to a much longer text than was initially planned. I hope you will find most of the material in these pages relevant.

It has been difficult to decide upon the level of prior knowledge that should be assumed from the readers of this PhD thesis. I have a background from computer science, but the research project also contains many elements from molecular biology and statistics. In order for the thesis to reflect the actual challenges met during the PhD project period, I have decided to write the thesis at a level understandable by a person with similar knowledge as I had when starting my PhD: only very basic knowledge of molecular biology and statistics is assumed.

It should also be noted that the order of the research papers has been chosen according to the most natural order of presentation. If ordered chronologically, Paper III should have come before Paper II.

At the start of the PhD period, few of the tools and methodologies that are most closely related to this thesis had yet been published. The most relevant tools were invented after the rise of next-generation sequencing, particularly after the development of ChIP-seq techniques. Because of this, I have chosen to present tools and methodologies in the introduction regardless of the time they were published. In the same vein, the state of the field of genomics is described at the time of writing this thesis, not the time the PhD project was started. The simple reason for this is the rapid development of the field. Much of what was state of the art in 2009 is already outdated today.

Let us hope that the following work will not be outdated in the very near future.
Acknowledgements

This PhD thesis was funded by Helse Sør-Øst and supervised by Prof. Eivind Hovig at the Department of Tumor Biology at the Norwegian Radium Hospital, Oslo University Hospital. I am very grateful for having received the opportunity to complete this PhD project.

As this work has been mostly informatics-based, I have been somewhat of a nomad, working partly in the research building at the Norwegian Radium Hospital, partly in the offices of Statistics for Innovation at the Norwegian Computing Center, partly in the sofa in the apartment of co-supervisor Geir Kjetil Sandve, and mostly in the offices of the Research Group for Biomedical Informatics (BMI) at the Department for Informatics, University of Oslo. I am especially grateful to Prof. Arnoldo Frigessi (SFI) and Prof. Ole Christian Lingjærde (BMI) for providing me office space.

I am very grateful to the collaborators in the HyperBrowser project for involving me and taking me seriously in the research discussions from the very beginnings, even if I did not have much competence in either biology or statistics. It was very inspiring to be a newcomer in such a competent research group. I would like to thank Professor Knut Liestøl for his calm wisdom and diplomatic attitude. I am grateful to Lars Holden, Managing Director of NCC, and Marit Holden, Chief Research Scientist of NCC, for their enthusiasm and friendliness, and to Egil Ferkingstad for his eagerness and ability to always produce quality work. A special thanks goes to Prof. Ingrid K. Glad and Prof. Arnoldo Frigessi for their enthusiasm, warmth, and inclusiveness, even in the heat of argument. At times when I have felt discouraged, it has been a real blessing just to meet any of them in the hallway, brightening my day considerably.

A special thanks goes to Morten Johansen, Kai Trengereid, Halfdan Rydebeck and Vegard Nygaard for the work they have poured into the Genomic HyperBrowser, and for the friendly and fun atmosphere they have all created in the work place. Also, thanks to Jonas Paulsen for needing ever more of the complex functionality in GTrack, giving me a major boost to my confidence! I am also very grateful to the rest of my collaborators and colleagues in the bioinformatics group of Prof. Eivind Hovig, as well as in the BMI group of the Department of Informatics, for the stimulating teamwork and the good-natured and lively work environment. Furthermore, I specifically want to thank Matúš Kalaš for intelligent feedback leading to major improvements to the GTrack format. And last, thanks to all the master students who have extended or used the Genomic HyperBrowser. It is motivating to see that the system is evolving.

My supervisors deserve an exceptional thanks. First, I am humbled by
the energy, clear-sightedness, enthusiasm, and sheer work capacity of my co-supervisor, now Associate Professor, Geir Kjetil Sandve. Also, thanks to you and Christin for the many invitations to come to visit your home, both for work and leisure. It is a great joy to be able to work with a friend, and it has been especially satisfying to experience the creative dynamics that has emerged from the differences in our personalities. I am also grateful for the constructive way you have dealt with my frustrations.

Second, but not least, a deep-felt thanks to my supervisor, professor Eivind Hovig, for believing in me and giving me this opportunity. Your attitude and humor is inspiring, as is your deep commitment and concern for the people that work in your group. Your insistence on holding me accountable for my promises has helped me to better understand how I function in stressful situations and made me realize that “the best may be the enemy of the good”. I am very grateful for the combination of warmth and seriousness you have showed me.

To my family: I want to thank my parents for always believing in me, no matter what, and for always being there for me. Your love has given me a solid base in life. I also want to thank my brother and sister, Torleif and Solfrid, for their love and support. Lastly, I want to thank the family of Guro for providing me with a “home away from home”.

The greatest thanks goes to the love of my life. Thank you, Guro, for loving me so immensely and for always being there by my side! If it weren’t for you, none of this would matter. Thank you for persevering with my unstable work hours! I am deeply humbled by your wisdom, your concern for others, and your beauty.

Sveinung Gundersen
Contents

Preface iii
Acknowledgements iv

Contents x

Glossary and List of abbreviations xiv

List of Tables xv

List of Figures xvii

List of Papers xix

1 Introduction 1
1.1 Molecular biology 3
1.1.1 Challenges to the central dogma 3
1.1.2 50 years of technological innovation 6
1.1.2.1 Hybridization 6
1.1.2.2 Blotting 8
1.1.2.3 DNA cloning and PCR 9
   DNA cloning 9
   PCR 9
1.1.2.4 Microarray 10
   Dot blot 10
   Microarray technology 10
   Applications of microarrays 11
   ChIP-chip 11
1.1.2.5 Sequencing 12
   Sanger sequencing 12
   Automation 13
   EST sequencing 13
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Next-generation sequencing</td>
<td>14</td>
</tr>
<tr>
<td>Sequencing-by-synthesis</td>
<td>14</td>
</tr>
<tr>
<td>Sequencing-by-ligation</td>
<td>15</td>
</tr>
<tr>
<td>Semiconductor-based sequencing</td>
<td>15</td>
</tr>
<tr>
<td>Third-generation sequencing</td>
<td>15</td>
</tr>
<tr>
<td>RNA-seq</td>
<td>16</td>
</tr>
<tr>
<td>ChIP-seq</td>
<td>17</td>
</tr>
<tr>
<td>DNase I hypersensitive sites</td>
<td>17</td>
</tr>
<tr>
<td>Hi-C and ChIA-PET</td>
<td>18</td>
</tr>
<tr>
<td>1.1.3 Implications of NGS on data analysis</td>
<td>18</td>
</tr>
<tr>
<td>1.2 Data representation</td>
<td>21</td>
</tr>
<tr>
<td>1.2.1 Fundamental aspects of data representation</td>
<td>21</td>
</tr>
<tr>
<td>1.2.2 Data representation in bioinformatics</td>
<td>22</td>
</tr>
<tr>
<td>1.2.2.1 Network-based representation and analysis</td>
<td>22</td>
</tr>
<tr>
<td>Basic entities</td>
<td>22</td>
</tr>
<tr>
<td>Network-based data integration</td>
<td>23</td>
</tr>
<tr>
<td>1.2.2.2 The importance of non-coding DNA</td>
<td>23</td>
</tr>
<tr>
<td>The ENCODE project</td>
<td>24</td>
</tr>
<tr>
<td>Redefining genes</td>
<td>25</td>
</tr>
<tr>
<td>Variants in non-coding loci</td>
<td>26</td>
</tr>
<tr>
<td>Function of non-coding DNA</td>
<td>26</td>
</tr>
<tr>
<td>1.2.2.3 Dataset types based on sequence-relative coordinates</td>
<td>26</td>
</tr>
<tr>
<td>Genome annotation</td>
<td>27</td>
</tr>
<tr>
<td>Repeating elements</td>
<td>27</td>
</tr>
<tr>
<td>Non-coding RNAs</td>
<td>28</td>
</tr>
<tr>
<td>Gene expression</td>
<td>28</td>
</tr>
<tr>
<td>Protein-DNA interactions</td>
<td>29</td>
</tr>
<tr>
<td>Epigenetic markers</td>
<td>29</td>
</tr>
<tr>
<td>Cis-regulatory modules</td>
<td>30</td>
</tr>
<tr>
<td>Other chromatin-based datasets</td>
<td>31</td>
</tr>
<tr>
<td>Variation</td>
<td>32</td>
</tr>
<tr>
<td>Virus integration sites</td>
<td>33</td>
</tr>
<tr>
<td>Sequence-derived datasets</td>
<td>33</td>
</tr>
<tr>
<td>1.2.2.4 Track model of genomic datasets</td>
<td>34</td>
</tr>
<tr>
<td>Representing tracks at the abstract level</td>
<td>35</td>
</tr>
<tr>
<td>Representing tracks at the technical level</td>
<td>36</td>
</tr>
<tr>
<td>Limitations of track representation</td>
<td>37</td>
</tr>
<tr>
<td>1.3 Data analysis</td>
<td>39</td>
</tr>
<tr>
<td>1.3.1 Exploratory data analysis</td>
<td>40</td>
</tr>
<tr>
<td>1.3.1.1 Genome browsers</td>
<td>41</td>
</tr>
</tbody>
</table>
UCSC, ENSEMBL, and GBrowse .................. 41
Savant, IGV, and other recent developments 42
Circos ........................................... 42
Galaxy Track Browser ........................... 43
Challenges with genome browser-based analysis 43
1.3.1.2 Pattern discovery ....................... 44
Heatmap approaches with seqMiner and Cis-
 trome ........................................... 44
ChromaSig ........................................ 44
Segmentation approaches ........................ 45
1.3.1.3 Track comparison ........................ 46
Correlative approaches with seqMiner and Cis-
trome ........................................... 46
Biplots ............................................. 46
BEDTools, MULTOVL, and Segtools .......... 47
Mutation rate co-variation ....................... 48
1.3.1.4 Functional associations ................. 49
DAVID, GREAT, and PRISM .................. 49
Cistrome (CEAS) and EpiExplorer ............ 49
1.3.2 Confirmatory data analysis ............... 50
1.3.2.1 Simple overlap tests ..................... 53
1.3.2.2 On the border between exploratory and con-
firmatory data analysis ....................... 54
GenomeRunner ................................... 54
EpiGRAPH ....................................... 54
EpiCenter ........................................ 55
1.3.2.3 Confirmatory tools and techniques ...... 55
GenomtriCorr ................................. 55
Subsampling methods of ENCODE .......... 56
1.3.2.4 Finding transcription factor binding co-occur-
rence ........................................... 57
1.4 Toward integrated data analysis ............ 59

2 Aims of the study .......................... 63

3 Summary of the papers ....................... 65
3.1 Paper I ........................................ 65
3.2 Paper II ...................................... 68
3.3 Paper III ..................................... 71
4 Discussion

4.1 Categorizing tracks and data analysis methods by track type  74
4.2 Confirmatory or exploratory data analysis? .......................... 80
4.3 Biological relevance ...................................................... 83
4.4 User experience ............................................................ 86
4.5 Implementational issues ................................................... 90
4.6 Comparison with existing methodologies .............................. 93
4.7 Future perspectives ......................................................... 95

5 Conclusions ................................. 99

References ............................................ 100

Errata ................................................. 119

Paper I ............................................... 121

Paper II ............................................... 135

Paper III ............................................. 155

A Analyses and tools in the Genomic HyperBrowser 167

B GTrack 1.0 specification .............................................. 175

C Matrix-based counting algorithm .................................... 211
Glossary and List of abbreviations

3C chromosome conformation capture.

bp base pair, e.g. a pair of complementary nucleotides.

cDNA complementary DNA.

CGH comparative genomic hybridization.

ChIA-PET chromatin interaction analysis by paired-end tag sequencing.

ChIP chromatin immunoprecipitation.

ChIP-chip chromatin immunoprecipitation followed by microarray analysis.

ChIP-seq chromatin immunoprecipitation followed by high-throughput sequencing.

chromatin the DNA molecules together with proteins that help to package the DNA inside the nucleus of a cell.

CNV copy number variation (CNV) reflects abnormal number of copies of parts of the DNA.

codon bias differences in the frequency of occurrence of codons coding for the same amino acid.

CpG positions along the DNA where cytosine is immediately followed by guanine.

CpG islands regions of the genome with elevated GC content, after some variation of region length and GC content thresholds.
CRM  cis-regulatory module.

DNA  deoxyribonucleic acid (DNA) is a double-stranded helical macromolecule consisting of complementary nucleotides, with adenine (A) pairing with thymine (T) and guanine (G) pairing with cytosine (C).

DNA methylation  cytosine modification by methylation at the C5 position.

enzyme  a type of protein that acts as a catalyst in specific biochemical reactions.

epigenetic modifications  epigenetic modifications of the DNA, i.e. modifications that do not involve a change in nucleotide sequence, such as histone modifications and DNA methylation.

epigenomics  the study of epigenomes.

EST  expressed sequence tag.

eukaryote  organisms with cells containing a nucleus (most multicellular organisms).

exome  the entirety of an organisms exons.

exon  coding region of a gene.

functional genomics  the study of the dynamic aspects of the genes, i.e. gene regulation and interactions.

GC content  the number of guanines and cytosines in a sliding window of some length over the sequence.

genome  the entirety of an organism’s hereditary information.

genomics  the study of the whole genome of an organism.

GSEA  Gene Set Enrichment Analysis.

GWAS  genome-wide association study.

histone  highly alkaline protein that helps organize DNA into nucleosomes. the four ‘core’ histones (H2A, H2B, H3 and H4) all share a long N-terminal polypeptide chain that can be modified by chemical alterations.
**in silico** experiments performed with a computer or via computer simulation.

**in vitro** experiments on isolated components of an organism.

**in vivo** experiments in whole, living organisms.

**intron** non-coding region of a gene.

**kbp** thousand base pairs.

**LCR** locus control region.

**LINE** long interspersed element.

**IncRNA** long non-coding RNA.

**LTR** long terminal repeat.

**MAF** minor allele frequency.

**Mbp** million base pairs.

**metagenome** sample of genetic material from environmental isolates, such as soil or water.

**miRNA** microRNA.

**mRNA** messenger RNA.

**NGS** next-generation sequencing.

**nucleosome** protein complexes of typically four histone protein dimers, around which eukaryotic DNA is wound.

**nucleotide** molecules forming the building blocks of nucleic acids (DNAs and RNAs). One of adenine (A), thymine (T), uracil (U), guanine (G), cytosine (C).

**oligonucleotide** small sequence of single-stranded DNA.

**ORF** open reading frame.

**paired-end sequencing** sequencing from both sides of the same sequence.

**PCR** Polymerase Chain Reaction.
**phenotype**  the set of an organism's measurable traits.

**piRNA**  piwi-interacting RNA.

**prokaryote**  organisms with cells without a nucleus, usually single-celled.

**protein**  polypeptide chain of amino acids, folded to a three-dimensional structure. Proteins perform a vast array of functions within an organism.

**proteome**  the entire set of proteins expressed by a genome, cell, tissue or organism at a certain time.

**proteomics**  the large-scale study of the structure and function of proteins.

**RNA**  ribonucleic acid (RNA) is a mostly single-stranded macromolecule consisting of the nucleotides adenine (A), uracil (U), guanine (G), and cytosine (C).

**RNA-seq**  high-throughput of RNA molecules.

**rRNA**  ribosomal RNA.

**SINE**  short interspersed element.

**siRNA**  small interfering RNA.

**snoRNA**  small nucleolar RNA.

**SNP**  single nucleotide polymorphism.

**snRNA**  small nuclear RNA.

**TF**  transcription factor.

**TFBS**  transcription factor binding site.

**transcriptome**  the set of all RNA molecules in one or a population of cells.

**transcriptomics**  the study of all RNA molecules transcribed from DNA.

**TRE**  transcriptional regulatory element.

**tRNA**  transfer RNA.

**TSS**  transcription start site.

**UTR**  untranslated region.

**YAC**  yeast artificial chromosome.
# List of Tables

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.1</td>
<td>Overview of typical tracks, categorized by track type</td>
<td>76</td>
</tr>
<tr>
<td>4.2</td>
<td>Overview of tools and methodologies for data analysis, categorized by track type</td>
<td>78</td>
</tr>
<tr>
<td>A.1</td>
<td>Selected descriptive statistics and hypothesis tests available through the Genomic HyperBrowser</td>
<td>168</td>
</tr>
<tr>
<td>A.2</td>
<td>Tools for statistical, visual and specialized analyses of genomic tracks in the Genomic HyperBrowser</td>
<td>170</td>
</tr>
<tr>
<td>A.3</td>
<td>Tools for tracks extraction, customization, generation, and formatting in the Genomic HyperBrowser</td>
<td>172</td>
</tr>
</tbody>
</table>
List of Figures

1.1 The explosion of ‘-ome’ and ‘-omics’ terms in science ........... 2
1.2 Transcription of DNA into mRNA and translation of mRNA into proteins ........................................ 4
1.3 Alternative splicing and transcription regulation ............. 5
1.4 Chromatin and epigenetic modifications ......................... 7
1.5 Contribution of different factors to the overall cost of sequencing projects across time ......................... 19
1.6 Screen shot of the UCSC Genome Browser .................... 34
1.7 Pattern discovery analysis of epigenetic markers using Cistrome 45
1.8 Comparative analysis of SOM-maps of global epigenetic profiles 47
1.9 Comparison of histone modification tracks using Cistrome 48
1.10 Functional associations with epigenetic markers using EpiExplorer ........................................ 51
1.11 Illustration of hypothesis test of overlap ....................... 53
1.12 Estimation of the effect of transcriptional synergy by a multivariate Hawkes model .......................... 60

3.1 Four-dimensional matrix mapping the relations of the fifteen track types ........................................ 68
3.2 Illustration of the geometric properties of the fifteen track types 70

A.1 Schematic overview of tool categories available at the Genomic HyperBrowser server. .......................... 174
List of Papers


Chapter 1

Introduction

The data explosion caused by the technological revolution in molecular biology has given rise to a range of scientific subfields, many of which have been named with terms ending with ‘-omics’. The use of the ‘-omics’ suffix signifies a focus on the use of large-scale information to understand the completeness, in some way or another, of the subject under study, often a particular population of molecules in the cell. Genomics was the first of these terms, first coined by Dr. Tom Roderick in 1986, when deciding on the name of a newly proposed journal [1]. The journal Genomics started publishing the following year. The name was based on the term genome, first used in 1920 to refer to the set of all chromosomes. Genomics is thus the study of the whole genome of an organism, i.e. all contents of the DNA of an organism, thus including all genes, but also all intergenic regions. Proteomics is a more recent term, first proposed in 1995, denoting the large-scale study of the structure and function of proteins. While transcriptomics has been defined as the study of “the complete complement of mRNA molecules generated by a cell or population of cells” [2], it is also used for the study of all RNA molecules transcribed from DNA, not only mRNAs. The term functional genomics has later been coined, denoting the study of the dynamic aspects of the genes, i.e. gene regulation and interactions. Lately, the importance of epigenetic modifications of the DNA, i.e. modifications that do not involve a change in nucleotide sequence, has been revealed [3]. Thus, the term epigenomics was created to denote the study of these modifications and their interactions with the rest of the genome. The rate of new findings in molecular biology have given rise to not just these ‘-omics’ terms, but to a plethora of them, steadily undermining their usefulness [4, 5]. Figure 1.1 shows an overview

---

1 As a side-note, it should be noted that the author of this thesis is not wholly innocent in these matters, as is evident with the terms terms inferential genomics and differential disease
of the growth of new ‘-omics’ terms.

Figure 1.1: The explosion of ‘-ome’ and ‘-omics’ terms in science, based on data from the Medline database. Republished with permission of Dow Jones & Company, from [6]; permission conveyed through Copyright Clearance Center, Inc.

The rise of “-ome” and “-omics” terms signals a significant shift from the study of individual entities, such as genes and molecules, to the study of their interrelations, a move from the particularities of each single instrument to the “symphony of the cell” that emerges from their combined contributions. This shift follows the radical improvements of experimental techniques of molecular biology that the field has witnessed the last 50 years or so. Together with the technical advances has followed a gradual realization that the complexity of interrelations in, and across, cells are crucial for understanding the basic foundations of life. Ever-increasing amounts of data of heterogeneous nature have been produced in order to unravel the complexities of molecular biology, much of which has been categorized and is publicly available from different databases. The search to make sense of such data has necessitated a increased focus on the field of bioinformatics/computational biology, a cross-disciplinary field bringing together methodology and knowledge from statistics, informatics and biology. Specific biological investigations based on data stemming from particular technolo-

\textsuperscript{2}In this thesis, the terms “bioinformatics” and “computational biology” will be used synonymously and broadly, even though some draw a distinction between them. In the opinion of the latter, bioinformatics should be used for typical engineering aspects, such as tool or database development, while computational biology should be used for the application of computational methodology on biological investigations, \textit{i.e.} for doing science [7].
gies have traditionally driven the development of methodology. However, the issue of integrating data sets from diverse “-omics” fields has seen much focus [8]. In order to carry out integrative analyses of heterogeneous genomic data, interoperable representation of data is needed, i.e. basic models of data structure that allow computerized storage and analysis.

This thesis focuses both on the representational aspect on data integration, in addition to both explorative and confirmatory data analysis. This introductory chapter will present a brief overview of major technological innovations of molecular biology from the last 50 years, focusing specifically on sequencing technologies and their implications on, and the importance of, data representation and analysis. Next, basic models of data representation are discussed as a background to argue the advantages of using genomic tracks, with coordinates relative to some reference genome, as the basic representational model for this work. In relation of this, an overview of common types of genomic data sets are presented. After this, types of relevant data analysis tools and methodologies are discussed, with a focus on full-genome analysis of data in the form of genomic tracks.

After the introduction, the aims of the thesis are presented, along with summaries of the included papers. The contributions to data representation presented in this thesis are then used to catalogue existing models, methodologies, and tools. Furthermore, the advantages and limitations of the contributions of the thesis papers are discussed, and future perspectives are sketched.

### 1.1 Molecular biology

#### 1.1.1 Challenges to the central dogma

The central dogma of molecular biology was formulated by Francis Crick in 1958, and restated in a Nature paper from 1970 [13]. The dogma establishes the transfer of information between the three basic classes of cellular molecules: DNAs, RNAs and proteins. Of the nine (3x3) possible transfers, four seemed to exist: DNA -> DNA (i.e. DNA replication), DNA -> RNA (transcription, illustrated in Figure 1.2A³), RNA -> protein (translation, illustrated in Figure 1.2B), and RNA -> RNA (replication of RNA viruses). All transfers from proteins, however, were postulated to be very unlikely to occur. According to the central dogma, the protein is thus considered the end

³This figure, as the rest of the thesis, is based upon the biology of eukaryotes, i.e. organisms with cells containing a nucleus. Eukaryotes include most multi-cellular organisms, such as mammals and plants. Prokaryote (bacteria and archae) have fundamentally different cellular biology, but much of the discussion in this thesis will still be relevant.
Figure 1.2: A) Transcription: in the nucleus of the cell, the deoxyribonucleic acid (DNA) sequence is read by a RNA polymerase enzyme which produces a complementary, antiparallel ribonucleic acid (RNA) transcript, including the nucleotide uracil (U) instead of the thymine (T) of the corresponding DNA sequence. The RNA strand is further processed into a messenger RNA (mRNA) molecule which then exits to the cytoplasm. Figure reprinted from [9] (public domain). B) Translation: ribosomes (made up of ribosomal RNA and proteins) facilitate translation by allowing the docking of transfer RNA (tRNA) molecules. Each type of tRNA molecule is bound to a corresponding type of amino acid and further contains three nucleotides (an anticodon) which recognizes a particular set of three nucleotides (a codon) of the mRNA sequence. As each tRNA docks to the mRNA, the attached amino acid is added to a polypeptide chain, which is further spontaneously folded into a protein. Thus, the mRNA sequence is uniquely translated into an amino acid sequence by the genetic code maintained by the different types of tRNA. Figure reprinted from [10] (public domain).
Figure 1.3: A) Alternative splicing: the direct transcription of a DNA sequence results in a precursor mRNA molecule, which is further modified by several post-transcriptional modifications, including splicing. In the splicing process, the introns, i.e. non-coding parts of genes, are removed, leaving only the exon sequences. Splicing can be carried out in alternative ways, of which ‘exon skipping’ (illustrated) is the most common. The same gene can thus result in different transcripts, and in the end, different protein isoforms. Figure reprinted from [11] (public domain). B) Transcriptional regulation: transcription factors (TFs) bind to transcription factor binding sites (TFBS) both proximally and distally to genes (which may be clustered into cis-regulatory modules). A transcription initiation complex is bound to the promoter region in front of genes, is stabilized by activators (TFs) and co-activator proteins, helping to position the RNA polymerase II protein over the transcription start site (TSS) of the gene to initiate transcription. Distal TFBS are believed to be positioned close to the transcription initiation complex by the looping of DNA in 3D. Figure adapted by permission from Macmillan Publishers Ltd: Nature Review Genetics ([12]), copyright 2004.
product of information transferral.

Over 50 years of research have shown the inadequacy of the central dogma to explain basic intracellular processes. Even though the basic model of information transferral from DNA to RNA to proteins still is central, current models are far more complex. One challenge came with the discovery of reverse transcription of RNA into DNA by RNA viruses. However, Crick still found this discovery to fit with the dogma [13]. Another challenge came with the discovery of prions, i.e. proteins that are able to change the conformation of other proteins. Prions are found to be the cause of spreading diseases such as scrapie in sheep, and is thus an example of protein-protein information transferral. A greater challenge came with the discovery of alternative splicing of exons in 1977 (see Figure 1.3A). Gene transcripts (i.e. mRNAs) are made up of shorter exon sequences spliced, or fused, together, removing the intron sequences lying in-between. A single gene may usually lead to different alternative transcripts, each representing a particular splicing of exons. A consequence of the fact that the same DNA sequence can give rise to several mRNA isoforms is that some form of information transferral apart from what is stored in the DNA sequence is needed. Both protein -> DNA (transcription factor binding [14], see Figure 1.3B) and protein -> RNA (alternative splicing regulation [15]) information transfer have been found to be important, directly contradicting the central dogma [16]. In addition to this, a host of new types of information encoding has been revealed, such as DNA methylation, histone modifications, and chromatin conformation, as illustrated in Figure 1.4.

A set of technological breakthroughs that has occurred in the last 50 years have given rise to increasingly detailed data and models on intra-cellular mechanisms, constantly challenging and expanding earlier hypotheses, as exemplified above. The following section will present some of the most important technological advances in molecular biology: blotting, polymerase chain reaction (PCR), microarrays, and sequencing technologies, including the recent surge of next-generation sequencing techniques.

1.1.2 50 years of technological innovation

1.1.2.1 Hybridization

DNA has a remarkable property that constitutes the basis of many technologies of molecular biology. As first described by Francis Crick and James D. Watson in 1953, DNA consist of two strands of complementary nucleic acids, with Adenine (A) pairing with Thymine (T) and Guanine (G) pairing with Cytosine (C) [18]. The double-stranded nature of DNA allows the strands to be
Histone modification
A combination of different molecules can attach to the ‘tails’ of proteins called histones. These alter the activity of the DNA wrapped around them.

DNA methylation
Methyl marks added to certain DNA bases repress gene activity.

Figure 1.4: Chromatin is the DNA molecules together with proteins that help packaging the DNA inside the nucleus of a cell. The primary proteins are histones. Four pairs of histones make up a nucleosome complex, around which the DNA is wound at regular intervals. Two of the most common epigenetic modifications are illustrated: histone modifications and DNA methylation. Figure adapted by permission from Macmillan Publishers Ltd: Nature ([17]), copyright 2006.
denatured, i.e. separated by heat or treatment with alkali, and then renatured, or hybridized, by the reverse process, reforming the double strand. The process was first described by Marmur and Doty in 1961 [19]. It was quickly established that the stability of the reformed DNA depended on the extent of the complementarity of the two sequences. Analytical methods for analyzing relationships between nucleic acid sequences were thus developed, based on hybridization of probe sequences to denatured DNA sequences. [20]

1.1.2.2 Blotting

Although primitive by modern standards, the technique of blotting constitutes the historical backdrop of modern techniques such as microarray and sequencing, and is also a surprisingly versatile technique in itself. The blotting technique was first published in 1975 by Sir Edwin Southern. The basic variant is commonly called the Southern blot, named after the inventor. Other main variants were developed within a few years, namely the Northern blot and Western blot. All blotting techniques are dependent on gel electrophoresis, introduced in the 1950s, which is used to separate molecules according to their size (or length), or other physical features of their molecules (e.g. isoelectric focusing). Molecules are moved through a gel by the application of an electric field and are thus sorted according to the rate of traversal, which is proportional to the size of the molecule [21]. Prior to Southern blotting, DNA fragments are produced by enzymatic digestion of chromosomes with restriction endonucleases. These enzymes cleave DNA only at specific cleavage sites determined by the nucleic acids sequence, so that the distribution of fragments is kept (relatively) constant. The DNA fragments are separated by gel electrophoresis, before the gel is covered by a sheet of nitrocellulose, absorbing the DNA. The DNA is then mildly denatured in an alkaline solution to make sure that the DNA is single stranded. Cloned or sequenced single-stranded DNA, the probe DNA, is then radiolabeled and incubated with the nitrocellulose sheet so that it hybridizes with the chromosomal DNA fragments. Lastly, the sheet is dried and covered with a photographic or x-ray film, so that the probe DNA can be localized. The basic application of Southern blotting is to identify whether a particular DNA sequence, for instance a gene, is present in the DNA of an organism, and if so, to find the distribution of copies of the sequence. The process can be used for diagnosis, for finding homologous genes in other organisms, or, if used with microsatellite DNA probes, for forensics. Northern blotting is similar to Southern blotting, but hybridizes complementary DNA probes with RNA fragments. Northern blotting can thus be used to characterize the tran-
scribed elements of an organism or tissue. In Western blotting, antibodies are used to probe a mixture of proteins, *e.g.* from serum samples. Western blotting has been widely used in clinical research. Southern, Northern, and Western blotting can be viewed as basic probing techniques for respectively the *genome*, the *transcriptome*, and the *proteome*. [22]

### 1.1.2.3 DNA cloning and PCR

**DNA cloning** Cloning of DNA sequences is an important basis of many techniques in molecular biology, *e.g.* for preparation of samples for Southern blotting. Methods for inserting DNA fragments into other DNA, typically viruses or bacterial plasmids (small, circular “mini-chromosomes”) have been developed from the 1970s. In these methods, restriction nucleases are used to cleave both DNA molecules prior to insertion and ligation. The recombined DNA is then introduced in bacterial cells, which grow and divide, producing large amounts of DNA fragments, of lengths ranging from 1000 to 30000 *base pairs (bps)*. Later, the use of *yeast artificial chromosomes (YACs)* has allowed the cloning of fragments up to 1 million bps. Recombinant DNA cloning techniques have been massively used to create genomic DNA libraries of DNA fragments from complete genomes. Similarly, *complementary DNA (cDNA)* libraries are obtained by the cloning of DNA stemming from reverse transcriptase of mRNAs, thus containing only parts of the genome that codes for proteins. [23]

**PCR** Cloning techniques based on recombinant DNA was, and still to some extent is, time-consuming and complex, requiring training and practice. In 1985, the much simpler process of *Polymerase Chain Reaction (PCR)* was introduced by Kary Mullis, although the method was arguably invented some 15 years earlier by Kjell Kleppe, a Norwegian researcher working at the lab of Nobel Price winner Har Gobind Khorana [24]. In the meantime, however, production of enzymes and *oligonucleotides, i.e.* small sequences of single-stranded DNA, had become routine, allowing the process to be economically viable. The technique broadly works as follows: A small sample of DNA is added to a tube, together with large number of copies of two oligonucleotides selected to complementarily match DNA that flanks the region of interest. A thermostable DNA polymerase (from a thermophilic organism, such as *Thermus aquaticus*) is added, before the solution is heated to just below 100°C, which separates the two DNA strands. The solution is then cooled, allowing the hybridization of the oligonucleotides to the two DNA strands. The oligonucleotides act as primers for the DNA polymerase, which extends the primers to form two new double-stranded DNA molecules. The
complete cycle is then repeated several times, doubling the number of DNA fragments every time, achieving e.g. a $10^6$ amplification after only 20 cycles. PCR was quickly adopted as a basic technique in most labs, and used for diverse applications, such as sample generation for blotting and DNA sequencing, and analysis of gene mutations and restriction fragment length polymorphisms. PCR has long been a standard component of routine procedures in e.g. forensics and medical diagnostics. [25]

1.1.2.4 Microarray

**Dot blot** The dot blot was first described by Kafatos et al. in 1979 [26], in essence simplifying the Southern blot method. Instead of separating DNA fragments by electrophoresis, multiple different samples of cloned DNA are spotted next to each other on a single nitrocellulose filter, which is then hybridized with radiolabelled DNA or RNA. The method provides semi-quantitative estimates of sequence concentration, but provides no information on molecule size, as in a Southern blot. The method is, however, simple, and provides an effective method for parallel analysis. It is not, however, possible to reduce the size of spots beyond certain limits, and dot blots were thus not able to meet the demands for large-scale genome analysis. Also, the poorly defined boundaries and shapes of the spots, as well as the dimensional instability of the porous membranes, made the method ineffective for automated analysis. Thus, impermeable supports, such as glass or polypropylene slides, were introduced. Impermeable supports provide easier access, faster hybridization, avoid perturbations by the membrane diffusion process, and improve reproducibility and accuracy. Also, impermeable supports allowed the miniaturization of the dot arrays, providing for parallel analysis on a much larger scale than previously. [20]

**Microarray technology** The first microarray system for monitoring gene expression (i.e. the abundance of mRNA for a particular gene transcript in a particular sample) was described by Schena et al. in 1995, measuring a total of 45 Arabidopsis thaliana genes. [27]. The basic method for profiling gene expression by microarray is as follows [20, 28, 29]: 1) Probe sequences are selected from cDNA libraries, amplified by PCR, and printed on coated glass slides using a high-speed robot. 2) The probe sequences are fixed to the surface by ultraviolet radiation or other methods. 3) The probe DNA is rendered single-stranded by heat or alkali. 4) The target RNA, or cDNA versions of the RNA, are labelled, typically by fluorescent labels, but also radioactive labels may be used. Typically, both a test and reference sample is used, and labeled accordingly. 5) The target sequences are hybridized with
the array. 6) The extent of hybridization is captured using a scanning confocal laser microscope (in the case of fluorescent labels) and imported to a computer as two monochrome images. 7) The images are analyzed in software in order to return an intensity value for each probe. The values are then normalized, and the ratio of expression levels between the test and the reference is typically reported. 8) For visualization, the two monochrome images are usually colored red and green, respectively, and overlaid in a single image. The first example of whole-genome microarray expression analysis of a eukaryotic genome (Saccharomyces cerevisiae, i.e. budding yeast) was published in 1997 [30].

**Applications of microarrays** In addition to transcriptional profiling, microarray technology has been used in a range of other settings, thanks to the generality of the basic platform. One of the most important applications is for use with genotyping. Microarrays with oligonucleotide probes for detection of known *single nucleotide polymorphisms* (SNPs), *i.e.* single base mutations, has been important for mapping the genetic variability in many genomes, most importantly for the human genome. Another application for variation analysis is *comparative genomic hybridization (CGH)*, which is used for analyzing *copy number variations (CNVs)* (abnormal number of copies of parts of the DNA). Other applications include the detection of *DNA methylation* (cytosine modification by methylation at the C5 position), analysis of *metagenomes* (*e.g.* samples of genetic material from environmental isolates, such as soil or water), and sequencing [31]. Important in many applications is the use of tiling arrays, where oligonucleotides are selected to match densely packed probes from a specific sequence. Tiling arrays can *e.g.* be used for detecting copy number variation or analyzing transcription of non-coding regions. The first set of whole-genome tiling arrays representing most of the non-repeating regions of the human genome was published in 2004 [32]. Each of the arrays contained around 390,000 probes, illustrating the immense advances in microarray technology in the decade since its inception.

**ChIP-chip** One microarray-based technology of particular importance to epigenomic studies is the combination of *chromatin immunoprecipitation (ChIP)* with microarray: *ChIP-chip*. Created in 2000 for finding locations of DNA-protein binding, particularly *transcription factor binding sites (TFBS)* [33], ChIP-chip is also used for describing the positioning of nucleosomes and histone modifications (see Figure 1.4). The protein-DNA variant of the methods works as follows: DNA is incubated with proteins in a solution, be-
fore the DNA-protein complexes are stabilized by crosslinking. The proteins are then precipitated (solidified) by the addition of an appropriate antibody. The proteins complexes are then removed from the solution, and washed. Subsequently the proteins are digested by proteinase K. The “free” DNA fragments associated with the complexes are then hybridized using microarrays. Locating transcription factor binding sites, as well as the epigenetic code stored in histone modifications, are crucial for unraveling the complexities of gene regulation [34], as well as for better understanding of cell development and disease [3]. [31, 35]

1.1.2.5 Sequencing

**Sanger sequencing**  Even though the first complete amino acid sequence of a protein, the B-chain of insulin, was determined as early as 1951 by F. Sanger [36], the first complete DNA sequence was not published before 1971, consisting of 12 nucleotides at the end of the bacterial virus Bacteriophage λ [37]. This delay was caused by several factors. First, compared to proteins, DNA molecules are more difficult to distinguish, and have longer chain lengths. Also, the different nucleotides have more similar chemical properties than the different amino acids. Lastly, DNAases, *e.g.* DNA-cleaving proteins, were not known at the time. The discovery of the DNA-cleaving type II restriction enzymes in 1970 paved the way for more advanced sequencing methods. [38]

A revolution in sequencing methodology came with the introduction of gel electrophoresis to the process. The first methods making use of gels were the “plus and minus” method of Sanger (1975) and the base-specific cleaving method of Maxam and Gilbert (Jan. 1977). In Dec. 1977 the “dideoxy method” (commonly called “Sanger sequencing”) was published by Sanger, fixing problems with his previous “plus and minus” method. Basically, the method is based on a synthesis reaction using copies of the single-stranded DNA template to be sequenced, a DNA primer annealed to the sequence, DNA polymerase, and all four nucleotides, one of which is radiolabeled for x-ray detection. Four parallel reactions are set up, where each reaction is in addition doped with a chain-terminating analog of a particular nucleotide, *e.g.* adenine. In each reaction, the DNA templates are extended until a chain-terminating nucleotide is bound. Since many copies of the DNA templates are present in the solution, all subsequences ending with the particular nucleotide will stochastically be synthesized. The solutions are then separated by gel electrophoresis in four parallel lanes and visualized on an x-ray film. The sequence can then be read directly from the order of the bands in the four lanes, where each lane corresponds to a particular nucleotide. Vari-
ations of the dideoxy-method were the most widely-used sequencing technologies up to the introduction of next-generation sequencing technologies in the mid-2000s. [38]

The first full genomes sequenced were the RNA bacteriophage MS2 (3569 bps) [39] and the DNA bacteriophage ϕX174 (5386 bps)[40]. After the introduction of the dideoxy-method, the rate of sequencing accelerated. The human mitochondrial genome (16.5 thousand base pairs (kbps)) was sequenced by the Sanger group in 1981, followed by the 172 kbps Epstein-Barr virus genome in 1984 and the 237 kbps cytomegalovirus genome in 1991, all, at the time, holding the record of the longest DNA sequence published. In 1995, the research group of Craig Venter published the first bacterial genome, the 1.83 million base pairs (Mbps) sequence of Haemophilus influenzae. [38]

**Automation** Automation of Sanger sequencing began in the mid-1980s. Before this, the output of primary sequence data from a single person would typically peak at 60 kbp per week. Automation began with the addition of fluorescent labeling of the primers with four different dyes. This allowed the reactions to be combined, with the colors automatically read by a detector near the bottom of the gel. In 1996, Applied Biosystems (ABI) introduced the first commercial sequencer using capillary electrophoresis (using small tubes instead of traditional slab gels), fully automating the sequencing process for the first time. Each sequencing machine was capable of producing 150 kbps of sequences per day. 300 of these machines were bought by Celera Genomics, founded by Applera Corporation and Craig Venter. Celera based their sequencing efforts on the whole genome shotgun method, in essence randomly sequencing a large amount of short, randomly fragmented DNA, and assembling them afterwards in software. In this context, paired-end sequencing was adopted to help fill the gaps in the assembled genomes. With this method, single, long DNA segments were partly sequenced from both ends. Using the information of the paired and complementary sequences, the sequences on either sides of a gap could be linked, and further closed by sequencing from an appropriate primer. The power of the approach is evident from the fact that Celera, in February 2001, published their draft of the human genome [41] in the same week as the publicly funded and hugely expensive Human Genome Project [42] (although Celera built heavily on progress gained from the public project). [38]

**EST sequencing** An important application of Sanger sequencing is the sequencing of expressed sequence tags (ESTs). ESTs are short sequence
fragments (100-800 bps) generated by unedited, random sequencing from both ends of sequences from cDNA libraries. As the cDNA libraries are generated by reverse transcriptase from mRNAs to DNA, ESTs typically represent short fragments of gene transcripts and have thus been used extensively for locating genes on a genome. Other important applications include the prediction of gene structure (identifying boundaries, splice sites, and alternative splicing variants), gene expression analysis, and for discovery and characterization of candidate SNPs. Problems with EST analysis include sampling bias (*e.g.* over-representation of common transcripts), sequence artifacts and low quality sequences. The methodology has, despite these issues, been important for the mapping of genomes and transcriptomes, much due the simplicity and low cost of the technique. [43]

**Next-generation sequencing** The main limitations of Sanger sequencing is the low throughput and the high cost per sequenced base. In 2004 the first *next-generation sequencing (NGS)* equipment became available. The common characteristic of NGS methodologies is the focus on massively parallel sequencing, making use of cyclic reading of clonally amplified DNA fragments that are spatially separated by some means. The two most common clonal amplification techniques are ‘emulsion PCR’ and ‘bridge PCR’. With emulsion PCR, small, primer-coated beads are isolated in water-in-oil ‘bubbles’, acting as micro-reactors for PCR of a particular DNA fragment. This technique is used in the 454 (Roche Diagnostics), SOLiD (Applied Biosystems), Polonator (Danaher Motion), as well as the Ion Torrent and Ion Proton (Life Technogies) systems. Bridge PCR is used in Solexis-based systems (bought by Illumina). In this technique, DNA fragments are cloned by spatially clustered amplification on an oligonucleotide-covered surface. The spatial organization is maintained by the hybridization of both ends of each DNA fragment to the surface before the polymerase reaction, followed by a denaturation step. The denaturation results in two complementary, single-stranded DNA fragments in close proximity, each fastened to the surface at one of the ends. The cycle is then repeated as many times as necessary. [44, 45]

**Sequencing-by-synthesis** The different NGS systems vary much when it comes to the actual sequencing techniques used. In the 454 system, a technique called pyrosequencing is used. The single-stranded DNA-coated beads are transferred to an array of small wells, with one bead per well. The DNA is primed, and nucleotides of a specific type is added together with DNA polymerase to start DNA synthesis. A chemical reaction including ATP
sulfurylase and luciferase is used to generate light if nucleotides are incorporated. The light is then detected by asynchronous, fiber-optic imaging.

The Illumina Solexa-based systems are, in contrast, based on synchronous cycles of DNA synthesis. The nucleotides are fluorescently labeled and modified so that only one base can be added at a time. All four nucleotides are added for each cycle, the fluorescence of incorporated nucleotides is captured simultaneously in four color channels, and the labels and modifications are removed for the next cycle. [44, 45]

**Sequencing-by-ligation** The SOLiD system is based on sequencing-by-ligation (i.e. linking of two DNA molecules using DNA ligase). Here, the DNA-covered beads are randomly scattered on a glass surface and primed, and ligation with a range of fluorescently labeled octamers (oligonucleotides of eight bases) is carried out. The fluorescence is coupled with the first two bases, and the last three bases are cleaved before the fluorescence is emitted and detected by imaging. Thus, two of five consecutive bases are probed for each cycle. After many cycles, the probe and extended product are cleaved off from the DNA, and a new primer is added, displaced by one base pair. In this way, all bases are, in the end, interrogated twice. The open-source Polonator system uses a different scheme of sequencing-by-ligation.

Another ligation-based sequencing technology is used in the Complete Genomics systems, making use of the rolling circle amplification method, resulting in the formation of DNA nanoballs of approximately 300nm. The DNA is sequenced by the use of a combination of oligonucleotides that anchor to known primers and oligonucleotides that probe unknown positions. [44, 45]

**Semiconductor-based sequencing** A different approach is used in the Ion Torrent systems. Here, the beads are deposited in small wells, with one bead each, similarly to the 454 system. However, the imaging step is replaced by a semiconductor-based detection of changes in pH. Unmodified nucleotides of a specific type is added to the wells, which, if incorporated with the DNA, releases one hydrogen ion each. The resulting change in pH level is detected by a semiconductor sensor plate in the bottom of the wells. [44, 45]

**Third-generation sequencing** The main differences of the NGS systems are on the lengths of sequences generated (typically much shorter than in Sanger sequencing), the base throughput (ranging from 1 to 200 gbps per day), the cost of sequencing, the complexity of library and chemical prepara-
tion, and the accuracy of sequencing. Also, system-specific errors are common, such as the difficulty of detecting homopolymers (i.e. uninterrupted runs of the same nucleotide) in systems such as the 454 and SOLiD.

Another problem with NGS systems is the dependence on DNA samples of a particular size, as well as amplification of the DNA fragments, which may introduce biases. In medical settings, e.g. in samples of tissues, noise is added from the fact that DNA is extracted from several cells. Third-generation, single-molecules sequencing techniques are a step towards sequencing of samples from single cells, in addition to the promised advantages in read length and throughput. An early third-generation system was the Heliscope sequencer (of the recently bankrupt Helicos Biosciences corp). The Heliscope uses sequencing-by-synthesis, but without the need for clonal amplification. Poly-A tailing of the DNA fragments are used for hybridization with poly-T oligonucleotides on the surface of a slide. Sequences are detected asynchronously by the imaging of fluorescence from labeled nucleotides. Third-generation techniques also include the Single-Molecule Real-Time Sequencing (SMRT) chip-based system of Pacific Biosciences, and nanopore sequencing techniques, such as the disposable USB-stick-sized sequencing device, MinION, recently announced by Oxford Nanopore Technologies. [44, 45]

RNA-seq The applications of NGS are even more diverse than the applications of Sanger sequencing, due to the increased throughput. Applications include de novo- or resequencing of whole genomes, exomes, or other targeted subsets of genomes, allowing, among other things, the detection and comparison of most types of DNA variation, fundamentally important for the study of diseases. One particularly important approach is RNA-seq, which uses NGS technology for transcriptome profiling. Conceptually similar to EST analysis, cDNA libraries are sequenced either from one or both ends using high-throughput sequencing machines. However, in order to follow the requirements of the sequencing platform, the cDNA sequences must be fragmented into smaller pieces (200-500 bps), prior to ligation with adaptor sequences. This introduces RNA-seq specific biases and complications. However, the advantages of RNA-seq as compared to EST sequencing or microarray based transcriptome analysis are many. First, throughput is higher. Second, the single-base resolution significantly improves precision. Third, sequencing avoids the high background noise of microarray solutions caused by incorrect hybridization. Fourth, a genome reference need not be available prior to analysis. Fifth, the required amount of RNA is lower, as is the cost of the experiments. Last, estimating transcription by counting sequences catches a wider range of the dynamics, especially for highly and
slowly transcribed RNAs. Post-experimental analyses with bioinformatics tools are important in order to assemble or map the reads to a genome, to reduce technology-specific biases, and to calculate a measure of expression. RNA-seq has been used extensively to characterize gene structure and expression, including alternatively spliced transcripts, in addition to allowing unprecedented characterization of non-coding RNAs. [2, 46]

**ChIP-seq** Another important NGS-based method is ChIP-seq, which is similar to ChIP-chip with the difference that the sequence fragments are sequenced, rather than hybridized on microarray. As ChIP-chip, ChIP-seq is used for characterization of DNA-protein interactions and histone modifications, as well as nucleosome positioning. Advantages over ChIP-chip include higher resolution, the possibility of full-genome coverage including repetitive regions, removal of noise stemming from imperfect cross-hybridization, and better measurements of the entire range of intensity signals. Basically, ChIP-seq consists of sample fragmentation and purification by immunoprecipitation using an antibody specific to the protein or modification of interest. The purified DNA fragments are then sequenced using some NGS technology, and mapped to a reference genome by aligner software. The quality of the antibody and the sample is, of course, vital for good results. Also, as with other NGS applications, it is important to properly determine the depth of sequencing, i.e. the number of reads, needed. Further, a control dataset is required to find regions of enrichment on the genome, a process carried out with ‘peak-calling’ software. The peak-calling software differ on the internal model used, the method for inferring accurate intensity profiles from the sequence tags, how non-uniquely tagged reads are handled, and whether the algorithm focuses on sharp, broad or mixed peaks. [35, 47, 48]

**DNase I hypersensitive sites** The advent of NGS methods have also improved existing methodologies for characterizing chromatin conformations, i.e. physical organization of DNA as wound around the nucleosome protein complexes. It has been known since the 1980s that areas of DNA that are prone to digestion of the DNase I enzyme, so-called DNase I hypersensitivity sites, are positively correlated with having genetic regulatory functions. DNase I HS sites were traditionally identified using Southern blots, but can now be identified with either tiled microarrays or high-throughput sequencing. Full maps of the open chromatin, i.e. DNA physically accessible to regulatory proteins or RNAs, were created for the human genome in 2008 [49]. Also, DNase I hypersensitivity has been used to identify exact binding sites of proteins (8-30 bps), so-called DNase footprints [50].
Hi-C and ChIA-PET Another path of methodology development is focused on the characterization of the organization of DNA in the nucleus, more specifically, identifying regions of the genome that are located closely in three dimensions, even though they may be located far apart on the DNA strand (i.e. in one dimension), or on different chromosomes. Chromosome conformation capture (3C) was developed and published by Dekker et al. in 2002, used for probing one-to-one interactions between restriction fragments (fragments resulting from the digestion by restriction enzymes) [51]. With the advent of NGS methods, 3C was scaled up to all-to-all probing with the use of Hi-C or chromatin interaction analysis by paired-end tag sequencing (ChIA-PET) methodologies. All methods are based on fixation of DNA using e.g. formaldehyde, followed by digestion by a restriction enzyme. Cross-linked pairs of DNA fragments, i.e. pairs of fragments that are located close together, are then ligated (joined) together and reverse-crosslinked, forming short fragments of DNA stemming from two different parts of the genome. With Hi-C, biotin-labeled nucleotides are used to purify these ligated sequences from the rest of the sequences, which are then sequenced by a NGS technique capable of paired-end sequencing and mapped to a pair of restriction fragments. A matrix of ligation frequencies between all restriction fragments are then created, giving a whole-genome estimate of 3D closeness. With ChIA-PET, chromatin immunoprecipitation is used to pull down only the ligation junctions bound by a particular protein. [52]

1.1.3 Implications of NGS on data analysis

The concept of the “thousand-dollar genome”, i.e. the idea that the cost of fully sequencing a human genome could be reduced to a mere $1,000, has long been considered the “Holy Grail” of sequencing. Achievement of the goal of the thousand dollar genome is by many expected to herald the age of personalized medicine: “US$1,000 is a magic number that would attract customers and would put genome sequencing in the same financial bracket as many medical tests” (quote George Church, Professor of Genetics at Harvard Medical School, [53]). The possibility of using full-genome sequencing as part of routine procedures in clinical settings holds the promise of individually tailored diagnostics and therapy for complex diseases, such as cancers. Benefits may include reducing the patient’s need to go through ineffective therapies, lower side- and off-target effects of drugs, and reducing the costs of health care [54].

The cost of sequencing the first human genome was about $3 billion dollars. The effort took several international institutes, hundreds of researchers and 13 years to complete [55]. At the time, the thousand-dollar genome
would seem an impossibility. In 2007, the noted geneticists J. Craig Venter and James Watson independently sequenced and publicly presented their personal genomes, the cost of the latter estimated at “well under US$1 million” [56]. In 2012, the company Life Technologies started shipping their Ion Proton™ System, which is able to sequence the human *exome* (~2-3% of the genome that is protein-coding exons) for under $1,000. The improved Ion PII™ Chip, which is designed to sequence the human genome (~3 billion bps) in “just a few hours for $1,000” is expected to ship during 2013 [57]. The Archon Genomics X PRICE is a competition that “will award $10 million to the first team to rapidly, accurately and economically sequence 100 whole human genomes to a level of accuracy never before achieved”, and to a cost near, or under $1,000. The competition is concluding 31 October 2013 [58]. Clearly, the age of the thousand-dollar genome has arrived.

![Diagram](image_url)

**Figure 1.5:** Contribution of different factors to the overall cost of sequencing projects across time. Left: the four-step process: (i) experimental design and sample collection, (ii) sequencing, (iii) data reduction and management, and (iv) downstream analysis. Right: the relative impact of these four components of a sequencing experiment at three time points: previous to the development of Next Generation Sequencing (NGS) techniques, currently, and in the future. Figure reprinted from [55] (published under the BioMed Central copyright and license agreement [59]).

As figure 1.5 illustrates, however, the cost of sequencing does not tell the whole story. As the cost of sequencing is reduced, it is predicted that the overall cost of sequencing projects will be dominated by experimental design, sample collection, and downstream analysis, the components that require most human resources [55]. Personalized medicine is an obvious example of the problem, the goals of which one may find overly optimistic.
In order for full-genome sequencing of individuals to have clinical impact, a series of analytical and procedural challenges need to be overcome [60, 61]: 1) Processing sequencing reads to find genetic variants, and perhaps also expression data (RNA-seq) and data on epigenetic modifications (methyl-seq, ChIP-seq). 2) Predicting functional consequences of the mutations and genomic alterations, both for variation occurring in coding regions, but also those occurring in non-coding regions. 3) Integrating data to identify complex interactions relevant for the phenotype, such as alterations in cancer, signaling and regulatory pathways. 4) Associating the variation in proteins and pathways with drugs that affect their function or activity, and draw clinically robust conclusions regarding drug selection, dosing, and adverse effects. 5) Developing user-friendly and interoperable software systems that are relevant to clinicians, bioinformaticians and computational biologists and ensures effective collaborations between these heterogeneous groups. Overall, the molecular processes leading to disease are usually very complex, and current methodology typically only finds small effects of genetic variation on disease risk. In addition, knowledge extracted from databases is often incomplete and noisy. Major obstacles to personalized medicine are thus the bottleneck represented by bioinformatics analysis and the development of methodology and software, as well as the shortage of individuals skilled to interpret individual genomic data [62].

When announcing the first draft of the human genome, then-president Clinton declared that it would “revolutionize the diagnosis, prevention and treatment of most, if not all, human diseases”. This revolution is not yet a reality; development of new drugs seems to take more time than the first predicted [63]. Traditionally, common multifactorial illnesses are believed to stem from combinatorial effects of common mutations. Recently, rare mutations have been found to hold a more important place than previously believed, a recognition that has lead to “a major paradigm shift in human genetics” [64]. This realization drastically increases the importance of individually tailored diagnostics and treatment, and thus, the stress on overcoming the challenges of data integration and analysis that comes with it.

Personalized medicine is still just one example where next-generation sequencing techniques can be used. NGS technologies are currently mostly used for basic research investigations in many directions. But the same holds here: as the price of full-genome sequencing is dropping, the importance of accurate and robust methods for integration and analysis of whole-genome data, along with corresponding user-friendly software tools, is steadily becoming more important.
1.2 Data representation

1.2.1 Fundamental aspects of data representation

At the fundament of data analysis lies the issue of data representation. In the context of bioinformatics analysis, two types of data representation are especially important. The first type is the mathematical formulation of data needed for the exact specification of e.g. statistical tests. The second type is the actual data structures used in the computer, for storage on disk and/or for use in memory during the analysis algorithms. The mathematical formulations has consequences for the type of analyses available, and the data structures has consequences for the complexity and resource requirements of the software solutions, and thus the feasibility of carrying out the analysis in “the real world”.

One may, however, look at data representation at a more abstract level, removed from the technicalities of mathematics and informatics. This is the standard “boxes-and-arrows” level commonly used e.g. when drawing on whiteboards or presenting figures in scientific papers. The focus is on conveying ideas to an audience, and to facilitate further discussion on a conceptual level. To distinguish between different data representations at the abstract level, it may be helpful to look at several basic properties of the representation. The most fundamental one is obviously (1) identification the basic entities used. Related to this is the (2) identification of the domain of the entities, i.e. the totality of which the entities take part. Important is also (3) what kind of information or measurements that are available for the entities. Lastly, one can inquire (4) whether and how interactions or relations between the basic entities are represented. Using double-blind, randomized clinical trials as an example, the basic entities are the patients under study. Here, the domain would refer to the population of patients onto which the results of the study may be generalized (which may not be much larger than the total group of patients under study if the selection criteria are biased [65]). The information available for each patient is e.g. demographics, symptom measurements, and treatment info. In double-blind clinical tests, information on the relations between patients are not explicitly represented, other than what can be gathered from correlations between the different variables.

Going from the abstract “boxes-and-arrows” level to the technical level, one needs to define an exact mathematical and/or informatic representation of data that fits with both the abstract ideas and the exact technical requirements. In the case of double-blind, randomized clinical trials, the mathematical representation of the data is usually in the form of stochastic variables, while the informatical representation is usually a table, where
each row represents a patient, and where the columns, representing variables, may have different data types (e.g. numbers, categories, or text). Other representations than these are possible; the data could e.g. be represented in the computer in the form of a relational database instead of as a single table. There is a complex interplay between these various levels of representation. In the case of clinical trails, clinical reality puts limits to which patients and measurements are available, while statistical considerations drives the exact requirements for the collection of data.

In the following, the three-level schema sketched above (abstract, mathematical and informatical representation) will be used to discuss two basic representational paradigms in bioinformatics: network-based and sequence-based representation.

1.2.2 Data representation in bioinformatics

1.2.2.1 Network-based representation and analysis

Basic entities The word ‘gene’ was first used by Wilhelm Johannsen in 1906 to describe the discrete units of heredity, following the laws described by Mendel (reviewed by Gerstein et al. in [66]). One could thus argue that the gene has been a basic representational entity of the field of genetics since its beginnings, and perhaps the most important one. Although the concept of the gene has undergone dramatic changes during the last century [66], genes has maintained a central position as a representational concept. Many bioinformatics databases and tools are based around the gene as the central entity. One common type of analysis has focused on the discovery of enriched associations of related terms based on a list of genes, such as DAVID [67] or Gene Set Enrichment Analysis (GSEA) [68]. Gene-centered databases include sequence databases, such as GenBank [69, 70] or EMBL [71, 72], or ontology databases such as the Gene Ontology [73], where genes are classified according their association to terms from a controlled vocabulary. As a gene is not a distinct molecule or another clearly defined physical entity, the domain of genes will not directly refer to a physical totality, such as a cell, but rather to a more abstractly defined totality, such as the total set of genes for a particular genome, or in the context of comparative genomics, the evolutionary history of the genes. Herein lies one of the more fundamental problems with using genes as the central representational entity: measurements are not directly available for the genes as such, but must be inferred from related measurements, such as the concentration of mRNA transcribed from the gene sequence.

Other common representational entities are RNA transcripts or proteins.
These more directly refer to distinct molecules, the concentration of which can be measured more directly, via e.g. microarray technology or RNA-seq (in the case of RNA), or quantitative mass spectrometry (in the case of proteins). The typical analysis domain is thus the cell, either of a particular cell type, or a generalized cell of some sort, and the totality under study is then the transcriptome or the proteome, respectively. Other data related to the transcripts or proteins are the sequence (of RNA or amino acids, respectively), post-transcriptional and post-translational modifications, cell localization, and folding structure, to name a few.

**Network-based data integration** Much research has focused on the task of integrating “omics” datasets of e.g. genes, transcripts, or proteins, as described above. Integral to such efforts are the many interactions that find place between entities of the different types [8]. In addition to the “central dogma”-processes of transcription from DNA to RNA and translation of mRNA into polypeptides, much focus has been on characterizing protein-DNA interactions (e.g. transcription factor regulation of genes) and protein-protein interactions, as well as the integration of “omics” data with described biological pathways for metabolism, regulation, or signaling. Much of this work has happened under the banner of “systems biology”, a branch of computational biology which is aimed at producing models of the complete behavior of intricate biological systems, such as the cell, with a special focus on the dynamics of the systems under study [74, 75]. Such integrative model building, making use of entities like genes, transcripts, proteins, and metabolites, fits well with using graphs, or networks, as the mathematical representational framework. The entities are represented as nodes in such graphs, while the edges represent interactions, as e.g. outlined above [8]. Graph-based analysis techniques are thus common, such as methods for characterizing structural properties, or for detecting interesting motifs or clusters [76]. Bayesian networks and differential equations are also used for building models and analyzing system dynamics [8, 75]. Graph-based mathematical models integrate well with the storage of data as tables in relational databases, which is the most common storage solution for biological data based on such molecular entities as described above. One major practical hurdle is, however, the matching of database keys, such as gene or protein IDs, across databases [67].

### 1.2.2.2 The importance of non-coding DNA

The major limitation with the network-based approach to integrating “omics” datasets, as described in the previous section (1.2.2.1), is the lack of focus
on DNA sequences other than the 2-3% that constitute protein-coding exons. One counter-example is the increased addition of non-coding RNAs to transcriptomics databases in recent years [77], but this does not change the fact that many of the biochemical events happening other places along the DNA are not included. A natural explanation for this lack is the close connection between particular technologies and particular data representations. More specifically, early techniques, such as blotting, as well as the later microarray technologies were inherently built around probes, typically of transcripts. Even though sequencing technologies have been living side-by-side, it is not until the later years that the bulk of available transcriptomic and epigenomic data has been based on sequencing techniques, with the inherent focus on relating entities along genome sequences.

**The ENCODE project** The Encyclopedia of DNA Elements (ENCODE) project was launched in 2003 with the aim of delineating “all functional elements in the human genome sequence” [78]. The term “functional elements” originally included:

- protein-coding genes, non–protein-coding genes, transcriptional regulatory elements, and sequences that mediate chromosome structure and dynamics; undoubtedly, additional, yet-to-be-defined types of functional sequences will also need to be included [78].

When the initial results of the ENCODE project were published in 2012, the datasets additionally included:

- RNA sequencing (of mRNA, but also of non-coding RNA of different types)
- Protein-binding regions for 119 transcription factors
- DNase I hypersensitivity sites and footprints
- Regions of histone modification
- DNA methylation
- Physical chromosome-interacting regions

All of the above-mentioned datasets were investigated genome-wide, and for a range of different cell lines [79]. None of these datasets (except for perhaps transcribed RNA), are readily represented by graphs without losing their fundamental nature, i.e. their direct association to regions on the human DNA sequence.
Redefining genes  The pilot phase of ENCODE, where around 30 megabases (~1%) of the human genome were scrutinized, was concluded in 2007 [80]. It became clear, already then, that the majority of the genome was, in fact, transcribed, not just the coding regions of the genes. Also, intricate patterns of gene regulation were described, with transcription factors found to bind more generally and distally to transcription start sites than previously believed. The findings of the pilot phase of ENCODE challenged the existing understanding of what constitutes a gene, prompting a re-evaluation of the term “gene” itself. Gerstein et al. [66] reviewed the history of operational definitions of the term “gene” and presented a set of findings that were causing problems for the existing definition. Some of the findings are listed in the following:

- Distant and complex gene regulation challenges the notion of a gene as a compact genetic locus
- Some genes have been found to overlap one another, either with one gene inside the intron of another, or by sharing the same DNA sequence in different reading frames or on the opposite strand. The notion of genes as non-overlapping hereditary units is thus challenged.
- The discovery of alternative splicing of genes in 1977 had already changed the “one-gene-one-transcript” notion to “one-gene-many-transcripts”. The discovery of trans-splicing (i.e. ligation) of mRNA molecules, as well as tandem chimerism, where two consecutive genes are transcribed into a single RNA, further challenge this notion, requiring some sort of “many-genes-many-transcripts” definition.
- Gene expression depends on epigenetic modifications and chromatin structure, which challenges the notion that the gene product can be predicted only by DNA sequence.

A novel definition of the gene was proposed by Gerstein et al., allowing, among other thing, a single gene to have multiple genomic loci. Other novel definitions have also been proposed [81]. Furthermore, in the 2012 update of ENCODE, it was conservatively reported that 62% of the genome was covered by long (>200 bps) RNA transcripts or exons [79] (collected by the GENCODE annotation group [82, 83]). Of these bases, only 5.5% were explained by exons. A significant portion of the remaining non-coding transcripts are characterized as long noncoding RNAs (lncRNAs), some of which has been associated with diseases [84]. It was thus argued that the transcript, and not the gene, should be considered the basic unit of inheritance [85].
Variants in non-coding loci  Another development pointing towards an increased focus on non-coding genomic loci is the acknowledgement that most genomic variants associated with phenotypic traits, such as disease, are not found in coding regions. As of sept 2012, around 93% of the genomic variants emerging from genome-wide association studies (GWAS) was found in non-coding DNA sequence. Furthermore, 76.5% of non-coding GWAS SNPs were found within, or in perfect linkage disequilibrium with (thus inherited together with), DNase I hypersensitivity sites (as reported by ENCODE), which again is highly associated with transcription factor regulation of genes [86].

Function of non-coding DNA  In total, 80.4% of the genome was found to be covered by at least one ENCODE-identified element, in at least one cell line. These regions were thus assigned “biochemical function”. It was widely reported in the media that these findings had laid to rest the idea that most of the human DNA is non-functional, “junk DNA”, as first put forward in a paper by Susumu Ohno in 1972 [87]. This claim has been controversial scientifically, as is evident by a recent rebuttal [88]. The rebuttal argues that the ENCODE claim that 80% of the genome is functional (as paraphrased by the authors of the paper) is incorrect, as only ~10% of the genome is under evolutionary, purifying selection. Without going into details on the arguments: it may seem that much of the conflict is in essence a discussion of definitions of terms like “functional” and “junk DNA”. It is, regardless, clear that the datasets released by ENCODE presents convincing arguments against “omics”-analyses that are only based on data represented as graphs, with genes, transcripts, proteins, or other molecular entities as nodes. Another data representation paradigm, based on the DNA sequences themselves, is thus needed. Such DNA-centric representations have also been developed and used alongside the graph paradigm since the first full-genome sequencing projects, but has gained importance the recent years, as argued above. The next section with present history and usage of data representations based on reference genome sequence coordinates.

1.2.2.3  Dataset types based on sequence-relative coordinates

The first full genomes sequenced were the RNA bacteriophage MS2 [39] and the DNA bacteriophage ΦX174 [40]. Already then, the practice of referencing to parts of the sequences by genomic coordinates was well established. As is intuitive, the coordinates were defined by counting from the start of the sequence (appropriately selected), one nucleotide at a time. In the paper introducing the MS2 genome, for example, subsequences that were predicted
to bind together, forming RNA hairpins, were denoted as “segments” and identified by a pair of start and end positions [39]. This practice was also used in software developed by Rodger Staden “to process the large amounts of data produced” during the sequencing of ΦX174, one of the first software packages created for sequence analysis [89]. Locating and describing interesting structural and functional elements using sequence-relative coordinates has been indispensable for understanding the function of DNA sequence. The following will introduce some common types of elements and data represented in this manner.

**Genome annotation**  One of the first questions asked after having successfully sequenced and aligned a genome is: “where are the genes”? This question is the central investigation of what has come to be called *genome annotation*. The term can mean annotation on the level of nucleotides, proteins, as well as processes [90], but is mostly used for the former, and this is the meaning which will be used in the following. Traditionally, genes were located using gene prediction techniques depending only on the DNA sequence itself, using so-called *ab initio* gene prediction. Methods are diverse, but typically include searching for *open reading frames* (ORFs) while taking into account *codon bias*, intron-exon length distributions, as well as consensus sequences for start codons, splice sites, and stop codons. Although *ab initio* gene predictors are good at predicting whether a nucleotide is part of an exon, they typically have more problems predicting the exon boundaries and the intron-exon structure of genes, and are not able to predict *untranslated regions* (UTRs) and alternatively spliced variants. Modern annotation pipelines usually combines *ab initio* gene prediction with evidence from RNA-seq or EST experiments, in addition to alignment with protein homologues from related organisms. [90–92]

**Repeating elements**  Traditional genome annotation also includes the identification of repeating elements, which is a common denominator for quite different types of sequences, covering over 50% of the human genome. On the top level, repeating elements are divided into two classes: *dispersed* and *tandem* repeats. Dispersed repeats include transposons, i.e. sequences that are able to copy or move between different loci on the same genome. Retrotransposons, i.e. elements transposed via reverse transcriptase of RNA products, include *long interspersed elements* (LINEs), *short interspersed elements* (SINEs), and *long terminal repeats* (LTRs), each having a specific structure and mechanism for transposition. LINE, SINE, and LTR elements cover respectively 21%, 13%, and 8% of the human genome. DNA-based
transposons constitute another class, covering another 3%. Tandem repeats are sequentially repeated sequences, divided into DNA satellites, minisatellites, and microsatellites, depending on the size of the repeat unit. Also, DNA sequences transcribed as tRNAs and ribosomal RNAs (rRNAs) are also classified as repeating elements. Lastly, repeating elements include paralogous genes and pseudogenes, *i.e.* genes that are copied in large DNA duplications, and retropseudogenes, *i.e.* reversely transcribed mRNAs. Repeating elements of the various types are associated with different kinds of function, although much is not yet clearly understood. The identification and annotation of repeating elements for analysis are thus important. The customary way of annotating repeating elements is by aligning with sequences of known repeats, using *e.g.* software such as RepeatMasker [93]. The number of *ab initio* repeat finding programs are growing, but have been found to produce profoundly different results [94], [95]

**Non-coding RNAs** The annotation of non-coding RNAs is a more recent discipline than the annotation of genes and repeating elements, but one that is advancing rapidly [92]. Annotation of RNAs means finding the location of the DNA from where the RNA was transcribed. In addition to the above-mentioned tRNAs and rRNAs, non-coding RNAs include microRNAs (*miRNAs*), small interfering RNAs (*siRNAs*), piwi-interacting RNAs (*piRNAs*), small nuclear RNAs (*snRNAs*), small nucleolar RNAs (*snoRNAs*), and long noncoding RNAs (*lncRNAs*), among others [85, 96]. Non-coding RNAs seem to be involved in a range of cellular processes, such as alternative splicing, post-transcriptional regulation and chromatin modification, and dysregulation of non-coding RNAs have been found to have relevance for many diseases. Much of their functionality is still not well understood [96–98]. The reference human genome annotation included in the ENCODE project, GENCODE, includes the annotation of many novel long non-coding RNAs not included in previous data sets [83]. Another ENCODE-related effort provided an overview of the transcriptional landscape of the human genome in 15 different cell lines, making use of RNA-seq, among other technologies [85]. The paper reported evidence of thousands of coding and non-coding transcripts not yet included in the GENCODE annotation, illustrating the need for further work on transcript annotation.

**Gene expression** The genome annotation types as described above can, if one disregards variation between individuals or specific cells, such as cancer cells, be viewed as static elements on reference genome sequences. In contrast, much of the data resulting from modern experimental techniques
provide information of a more dynamic nature. A basic example is measures of gene expression, as measured by microarrays or RNA-seq. These intensity measures contain information on the abundances of transcripts in a particular cell or domain, and are thus directly connected to the transcript as the basic entity rather than the DNA. Traditionally, microarray analyses are carried out with genes as entities, with expression values arranged in tables, with the different genes as rows or columns. It is, however, in many cases fruitful to map the intensity values to the DNA region from where the RNA was transcribed, in order to compare the values with other data mapped in the same way. In the case of RNA-seq data, such mapping is the natural byproduct of analysis, as the individual reads usually, at some point, are mapped to regions of a reference genome. However, the common way of analyzing expression is still to count the number of reads that, along some rules, overlap each gene, and then continue the analysis in a gene-centric way, similarly to earlier microarray analysis [99]. Such gene-centric analyses will capture more complex transcriptional patterns.

**Protein-DNA interactions** Transcription factors, *i.e.* gene regulatory proteins, are essential to characterize in order to understand gene regulation [14]. Traditionally, predictions of *transcription factor binding sites* (TFBS) has been handled by a range of sophisticated computer algorithms and learning methods. Such methods are mostly based on analysis of DNA sequences, typically using the sequences of gene promoters (a region in front of a gene where the basic transcription machinery docks [14]) as input [100]. As described in section 1.1.2.5, the arrival of ChIP-chip and ChIP-seq analysis, as well as DNase I footprinting, has allowed the direct probing of DNA-protein interaction sites, at a genome-wide scale. Using ChIP-seq as an example, sequence tag reads are first mapped to a reference genome, after which a signal profile is generated over the whole genome. The signal profile is then used to locate *peaks*, *i.e.* regions of enriched read intensity, typically of size 100 bps and up [47, 48]. The set of DNA-binding regions over the genome is typically specific for the combination of transcription factor and cell or tissue type used in the experiment. Also DNase I footprints return regions of DNA binding, or more specifically, regions where bound proteins protect the DNA from cleavage by the DNase I enzyme. The DNase I regions are shorter (8-30 bps) than the ChIP-seq peaks, specific of the cell or tissue type, but unspecific regarding the particular protein [50].

**Epigenetic markers** Epigenetic marks are information elements stored as chemical modification of cytosine nucleotides (DNA methylation) or on
the tails of the histone proteins (histone modification) that make up the nucleosome, around which DNA is wound. DNA methylation occurs mostly in CpG dinucleotides, i.e. positions along the DNA where cytosine is immediately followed by guanine. DNA methylation is essential to normal development, and has been involved in a range of processes, among other things, in tissue-specific gene expression regulation. DNA methylation has also been found to be inherited through cell division. Technologies for characterizing DNA methylation include bisulfite sequencing and digestion with methylation sensitive restriction enzymes (MSRE) before array-hybridization or sequencing. DNA methylation datasets are typically based on a particular cell line or tissue type and contain the exact locations of the methylated cytosines relative to a reference genome. [3]

As introduced in section 1.1.2.5, ChIP-based methods have been widely used to characterize modifications of histone proteins. Histone modifications include methylation, acetylation, phosphorylation, ubiquitination, and other chemical modifications of particular amino acids along the N-terminal tail of the histone proteins. The combination of the particular histone protein (of the four core histones H3, H4, H2A, H2B), the number of the amino acid, and the type of chemical modification, provide information that can be recognized by other parts of the cellular machinery. Thus, a high number of different types of histone modifications exist, as well as variants of the histone proteins themselves, and deciphering the “code” of their function is a complex undertaking. Examples of such function include the involvement of methylation of H3K4 (histone 3, residue number 4) and H3K27 in respectively transcription activation and repression. Datasets are based on the combination of a particular modification with a particular cell line or tissue type, and typically contain e.g. raw ChIP-seq reads, signal profiles, or enriched peaks, where the peak regions typically coincide with the region of the DNA that is wound around the nucleosomes containing the modification in question. [3, 34, 101]

**Cis-regulatory modules** Some regions of the DNA are associated with regulatory processes of genes. These regions are called transcriptional regulatory elements (TREs). TREs are divided into two classes: 1) trans-acting regulatory elements, which are genomic elements that act via other molecules, such as genes coding for transcription factors (introduced previously), and 2) cis-regulatory modules (CRMs), which typically are small regions of the DNA containing clusters of TFBS. CRMs include promoter elements, enhancers, silencers, insulators, and locus control regions (LCRs). Promoters, enhancers, and silencers are all regions containing binding sites for several transcription factors, being involved with the regulation of a gene. Promot-
ers lie just upstream of genes, while enhancer and silencer regions located
distally from the genes they regulate. Enhancers and silencers provide up-
and down-regulation of transcription, respectively. Insulators are regions
thought to either block communication between enhancers and promoters,
or prevent the spread of repressive chromatin, or both. LCRs are groups of
regulatory elements (e.g. enhancers, silencers, insulators) involved in the
regulation of a cluster of genes. A common model explaining the ability of
distal regions to be involved in transcription regulatory processes is by loop-
ing of DNA, bringing the regions close (in three dimensions) to the promot-
ers, as illustrated in Figure 1.3. Prediction of TREs is a rather complex mat-
ter. One method used for predicting TREs is by using reporter-gene assays,
where e.g. a promoter region is cloned in front of a easily assayable reporter
gene in a plasmid. Other methods include algorithms scanning of the DNA-
sequences for e.g. promoter-related motifs and contexts. Also, the use of
chromatin-information from e.g. ChIP-based or DNase I-based techniques
has been used. One example of this is the use of Hidden Markov Models
(HMM) to map promoters and enhancers in nine human cell types, based
on diverse chromatin marks [102]. Another example is from one of the EN-
CODE articles, where chromosome conformation capture carbon copy (5C),
a further development from 3C technology, was used to interrogate interac-
tions between transcription start sites and distal, regulatory elements in the
ENCODE pilot regions [103]. [14]

Other chromatin-based datasets As previously indicated (in section
1.1.2.5), ChIP-based methods can be used to identify the locations of nucle-
osomes, or more specifically, which regions of the DNA (~ 147 bps) that are
wound around nucleosomes. The positioning of nucleosomes regulates the
access of enzymes to packed DNA, e.g. by directing transcription initiation
by RNA polymerase (Pol) II to the start of the genes rather than in the middle.
Typical structures, including a nucleosome-free region at the start and end
of genes, allow the prediction of transcription start sites and promoters by
nucleosome positioning data. The format of datasets is in this case similar to
the different versions of ChIP-seq datasets as described under Epigenetic
markers above.

The variants of chromosome conformation capture (3C) technologies pro-
duce datasets that are drastically different from most of the other types of
data presented in the above. The crosslinking and ligation steps lead to the
sequence reads being composed of a pair of sequences from different parts
of the genome, even from different chromosomes. The most important char-
acteristic of such data is therefore the linking of the regions, and not just the
regions in themselves. In the case of Hi-C, the regions represent a parti-
tion of the chromosomes by a restriction enzyme. In the case of ChIA-PET, the regions are similar to enriched ChIP-seq reads, but are linked together in a pairwise manner. The pairs of linked regions results from either self-ligation in the same protein complex, or intra-ligation between different protein complexes [104]. [52]

**Variation** As most of the genome is identical between individuals, focused characterization of the parts of the genome that varies is important in order to understand phenotypic differences, such as disease susceptibility. Such variation comes in different forms. The basic, and perhaps most researched, type of variation is in the form of *single nucleotide polymorphisms (SNPs)*. These are common single base mutations that by definition have a *minor allele frequencies (MAF)* in the population of more than 1%. For an individual, the majority of single base mutations are common in the population, *i.e.* categorized as SNPs. Much research has focused on the annotation of these common variants, both for technological reasons, but also because their common occurrence has been thought to argue for their importance in contribution to disease phenotypes. However, genome-wide association studies of common variants have, as indicated previously, largely failed to explain most of the heritability of common diseases, and it now seems that the contribution of rare mutations is larger than previous believed [64]. [105]

Other variant types include: insertion-deletion variants (indels), where nucleotides are either inserted or removed as compared to a reference genome; inversions, where a part of the DNA sequence is reversed; duplications, where a region of DNA is sequentially copied; and translocations, where sequences are deleted from one locus and inserted into another. Large-scale variations, such as chromosomal rearrangements or full-chromosome duplications, have for a long time been observed by microscope. With the advent of molecular biology, small-scale variation (of sizes <1 kbp) could be observed and thought to constitute most of the genetic variation. Lately, submicroscopic variants of sizes from 1 kbps up to 3 Mpbs have been found to be more pervasive than previously believed. *Copy number variation (CNV)* is a generic term for variation in this size spectrum where a variable copy number is observed compared to a reference genome (*i.e.* insertions, deletions, and duplications resulting in the number of copies being different from the typical two copies present in a diploid organism). CNVs have been the focus of much research in the recent years. [105–107]

By 2009, only four complete individual human genomes had been sequenced, two Caucasian genomes, one Asian and one African. With the dramatic reduction of sequencing costs, individual genome sequencing has
increased dramatically. In November 2012, the 1000 Genome Project published the genomes of 1092 individuals from 14 populations, based on a combination of low-coverage, whole-genome sequencing and high-coverage, exome sequencing [108]. This data source provides an invaluable resource for scrutinizing genomic variation of all sorts, including the characterization of rare variants.

**Virus integration sites** Retroviruses, such as HIV, are RNA-based viruses that replicate through integration by reverse transcriptase into the host DNA, before using the cellular transcription and translation machinery to generate new viral proteins. Retroviral integration sites are not entirely random, and different types of retroviruses show different integration site preferences. Large-scale studies of retroviral integration sites are thus important contributions for understanding retroviral behavior. [109]

**Sequence-derived datasets** Some genomic datasets do not contain experimental information at all, but are purely created by algorithms taking the DNA sequence itself as input. One basic dataset of this type is GC content (the number of guanines and cytosines in a sliding window of some length over the sequence). Another helpful type of dataset contains the positions of oligonucleotides (or nmers), i.e. the genomic positions of all occurrences of a particular subsequence.

A particular type of sequence-based dataset is constituted by predictions of CpG islands, which are regions of the genome with elevated GC content, after some variation of region length and GC content thresholds. CpG islands are found to colocalize with promoters and exhibit little of the otherwise pervasive DNA methylation of CpG sites (see section Epigenetic markers. CpG island prediction tools have, however, started to include other genomic and experimental information, such as repeat predictions and chromatin modifications. [110]

As described in section 1.1.2.1, double-stranded DNA denatures, or melts, into single strands at a certain temperature, allowing a range of techniques based on hybridization of single-stranded DNA, such as PCR. PCR quality is dependent on finding the right temperature of denaturation, depending on the primer used. In this case, as in other settings, prediction of DNA melting temperature is useful. As the melting temperature is dependent of the base composition of the sequence, algorithms have been designed to predict the melting temperature for each base pair along the DNA, based on the DNA sequence. [111]
1.2.2.4 Track model of genomic datasets

The UCSC Genome Browser, a web-based system for visualizing sequence-based datasets, first published in 2002, introduced the concept of “track” to delineate such datasets [112]. The idea of the term seems to come from the fact that such tracks were visualized side-by-side in the software, allowing interesting biological observations to be drawn by visual inspection (see Figure 1.6 and section 1.3.1.1). Although the “track” concept has not been widely adopted outside the world of genome browsers, there is no consensus on alternative nomenclature, either, at least in the knowledge of the author of this thesis. Typically, very general terms such as “genome-scale data” is used, or the datasets are referred to according to the technology used for manufacture, such as “ChIP-seq peaks”. The “track” metaphor has some appealing connotations not present in the other terms: first, the term conveys the sequential nature of the datasets (even though, unlike track runners, the entities in a genomic track usually stand still); second, illustrating the DNA sequence as a line is intuitive and widely used in molecular biology; third, the track metaphor carries with it the notion that the datasets does not contain the DNA sequence itself, but information that can be placed somewhere along the sequence; and last, the comparison of tracks at the same position (a number of nucleotides from the starting position) is similar to standing at a particular place (e.g. a number of meters from start) along a race-track and inspecting the parallel tracks.

Figure 1.6: Screen shot of the UCSC Genome Browser [112] of an area around the BRCA1 gene, with coordinates relative to the human reference genome ‘GRCh37/hg19’. The tracks displayed are as follows: GC content; genes (RefSeq); ChIP-seq signals of the histone modifications H3K4me1, H3K4me3, and H3K27Ac for 7 cell lines (ENCODE), DNaseI Hypersensitivity Clusters in 125 cell types (ENCODE), CpG Methylation of 6 cell lines (ENCODE), common SNPs (dbSNP); and Simple Tandem Repeats (TRF)
Representing tracks at the abstract level  The data representation framework described in section 1.2.1 introduced four aspects of data representation: the basic entities, the domain of the entities, the information available for the entities, and the relations between entities. The basic entities when representing data as “tracks”, or “genomic tracks”, is single nucleotides. A region of the genome can thus be thought of as the set of nucleotides contained in the region. The domain, or the totality of these entities is the reference genome used for the dataset. With reference genome means a version of a genome assembly, or other sequence type, sanctioned as a common reference by some acknowledged body, such as the Genome Reference Consortium [113]. By this point of view, the different types of information or measurements available for the nucleotides are kept as separate tracks. The inclusion of a nucleotide in a track, e.g. a track of transcription factor binding sites of a particular protein, conveys the information that some property is set for the nucleotide, in this case that the nucleotide is part of a TFBS of the protein in question. Note that the definition of nucleotides as entities, as described here, is different, but compatible, with the definition of “genomic elements” which will follow in Paper II.

The last aspect of the data representation framework is whether and how relations between the entities, in this case nucleotides, are represented. This question has several answers, depending on what kind of relation is considered. As a track can be though of as a set of nucleotides that are collected together by some algorithm or experimental technique, the tracks themselves are representations of nucleotide relations. For example, a track of exons relates all the contained nucleotides in that they are all part of exons. The tracks can also be inter-related in different ways, but such inter-track relations are not usually represented directly in the data structures. Instead, inter-track relations are typically indirectly represented as structural similarities between the tracks, as answers to questions like: do the tracks overlap?; if so, how many nucleotides are overlapping?; are the nucleotides of these tracks located close together?; or is there a connection between the values attached to the nucleotides in the tracks? Answering these and more complex versions of such questions is the task of many of the data analysis tools and techniques to be introduced in chapter 1.3.

Another type of relations may arise from relations between specific nucleotides or regions directly, other than being part of the same track. In the case of techniques for probing the three-dimensional structure of DNA, such as Hi-C and ChIA-pet (see section 1.1.2.5, ways of relating a pair of regions from different parts of the genome, are needed. Another example is data relating regions involved in transcriptional regulation to the genes they regulate. Often, one simply assumes that regulatory regions regulate
the closest-lying gene, which is a simplification. Recent datasets now allow more complex regulatory relations to be studied at an unprecedented detail [114]. Data structures for representing such inter-related data as tracks are yet not well developed.

**Representing tracks at the technical level**  Mathematically, two common representational paradigms for tracks are common. The first, and most common, is the representation of tracks as intervals (or regions/segments) along the coordinate system defined by the reference genome. More specifically, the reference genome constitutes a discrete, one-dimensional coordinate system, allowing methods of euclidean geometry. This includes questioning the amount of overlap between the track, which is a typical operationalization of related-ness (as in e.g. [115]). The other type of mathematical representation is the description of the data as a sequence of numbers, or variables, one for each nucleotide. This is especially applicable for datasets that assign a value to each nucleotide, such as ChIP-seq signal tracks, or DNA melting tracks, but can also be used for tracks of intervals. In statistical applications, representing each nucleotide by a stochastic variable has many advantages [21]. In both types of common mathematical representations, time series analysis is readily applicable [21].

The mathematical representations have obvious analogs for computational representation. Each interval in an interval track is typically represented by a pair of start and end coordinates, usually accompanied by a sequence id tag, representing e.g. the chromosome where the interval is found. An interval may e.g. be represented by the following triplet: (‘chr1’, 123, 456). The data structure analog of the number sequence representation is as one, or several, arrays of numbers (in the case of value data), or bits (in the case of interval data). Representing interval tracks this way requires a bit to be set to 1 for all nucleotides included in the track, and 0 for all the others.

Finding a uniform way of representing all the different types of track datasets is intuitively very useful, both for theoretical uses, as well as for practical applications in e.g. software tools. Unified representations allow easier integration of diverse datasets and tools, as well as opening opportunities for new types of questions to be answered based on the integration of new combinations of datasets. Network-based representation is in its very nature a very uniform type of representation, capable of integrating data of diverse types, as discussed in section 1.2.2.1. Allowing similar integration capabilities of track data would open up new analysis opportunities, making use of the sequential nature of such data in new ways.

As array-based representation could be used for both value- and interval-
type data, one may suspect such representation to be suitable as a unified representation of track data in all its forms. A particularly useful feature of such representations is that a set of tracks could be easily represented as a binary matrix, as done in e.g. [116]. However, such solutions have several drawbacks. First, such representations cannot uniquely distinguish between all cases of interval tracks. One example is that two intervals that are immediately following one another appears exactly the same as one long interval of the combined length. Another example of such ambiguity is in the case of overlapping intervals in the same track. Such cases may be represented by an array of integers, where each integer represents the number of intervals covering each nucleotide. With such representation, however, a case where the end of one interval overlaps with the start of another may have the exact same representation as another case where a small interval overlaps in the middle of a longer one. The second drawback of array representation is the required storage space. Representing the whole human genome requires over 3 billion numbers, thus requiring at least 3GB for a track of intervals that can overlap internally.

Basing a uniform data structure for tracks on interval representations is a better choice, as interval representation does not entail the same problems of ambiguity as with array representation. In order to cater for value-type tracks, one may add a fourth element to the interval triplet, containing the associated value. However, another type of storage drawback becomes apparent if one tries to represent genome-wide value-type data at the nucleotide scale in this manner. Each nucleotide would then be represented by a structure of four elements, e.g. (‘chr1’, 123, 124, 0.5), effectively quadrupling the storage requirement of array-based representation. Also, a drawback with integer-type representation is the need for an indexing scheme in order to locate intervals in a specific region of the genome in a large dataset.

As both interval-based and array-based representations have their advantages and drawbacks, a way towards a more powerful uniform representation scheme may be to combine the power the two approaches, while minimizing, or removing the drawbacks.

**Limitations of track representation** Even though track-type representations of sequence-related datasets are much used and have many advantages, this type of representation entails several limitations. The first limitation is the need for specifying a reference genome to act as the coordinate system. Even though reference genomes provide a way to more easily compare different datasets, the “averaging” nature of a reference genome may lead to neither of the tracks being correctly represented. Also, in the case of genomes that have not yet been assembled, the lack of a reference se-
quence hinders many types of analyses dependent on track-type representation. Another problem arises in the area of “comparative genomics”, where one compares datasets or sequences related to different genomes. Even though one may select one of them as the reference and align the sequences of the other genome to this reference, this adds a layer of complexity not present in single-genome analysis.

Another major limitation of track data is the static nature of the representation structure. Network-type data have been much used in systems biology for modeling dynamic behavior [74], but little has been done in this direction for track-type representations, at least as far as the author of this thesis is aware. As the nature of the cellular environment of the DNA is obviously highly dynamic, this may prove to be a major limitation of track representations. However, dynamic data require technologies capable of providing information of a dynamic nature. Such technologies are not yet common, at least at the nucleotide-level scale.

From a statistical point of view, the representation of exact base pair positions may be problematic, as any positional uncertainty is removed. In the case of high-throughput sequencing data, the union of all reads will work as an approximation to the underlying distribution, reducing the problem. Much track data is also of an exact nature. However, when dealing with simplified data such as ChIP-seq peaks, some measure of the uncertainty of the start and end positions may increase the robustness of statistical analyses.

Static track-type datasets based on reference genome coordinates have, however, in the later years provided new insights in cellular processes at an unprecedented scale. The use of datasets stemming from next generation sequencing techniques has been especially important in this direction. Better unification of the mathematical and computational representations of such datasets will improve data integration and increase usefulness of analysis tools. Of particular importance is the introduction of connectivity-type data to track representation. Connectivity-type data include those stemming from paired-end sequencing and chromosome conformation capture techniques. To quote a recent article in Nature Reviews Genetics: “Additional research is needed to find the most intuitive ways of analysing and visualizing this type of data” [117].

The next chapter will introduce different techniques and methodologies for integrated analysis of genome-scale data in the form of genomic tracks.
1.3 Data analysis

The term “data analysis” was introduced in 1962 by the American statistician John W. Tukey in a seminal paper titled “The Future of Data Analysis” [118]. The paper has been enormously influential, and the term “data analysis” has since largely replaced the term “applied statistics”. The paper was written in a historical context where statistics for many years had been thought of as a mostly theoretical discipline, a subfield of mathematics, and much published work focused on mathematical derivations and proofs [119]. In the first half of the century, the field of statistics had witnessed a fierce controversy between the “significance testing” approach of R. A. Fisher and the “hypothesis testing” approach by J. Neyman and E. S. Pearson. The approach of Fisher focused on generalizations of experimental results, considering only one null hypothesis, while the later Neyman-Pearson approach aimed at deciding between two alternative hypotheses, but without epistemological interpretations. In the 1940s, textbooks in fields like psychology started to teach hybridized, and logically inconsistent, variants of these methods, which has evolved into modern hypothesis testing techniques used, often uncritically, in most fields of science [120, 121]. In the words of Tukey: “Anything to which a confirmatory procedure was not explicitly attached was decried as ‘mere descriptive statistics’, no matter how much we had learned from it” [122, p. vii]. It was in this climate that Tukey argued a refocus: “Data analysis is a larger and more varied field than inference, or incisive procedures, or allocation” [118, p. 2]. By “inference” he meant validation (specifically by significance testing), by “incisive procedures” he meant clever approaches to perceiving data, and by “allocation” he meant something along the lines of guidelines for experimental design [21]. Although leaving a place for inferential methods, Tukey argued specifically the importance of “indications”, i.e. the suggestions inherent in the data: “the main need in this field is for techniques of indication, for ways to allow the data to express their apparent character. The need for significance and confidence procedures will only begin to arise as respectable indication procedures come into steady use” [118, p. 45]. The approach was a fundamentally pragmatic one. Tukey argued that data analysis “must seek for scope and usefulness, rather than security” [118, p. 6], that one should seek out new problems, more realistic frameworks, try out new techniques for summarizing data, and use sound judgment, rather than aiming at optimal solutions: “Far better an approximate answer to the right question, which is often vague, than an exact answer to the wrong question, which can always be made more precise” [118, p. 13].

---

4The terms “significance testing” and “hypothesis testing” will be used interchangeably in the rest of the thesis, denoting the modern hybrid variant unless otherwise noted.
Such pragmatism is also obvious in his partial definition of “data analysis”:

procedures for analyzing data, techniques for interpreting the results of such procedures, ways of planning the gathering of data to make its analysis easier, more precise or more accurate, and all the machinery and result of (mathematical) statistics which apply to analyzing data. [118, p. 2]

In this thesis, the term *data analysis* will be used similarly as described by Tukey, but a significant part will be omitted in order to focus the discussion; methods cleaning, transforming and characterizing data will (mostly) not be described. This is in contrast to much of the bioinformatics literature, where exactly such methods are often in focus. As an example, a recent paper describing a protocol for constructing Illumina sequencing libraries uses the term *data analysis* for everything that happens after the raw sequences has been produced by the Illumina sequencing technology [123]. This includes assembling the sequence, calling SNPs, finding ChIP-seq peaks, and mapping RNA-seq reads to genes. The focus in the paper is on checking the quality of the data, reducing the noise stemming from technological or experimental artifacts, removing bias, transforming data into a proper format, and mapping it to a reference genome. In this thesis, such parts, which may be termed “initial data analysis”, will be largely skipped, focusing instead on all analyses that comes after such steps have been carried out and are aimed at answering the research question under study. Specifically, this section will focus on different techniques and tools for analyzing genomic data represented as tracks (see section 1.2.2.4). These tools will be divided into two main categories, following two terms established by Tukey: exploratory and confirmatory analysis. Commercial tools will not be included in the discussion, as they are usually not published in peer-reviewed journals and are often not readily inspectable.

### 1.3.1 Exploratory data analysis

Although John W. Tukey is the inventor of the term “exploratory data analysis” (EDA), he did not provide a single, exact definition of this term. Instead, he described it in several ways [124], including: “1. it is an attitude, AND 2. a flexibility AND 3. some graph paper (or transparencies, or both)” [125]. In 1977 he published an influential textbook bearing the name of the method, “Exploratory Data Analysis”, containing an overview of a range of methods of data exploration, most of which were inherently visual techniques. Since the days of Tukey, the computer revolution has allowed explorative methods based on a more algorithmic nature. Exploratory analysis is often regarded
as model-free, but one may argue that simple models have been a part of EDA since the beginnings [126]. Here, model-based exploratory methodologies will be included in the discussion of exploratory data analysis.

This section will describe a selection of visual and algorithmic methodologies and tools for the explorative analysis of genomic track data. This discussion is not meant to form a complete catalogue of track analysis techniques, but is rather meant to work as an introduction to some typical directions of integrated track analysis in the literature. As the main focus is on integrational aspects, techniques for single-track summaries are not directly included. First, a range of genome browsers will be introduced. Next, different types of algorithm-based tools and methodologies will be presented, divided into three categories: 1) pattern discovery techniques; 2) techniques for comparing tracks; and 3) techniques for discovering functional associations.

### 1.3.1.1 Genome browsers

For Tukey, visual analysis held an especially important position:

> If one technique of data analysis were to be exalted above all others for its ability to be revealing to the mind (...), there is little doubt which one would be chosen. The simple graph has brought more information to the data analysts’s mind than any other device.” [118, p. 49]

Such focus on visual analysis is also prevalent in the field of bioinformatics. A reason for this may be the strong precedence for visual analysis witnessed in the history of biology, from detailed visual description of different species, through microscopics, via dye-terminator Sanger sequencing, to microarray coloration (see section 1.1.2). It may be that the visual focus of the training of biologists favors analysis techniques of an inherently visual nature. For instance, a tutorial for analyzing ChIP-seq transcription factor data [127] recommends the visualization of the ChIP-seq data using a genome browser: “Interesting patterns may emerge by simply eye balling the data. These patterns may create new hypotheses and suggest future research directions.”

**UCSC, ENSEMBL, and GBrowse** The need for visualizing genomic data-sets followed with the completion of draft genome sequences of diverse organisms, most notably the human genome. The UCSC Genome Browser (already introduced in section 1.2.2.4) is one example of web-based software
created especially for this purpose (see Figure 1.6 for an example visualization). Other early genomic browsers include the MapViewer software embedded with the web server of the National Center for Biotechnology Information (NCBI) [128]. NCBI is a U.S. government-funded national resource for molecular biology, serving important databases and resources such as PubMed, GenBank [69], Online Mendelian Inheritance in Man (OMIM), and RefSeq [129]. The European counterpart of NCBI, ENSEMBL (a joint project between the intergovernmental European Molecular Biology Laboratory (EMBL) and the Wellcome Trust Sanger Institute), also developed a competing genome browser software in connection to the databases hosted from their web site [130]. Another of the early genome browser software solutions is the Generic Genome Browser (GBrowse) system, developed by the Generic Model Organism System Database Project (GMOD) [131]. The GBrowse system was developed in order to provide a portable, flexible and simple open source visualization system fit for deployment on web sites specific for model organism genomes. Common to all web-based genome browsers is the ability for integrated visualization of tracks of different types, often connected to a large set of datasets hosted on the server, in addition to tools for searching, retrieving and uploading data [132].

Savant, IGV, and other recent developments  In addition to web-based genome browsers, a range of different systems are developed as stand-alone software packages to be run on the users own computers, such as the Savant Genome Browser [133], or the Integrative genomics viewer (IGV) [134]. These last two are also based upon multi-resolution data models that are able to provide access to data at multiple resolutions, allowing easy scalability to gigabyte-sized datasets. A third type of genome browsers, so-called synteny viewers, are designed especially for displaying genome alignments within or between species [132]. With the recent possibilities of analyzing the same type of tracks over different contexts, such as cell types, tissue types, or clinical samples, several genome browsers have added visualization of metadata, with the additional ability to organize or sort tracks according to metadata parameters. Genome browsers with such metadata support include the aforementioned IGV system, as well as the UCSC Cancer Genomics Browser [135] and the Human Epigenome Browser at Washington University [136].

Circos  All of the aforementioned genome browsers base the visualization upon viewing the genome reference sequence as a line, or as a set of lines. This works very well in most cases, but is inconvenient for visualizing con-
nections between different parts of the genome, such as genomic rearrangements, or for use in comparative genomics. Circos [137] is a specialized tool for track visualization that is based upon viewing the genome sequence as a circle. It is a command-line based tool that “compiles” images based upon configuration files, and is thus not an interactive genome browser, as the others described here. It has, however, been much used, especially for creating impressive figures for use in publications.

**Galaxy Track Browser** Even though some of the genome browsers mentioned here include a few analysis tools, such as simple statistics, most of them are used simply to create visual summaries of the data, or to correlate tracks by visual comparison. The Galaxy Track Browser is aiming at being a truly visual analysis system [138]. The system also provides efficient performance by the use of a multi-resolution data model, as well as supporting sharing and publishing of visualizations in a simple manner. The browser is integrated with the Galaxy platform [139–141], a web-based analysis framework, containing hundreds of bioinformatics tools. The Galaxy Track Browser is unique in enabling its users to apply filters or run a range of different analyses directly from the track visualization screen, updating or creating new tracks “on the fly”.

**Challenges with genome browser-based analysis** Purely visual analysis of genomic tracks and their relationships using genome browsers may be problematic. Although genome browser-based analyses may provide interesting hypotheses, conclusions may be drawn on thin grounds, for several reasons, mostly related to the typical massive sizes of genomic datasets. First, as genome browsers usually only provide comparison for specific regions of the dataset, global patterns may not be observed. Second, generalization from limited parts of the genome may be incorrect, as the global distributions are not taken into account. Third, manual comparison is highly dependent on human pattern-finding abilities, which are error-prone and not open to operationalization.

To remedy such problems, a range of methodologies and tools have been developed for comparing genomic tracks at a global level, typically providing global-scale summaries of interesting aspects of the data. One may, however, argue that the massive size, and complexity, of genomic datasets makes them extremely difficult to interpret by humans, even if appropriate summaries are presented. Most of the tools and techniques presented in the following thus include some sort of algorithmic analysis backbone, helping the user to interpret their datasets. Software tools presented also differ in
the amount of visual feedback presented to the user, both in terms of graphical user interfaces (GUI) and in the summarization of results as graphs and figures.

1.3.1.2 Pattern discovery

Heatmap approaches with seqMiner and Cistrome With the advent of the ChIP-seq technology, the characterization of epigenetic processes, such as chemical modifications of the histone proteins, became possible at unprecedented scales and precisions (see section 1.2.2.3). As the different epigenetic markers seemed to act in concert, much research focused on the deciphering of the “epigenetic code” in order to better understand the roles of epigenetic markers in e.g. gene regulation [34]. A set of techniques and tools were developed for discovering global patterns created by sets of epigenetic modifications in particular regions of the genome, such as around the TSS of genes. One such tool is seqMiner, a stand-alone GUI-based software system [142] focusing on integrative analysis of ChIP-seq datasets. The pattern detection is carried out over a user-defined set of genome regions, the “reference coordinates”. All reference regions are split into a user-defined number of sub-bins, and for each sub-bin, the maximal number of overlapping reads of an input ChIP-seq dataset is calculated. Such “density arrays” are created for all combinations of reference regions and input ChIP-seq datasets, optionally normalized, clustered by k-means clustering, and presented to the user as heatmaps. Global patterns of epigenetic markers across the reference regions (e.g. around the TSS of genes) can then be easily identified by the user. Similar functionality is available using the “Heatmap” tool from the Cistrome system [143], which is based on the open source Galaxy framework [139–141], and includes a range of ChIP-chip and ChIP-seq specific tools. Output from the Cistrome “Heatmap” tool is shown in Figure 1.7. The Spark system [144] improves on this clustering approach by allowing the user to dynamically guide the analysis by splitting interesting clusters into sub-clusters, thus providing better support for interactive explorative analysis.

ChromaSig All of the methods described above are based on user-selected analysis regions. This may be acceptable for focused analyses, but does not allow global screening for patterns. The ChromaSig method [145] automatically screens the genome to find loci of length 4 kbps that are enriched in chromatin modifications. From these loci, seed motifs are created by combining ~ 20 similar loci, with similarity defined by a Euclidean distance measure. For each seed motif, all loci are considered for alignment with the mo-
tif on the basis of a likelihood maximization calculation taking, among other things, the probability of aligning with a null background into account. In the alignment step, different offsets and orientations of the loci windows are tried. The ChromaSig method has been configured for use with both ChIP-chip and ChIP-seq datasets.

**Segmentation approaches**  Another approach to pattern discovery based on genomic tracks is to carry out a segmentation of the whole genome, dividing the genome into contiguous intervals, each associated with a particular pattern of chromatin signals. The HMMSeq tool [146] models the genome by a fully connected Hidden Markov Model (HMM), trained on a set of input data to classify each consecutive region into one of several states. The software includes wavelet smoothing in order to support large-scale datasets. Another approach, termed “Spatial Clustering”, trains a two-level HMM model in order to capture higher-order structures [147]. Both of these approaches learn the HMM model based on estimated normal multivariate distributions across all tracks. The ChromHMM software [148] (described more in detail in [149]) instead explicitly models the presence/absence frequency of each epigenetic mark by Bernoulli random variables, which improves robustness, avoids parametric assumptions, and are more directly interpretable. The ChromHMM approach was used for the mapping of promoters and enhancers in nine human cell types [102], as referenced above in section 1.2.2.3.

All the segmentation approaches described above makes use of Hidden Markov Models. Another approach is used by the recent Segway tool [150],
which instead models the chromatin data by Dynamic Bayesian networks (DBNs), which is able to represent more complex relationships among variables at nearby positions in the genome, in addition to allowing easy incorporation of various types of prior knowledge. Compared to ChromHMM, Segway produced more precise predictions of known elements, at a finer resolution, and with a better handling of missing data.

A recent semi-supervised classification approach made use of random forest classifiers to predict regions of active and inactive binding, with extremely high or low degrees of co-binding, and regulatory modules proximal or distal to genes [151]. The classifications were based on transcription factor binding, chromatin, gene annotation and expression data from the ENCODE project [79].

A very different segmentation approach was used in a recent study, where chromatin data were clustered in two dimensions with the use of self-organizing maps (SOM) [152]. While such views obscures the detailed positioning of the patterns along the reference genome, it allows simple visual comparison of global epigenetic profiles across e.g. different cell types (see Figure 1.8 for an example).

1.3.1.3 Track comparison

**Correlative approaches with seqMiner and Cistrome** While the pattern discovery techniques introduced in the previous sections focused on the analysis of similarity across tracks in specific regions, a set of other techniques focus on finding similarities (and dissimilarities) between genomic tracks at a global level. An example of this is found in functionality included with the previously introduced seqMiner system [142]. Here, two ChIP-seq tracks are compared across a set of user-defined regions. In each region, the total number of reads for both tracks is calculated, and the resulting counts are visualized as scatterplots for visual identification of track correlations. The Cistrome tool suite [143], also introduced previously, contains similar scatter-plot functionality, but with several improvements: calculation of Pearson correlation coefficients, visualization of trends in the scatterplots, and the visualization of results as a matrix of pair-wise combinations of a set of genomic tracks (see Figure 1.9). In addition, Cistrome includes functionality for drawing Venn diagrams of over the number of unique and overlapping regions between different tracks.

**Biplots** Another visual approach for comparing was presented by a paper on statistical analysis of tracks from the pilot phase of the ENCODE project [116]. Here, the interrelations between ChIP-chip tracks of histone
modifications and transcription factor binding sites were presented visually by biplots over a set of 5-kbps non-overlapping genomic bins. Biplots is a graphical representation of the datasets according to the two first principal components of the data, providing a powerful way to visualize both correlations among the tracks and similarities between the profiles of the genomic bins. In the same study, a matrix-based correlation technique, similar to the one described above for the Cistrome tool, was also presented.

**BEDTools, MULTOVBL, and Segtools** The command-line-based BEDTools library [155] contains functionality for assessing overlap between pairs of genomic tracks with interval data. The library also allows the calculation of the coverage (in bps) of one interval track relative to another, as well as the extraction of the closest regions of one track from the features of another. In addition to this, BEDTools provide functionality for interval operations such as intersection and union, as well as support for tracks with paired-end
sequencing data. Similar interval operations have been included with the Galaxy web-based framework [139–141] since the first version of the system. BEDTools was created partly to provide a faster and more flexible interface to such operations, focused on users with experience with programming or working on the command line. The popularity of the BEDTools approach is evident in the appearance of competitor libraries providing even faster performance [156, 157]. The MULTOVL tool [158] provides fast overlap calculation in the vein of BEDTools, but supports more than two tracks. The Segtools library [159] provides a set of operations and comparative plots between segmentation tracks, i.e. interval tracks where each region is annotated with a label. The tool distinguishes between segmentations, where the intervals are not allowed to overlap, and associations, where overlaps are allowed, and different operations are provided depending on which type of track is used. Association tracks are in essence an alternative representation a set of simple interval tracks as a single track. Segmentation tracks, which is the main focus of the library, can be created from a set of interval tracks by “flattening” a set of interval tracks, labeling each interval with the names of the combination of tracks that overlap at that interval. Segtool may thus be viewed as a library for comparing multiple interval tracks.

**Mutation rate co-variation** Another approach to track comparison was used in a paper analyzing human mutation rate co-variation [160]. In the approach, the human genome was divided into 1 Mbp windows, and the rates of four different types of mutations were calculated in each window for two
human-specific sub-genomes, as compared to orangutans. These mutation rates were analyzed by principal component analysis (PCA) in order to characterize the underlying components of the mutation rate co-variation. Subsequently, canonical component analysis (CCA) was carried out, relating the rates of different types of mutations with a set of genomic tracks that were found to provide plausible explanations to the mutation rate co-variation. Non-linear version of these multivariate techniques were also carried out on the data. All analysis types are available in a set of Galaxy tools included with the standard Galaxy installation [139–141].

1.3.1.4 Functional associations

DAVID, GREAT, and PRISM  In 2000, the Gene Ontology Consortium presented three independent ontologies for genes, classifying genes by the use of structured, controlled vocabularies focused on respectively biological processes, molecular function, and cellular components, a knowledge database that has been continually expanded ever since. When presented with a list of genes, such as differentially expressed genes resulting from a microarray experiment, a common approach for probing the function of the genes has long been to find enriched gene ontology (GO) terms by submitting the gene list to software tools such as DAVID [67]. The Genomic Regions Enrichment of Annotations Tool (GREAT) [161] improves on such gene-list-based approaches by allowing to search for enriched GO terms based on any set of genomic regions, not only genes. The basis of the approach is to associate the genome regions to related genes by applying a set of simple rules that approximates binding site regions proximal or distal to genes. The main focus of GREAT is on ChIP-seq datasets of transcription factor binding, but the tool may also be used for other datasets relevant for gene regulation, such as epigenetic markers. The recently published PRISM system enhances the GREAT analysis by predicting transcription factor binding sites (TFBS) for a range of TFs using a Monte Carlo based null model to predict binding sites that were more conserved than expected [162]. All predicted TFs were run through the GREAT tool for functional annotation.

Cistrome (CEAS) and EpiExplorer  A different approach is constituted by software systems that associate genomic track data from the user with known, functionally related tracks already incorporated with the system. The cis-regulatory element annotation system (CEAS) [163] is included with the aforementioned Cistrome system [143] and provides a simple application of such associations. CEAS automatically characterizes input tracks according to overlap with exons, introns, and cis-regulatory modules, in addition


to providing plots of the amount of inter-species conservation associated with the nucleotides covered by the dataset. A more expansive approach is provided with EpiExplorer [164]. By the use of a text-based search engine, EpiExplorer calculates the amount of overlap between the user-specified track and a range of epigenetic tracks incorporated with the system, including histone modifications, DNaseI hypersensitivity and transcription factor binding sites. By comparing with a uniformly randomized control track, the system provides a simple way of comparing overlap with chance. The paper on the system does not, however, regard this simple comparison with a control set as a substitution for confirmatory analysis, but only as a simple indication for use in hypothesis generation. EpiExplorer includes a set of visualization options, as well as the calculation of traditional gene-centric GO-associations. In addition, the system contains functionality for dynamic, interactive filtering of input regions according to user-defined criteria, including overlap with other tracks, thus providing a simple way for users to pinpoint interesting hotspot regions.

1.3.2 Confirmatory data analysis

Some researchers argue that biology is too complex to be based on hypotheses and models and that biological research should be data-driven [165, 166]. This entails that one carries out correlative analyses on massive data sets, searching for interesting patterns, as exemplified with the tools or techniques of exploratory data analysis as described in the last section (1.3.1). There is currently much effort poured into the generation of public, genome-wide datasets by large-scale collaborations, such as the ENCODE [79] and the NIH Roadmap Epigenomic [167] projects. Other efforts include the creation of massive databases of biological samples, or knowledge-based information of different kinds. All these data-driven enterprises occur in the name of science, with increased knowledge as the ulterior goal. The implications of the surge of data collection not driven by hypotheses may be a fundamental shift in research strategies. A recent editorial in Future Microbiology promotes data-driven science, arguing that “the most important findings have been driven by projects without real hypotheses and instead were discovered base on real serendipity” [166].

Data-driven science is based on inductive reasoning, *i.e.* inference from observations to theories. The hypothetico-deductive method, on the other hand, is based on logical deduction and is often seen as an opposition to data-driven science [168]. Others argue that data-driven science, on the other hand, complements the hypothetico-deductive method. In a short paper in The American Statistician in 1980, John W. Tukey argued for the need of
both exploratory and confirmatory analysis [125]. Tukey defined the “confirmatory paradigm” as a series of steps similar to the following steps of the hypothetico-deductive method [169]: 1) gather observations; 2) formulate a hypothesis that would account for the observations; 3) deduce observational predictions from the hypotheses; and 4) test the predictions. Tukey argues that the stages of the confirmatory paradigm corresponding to stage 2 and 3 in the hypothetico-deductive method in practice are carried out iteratively. One starts with an “idea of a question”, and refines the question and the associated experimental design iteratively by exploration of (past)
data. In addition, Tukey also holds this part of the scientific process in especially high regard: “Finding the question is often more important than finding the answer” [125]. Similar arguments about the importance of exploratory, or descriptive, studies as part of the hypothesis-driven scientific process are found in more recent discussions by researchers in life science [170–172]. The editorial in Future Microbiology that was referenced above argues, on the other hand, against hypothesis-driven research on the basis that research “assumes a bias, as it favors the proposed hypothesis” [166]. Also, granting research proposals based on hypotheses is found too slow a process to be able to keep up with the “acceleration of data acquisition” [166].

The questions regarding philosophy of science raised by the technological revolution in molecular biology has, as referenced above, been discussed somewhat in the literature. One may, however, wonder whether the more fundamental scientific consequences of the “data deluge” do not receive the amount of attention they deserve. In a commentary from 2001, which has not lost its actuality, Prof. John F. Allen describes his view on the situation in this jokingly manner: “It is as if we are too busy with the all-important task of generating more data, and have come to view thinking as a distracting waste of time” [168]. There may, joke aside, be some truth in the observation that the scientific process is not clearly defined in the field of bioinformatics. Molecular biology has a history of testing hypotheses by carefully designed experiments, but the same frameworks do not seem to have been built in the field of bioinformatics. It is often not evident whether a bioinformatics approach is meant to be explorative or confirmatory. One may simply regard approaches that make use of hypothesis tests as confirmatory, but this is a simplification. One example is Gene Set Enrichment Analysis (GSEA) [68], which is a commonly used method for finding e.g. differentially expressed genes from microarray experiments. With GSEA, a p-value is calculated for each gene, denoting the probability of the observed difference in gene expression, or differences more extreme, happening at chance. Even though a hypothesis test is carried out, it is the context that defines whether GSEA is used confirmatorily. One may, for example, be interested in finding the genes that are differentially expressed between microarray experiments divided into two classes according to some phenotype. This is a case of exploratory analysis, but guided by the null model assumed by the GSEA methodology.

The following description of confirmatory analysis techniques for genomic track analysis will start by introducing overlap tests, which arguably is the

5It is important to note, however, that one should avoid testing a hypothesis on the same data that was used to suggest the hypothesis, as this can lead to false positive findings
most basic form of confirmatory data analysis. Following this, some techniques of similar kind as GSEA are introduced: methodologies that include hypothesis tests, but often used in exploratory settings. Next, tools and techniques that are confirmatory in nature are discussed. A particular area is then discussed in some detail, namely the comparison of tracks of transcription factor binding sites for co-occurrence.

1.3.2.1 Simple overlap tests

One type of hypothesis test holds a particular position when dealing with the comparison of genomic tracks: the testing of overlap between the intervals of a pair of genomic tracks (see Figure 1.11). To carry out such a test, it is necessary to specify a particular test statistic describing the exact relationship between the genomic tracks that one is interested in. In this case, the test statistic could e.g. be the number of base pairs covered by the overlapping intervals. In addition, a null model needs to be specified, for instance uniform distribution along the genome, often approximated by randomization. The typically most interesting result of the hypothesis test is the \( p\)-value, which is the probability of observing the actual value of the test statistic, or something more extreme, given the null model. If the \( p\)-value is low enough, one may discard the hypothesis represented by the null model. Variations of such overlap tests had been carried out in several studies before the start of this PhD project [116, 173, 174]. Randomization of genomic tracks are possible from the BEDTools library [155], and simple Monte-Carlo based overlap tests are available from some of the competitor libraries described in section 1.3.1.3 [156, 158].

![Overlap test diagram](image)

Figure 1.11: Illustration of hypothesis test of overlap between the intervals of two tracks.
1.3.2.2 On the border between exploratory and confirmatory data analysis

**GenomeRunner**  Given a hypothesis test of overlap between two tracks, along with a large database of interval tracks, a natural extension is a system for systematic hypothesis testing of pairs of tracks. The GenomeRunner stand-alone software tool [175] is exactly such a system. GenomeRunner compares one input track with a range of tracks already incorporated with the tool, typically downloaded from the UCSC Genome Browser database [176]. For each pair of tracks, an enrichment test (*e.g.* an overlap test) is carried out. Three hypothesis tests are developed: one Monte Carlo-based, one analytical, and one Chi-square test. Even though the tracks are compared with hypothesis tests, it is clear that the system is meant for use in an explorative setting, hence the title: “GenomeRunner: automating genome exploration”. This is also evident in the lack of multiple testing correction. The multiple testing problem occurs when a set of hypothesis tests are carried out. If enough hypothesis tests are tried, some are bound to produce p-values below some significance threshold by chance alone. To remedy this situation, one should use some form of multiple testing correction, such as Bonferroni correction or False Discovery Rate (FDR), when applying the hypothesis test in a confirmatory setting [177].

**EpiGRAPH** Another type of hypothesis tests is used for the comparison of two interval tracks, one regarded as the “case” and the other as the “control” data set. The null hypothesis in question is that the two tracks do not differ in their overlap, or enrichment, with other interval tracks. The PinkThing web-based system [178] provides a simple implementation of such statistics. Two input tracks are compared with a segmentation of the genome in the following types of regions: introns, exons, and different categories of closeness to genes. The system counts the number of loci of the two tracks falling in each category of the segmentation and represents the counts as a contingency table. The table is then tested using appropriate hypothesis tests. The EpiGRAPH web server [179] is a more elaborate implementation of the same concept (and was published in 2009, four years before PinkThing). In EpiGRAPH, the input tracks are compared with a range of different tracks incorporated with the system, including a range of epigenomic tracks. The systems supports the calculation of a range of test statistics, stores the intermediate results in a contingency table, and uses either a nonparametric Wilcoxon rank-sum test, or a Fisher’s exact test, to calculate p-values. The system also sports graphical output, as well as a machine learning module for classifying new data as either case or control, based on
track associations learned from the two original input datasets. The Pink-Thing does not include multiple testing correction and is clearly meant for exploratory use. EpiGRAPH, however, includes both Bonferroni and FDR correction. It thus seems fit for use in both exploratory and confirmatory settings.

**EpiCenter** The last of the combined exploratory/confirmatory tools to be described is the EpiCenter system [180]. EpiCenter works directly on mapped reads from ChIP-seq or RNA-seq experiments, and includes methods for read filtering, noise reduction, and normalization of the input data. The aim of the system is to detect genomic regions that differ in the density of epigenetic markers in two input tracks, e.g. derived from two different cell types. Three hypothesis tests are carried out. The first test filters regions according to whether they contain significantly more counts than expected from random Poisson background noise. The second test is a binomial test on whether the Poisson rates of the markers in the two tracks differ, while the third test is a z-test on the assumed Gaussian distribution of the fold change (log2 ratio) between the tracks. As the two last tests have different properties, the results are combined by taking the maximum p-value of the two. EpiCenter includes multiple testing correction, and can thus be used both in exploratory and confirmatory settings, similarly to the GSEA technique described in section 1.3.2.

**1.3.2.3 Confirmatory tools and techniques**

**GenometriCorr** [181] is a library that provides a set of other hypothesis tests for interval track comparison. The included tests are as follows: 1) a relative distance test for measuring whether the intervals of two tracks are closer together or further apart than expected by chance, where the exact distances are not important; 2) an absolute distance test, for cases where the exact distances are important; 3) a projection test, measuring whether the midpoints of the intervals of one track is located inside the intervals of the other track, more than expected by chance; and 4) the Jaccard test, which is a variant of the overlap test. Although the test statistics are rather well thought out, the null models are still based on uniform randomization. As further discussed in the next section, uniform null models are often too simplistic for complex genomic data. GenometriCorr is available as a R package.

---

6Note that multiple testing correction can be used also in exploratory data analysis. It is the research setting that determines whether the analysis should be considered exploratory or confirmatory.
[182] and also as a Galaxy tool [139–141], but with limited graphical user interface (GUI).

**Subsampling methods of ENCODE**  A general subsampling method for handling hypothesis tests on genomic annotation tracks was presented in the supplementary information to the Nature paper publishing the results of the ENCODE pilot phase [80]. The method was later expanded, given a thorough statistical treatment, and published separately in 2010 [183]. The paper still stands as the most thorough treatment given to confirmatory analysis of genomic tracks, and is the version of the subsampling method that will be presented here. The paper acknowledges that the main problem of confirmatory analysis (or validating statistics, which is the term used in the paper) in the field of bioinformatics is usually the inadequacy of the null model to describe the “real” null distribution. Most existing methods at the time of writing (as well as most of the techniques described above) assumed some form of uniform or independent placement of genomic elements on the genome (as done in *e.g.* [173, 174]). An assumption of uniformity or independence ignores the natural tendency of many genomic elements to occur in clusters along the genome. Even more problematic than such clustering is the fact that the genome sequences, and thus features described in relation to the sequences, are products of random evolution, a process to complex to describe in any usable model, at least in the foreseeable future. Although this is difficult to prove, experience shows that using too simple null models often lead to underestimating the p-values, potentially giving rise to type I errors (incorrectly supporting the alternative hypothesis) 7. The methods described in the paper avoids the specification a null model based on randomization, as is often done. Instead, the null model is based on subsampling, which entails repeated sampling from the actual data. Subsampling is a variation of bootstrapping [184], with the distinction that the sampling distribution is approximated by drawing a smaller number of observations than in the original dataset [185].

In the paper describing the subsampling methods [183] it is recognized that genomic sequences are not usually stationary. A stationary process here means that the joint probability distribution of the genomic track or track relation under study, as defined over a region of subsequent base pairs, is constant along the genome, for any region size. A recursive segmentation algorithm is introduced, dividing a genome sequence such that each region is approximately piecewise stationary, *i.e.* where the joint probability distribution of the genomic track or track relation in each region is relatively

---

7A manuscript discussing these aspects are currently in submission by some of the co-authors of **Paper 1**
constant. After the segmentation, a stratified block subsampling algorithm is carried out, producing subsamples of the global track from the concatenation of a set of smaller samples, one for each of the piecewise stationary regions. The length of each of the small samples is selected in proportion to the length of the piecewise stationary region from which it is sampled. The distribution of subsamples is then used to calculate Z-scores and p-values. The paper proves that, under a set of relatively weak assumptions, the subsampling algorithm provides a conservative estimation of p-values. The paper also proposes a method for deciding the size of the subsamples. The subsampling methods were found by simulation to work better than Monte Carlo simulation using simple start-site shuffling of track elements, as well as better than ordinary bootstrap.

The subsampling methods where implemented in a Python module for track overlap analysis, named Genome Structure Correction (GSC), available from [186]. The software was used in several of the ENCODE papers.

1.3.2.4 Finding transcription factor binding co-occurrence

The subsampling methods of [183] are general solutions for finding relations between genomic annotation tracks, and can be used for a range of different problems, several of which are listed in the paper. One of the problems mentioned is the problem of finding regulatory cooperation between transcription factors proteins (TFs). The methodological literature on confirmatory analysis of genomic track relations is sparse, and much of it has focused on this exact problem. Basically, one carries out ChIP-seq experiments with transcription factors as antibodies and process the results in order to pinpoint hotspots, or “peaks”, along the genome. Such peaks identify regions where a specific TF binds to the DNA, at a transcription factor binding site (TFBS), possibly participating in the regulation of the transcription of nearby genes. An interesting phenomenon is transcriptional synergy, i.e. that the regulatory effect of multiple transcription factors working together is greater than the sum of the effect of each TF individually. One assumption in this kind of analysis is that TFs are likely to cooperate if their binding sites are often located close together, where often is operationalized as more than expected by chance, as described by a particular null model. As the ChIP-seq peak regions are larger than the TFBS (typically more than 100 bps as compared to a typical 6-12 bps for the TFBS), one often detects closely located TFBS by detecting overlapping ChIP-seq peak regions [14].

The importance of null models for being able to describe TFBS clustering was discussed in [187]. The paper reviewed two previously described null models that were based on randomization of genomic locations, either
uniformly randomized along the genome, or drawn uniformly from the joint pool of all locations found in all tracks (i.e. a permutation). Both methods supported more than two TFs in the same analysis. These null models were compared to a novel null model where the locations of all TFBS except those of the target TF were randomized out of the joint pool. The novel null model was found to improve on the simpler ones. In [188], the above-mentioned method was reviewed, together with other efforts using respectively the hypergeometric test, the chi-square test and a log-linear model on a contingency table of genomic regions. In [189], two novel null models were introduced, both for pair-wise analysis. The first assigns genomic locations out of the joint pool of locations, as previously described, but this time for each TF independently. The second method is a hybrid method of the permutation and independent drawing methods. The independent method was found to be unusable, while the hybrid method was found preferable for pair-wise analysis over the permutation method. For multiple TFs, the permutation method was found to be effective.

Instead of the traditional parametric methods described in the previous paragraph, [190] propose to carry out a set of hypothesis tests, one for each ChIP-seq peak of the target track. Each hypothesis test compares a ChIP-seq peak of the target track with the nearest peak of the other track using a test statistic able to describe both proximity and overlap of segments. The p-values are calculated “exactly”, that is, using combinatorial calculations based on all possible peak locations. The distribution of all p-values is then compared to a uniform p-value distribution to find co-occurrence. The paper erroneously claims that the method does not have any assumptions on the distribution of the peaks. However, the “exact” p-value calculations assume uniform distribution of peaks, as usual. The method is, however, asymmetric and also robust against global distribution biases, such as the average length of ChIP-seq peaks (see next paragraph).

All of the above methods are, in comparison to the subsampling methods of [183], rather simplistic. The detailed assumptions or the properties of the null distributions are not discussed in much detail. Also, validation is typically done using only a couple of simulated and real-life examples.

Other, more thorough approaches have also been published. The main ENCODE paper from 2012 [79] describes pairwise co-localization of 114 transcription factors for a range of cell lines, using the subsampling methodology from [183]. Another ENCODE paper from the same Nature issue, however, highlights four problems with such methods for assessing global, pair-wise co-localization (see supplementary information to [114]): 1) binary treatment of binding events is highly dependent on thresholds of ChIP-seq peak calling and peak overlap determination; 2) cooperation between TFs is of-
ten context-specific, which is difficult to ascertain in genome-wide analyses; 3) pair-wise analyses are not able to reveal higher-order dependencies between TFs; and 4) most global pairwise co-association statistics are symmetric, while TF cooperation is asymmetric. To meet these concerns, a machine learning methodology was developed, learning higher-order, context-specific and asymmetric associations between transcription factors using a random forest based decision tree algorithm. The contexts used for learning corresponded to a particular TF for a particular cell line, and for each context, all other TFs were assessed for co-operability. The method is thus predictive, but the authors plan to improve the method with some form for confirmatory analysis.

Of interest is also a paper using a multivariate Hawkes process model to describe how the binding occurrences of one TF is affected by the binding of another, using a likelihood ratio hypothesis tests of pairwise independence [191]. The analysis is limited to pair-wise associations, but as it is multivariate, all pair-wise associations are handled at the same time. In addition, the effect of transcriptional synergy is found and plotted as functions of the distance between the binding occurrences, for each pair of TFs (see Figure 1.12). The associations found are also asymmetric.

### 1.4 Toward integrated data analysis

The incredible technological breakthroughs witnessed during the last 50 years in molecular biology have given rise to a field bustling with massive genomic datasets in need of analysis. Much focus has been given to the practical issues of data storage and computational infrastructure in order to handle the large amounts of data [192]. Such concerns are, however problematic, still basically technical issues. A different type of issues confronts developers of methodologies and tools for data analysis: how should one represent the data, both computationally, and also mathematically. The genomic track model has emerged as a favorite representational paradigm for next-generation sequencing data, as well as other sequence-related datasets. Many methodologies are limited to handling only interval-based track representations. Other types of track representation, involving *e.g.* associated values or labels, are also used, but there is no consensus on a common nomenclature. Each development effort basically needs to define whatever representation needed on an *ad hoc* basis. A range of different file formats for representing tracks are also available, and some are *de facto* standards, but also in this case are custom extensions and *ad hoc* formats commonly needed.

The ultimate goal of data representation is to provide for data analysis,
Figure 1.12: Estimation of the effect of transcriptional synergy by a multivariate Hawkes model. All pairs of a set of transcription factor are compared based on ChIP-seq data from mouse embryonic stem cells. The height of the plots indicates the likelihood that a transcription factor (TF) co-occurrence happens at the different distances (in base pairs). The plots are created for the forward direction only. The blue areas indicate the pointwise 95% confidence intervals. Each pair of TFs is also tested for co-occurrence using a likelihood ratio hypothesis test (not shown). Figure reprinted from [191] (published under the BioMed Central copyright and license agreement [59]).

of which two major categories can be delineated: exploratory and confirmatory data analysis. A range of different tools and methodologies for data analysis of genomic tracks has been presented. However, almost all of the tools presented here have been published in parallel with the PhD work described in this thesis (after spring 2009). Tools for data analysis of genomic tracks were more scarce at the beginning of the present project, disregarding a number of genome browser systems available. Confirmatory data analysis of genomic tracks, in particular, was not adequately supported. Today, the availability of tools for data analysis of tracks has risen to a more comforting level. The user, however, still faces challenges in choosing methods, understanding the fundamentals of the tools, and making them work in practice. As each tool often has a narrow focus, the user is faced with a continuous task of installing new software or understanding new web interfaces. Workflow systems such as Galaxy [139–141] or Taverna [193] help the user by collecting several independent tools in the same framework. Such systems are, however, more akin to operating systems than really being integrated analysis solutions, in that they allow the flow of data between inde-
pendent tools, but do not in themselves impose structure on the data analysis methodologies. Also, much focus is still on initial data analysis, as compared to exploratory and confirmatory approaches. The statistical analysis system R [182], enhanced with Bioconductor libraries [194], provides similar functionality, but with more focus on exploration and confirmation. The R/Bioconductor system is still, however, largely a collection of independent tools. The system is also not very user-friendly, in practice requiring programming experience to use.

There seems to be a place for general-purpose data analysis systems for genomic tracks that are based on a more structured, integrational approach to the underlying representational aspects. Such analysis systems should also take an effort in adapting the underlying models in order to better describe the biological complexity, rather than basing analysis on oversimplified assumptions. The focus should instead be on the validity of the models: “Intuitively, statistical models chosen to understand biological processes should be fitted to the biological data at hand, rather than the reverse” [191].
Chapter 2

Aims of the study

The aims of this study were threefold:

A) To catalogue and structure the different types of data representations of genomic tracks, developing a unified basis for a range of data analysis approaches. Mathematically, this implied the identification of the fundamental types of information inherent in genomic tracks as a prerequisite for the specification of data representation as mathematical models. Computationally, there were two main aspects: 1) the development of a uniform data structure that was flexible, easily accessible, and compact, but still supported efficient analysis; and 2) the development of a uniform file format that was flexible, easy to understand, and simple to manipulate, along with tools for reading, writing, conversion, and manipulation.

B) To develop a software system for integrative data analysis of genomic tracks, focusing especially on confirmatory data analysis. The system was to be built using the uniform data representation as described in aim A and should to be user-friendly, powerful and flexible. Also, a set of genomic tracks, particularly chromatin-related ones, should be easily incorporated into the system. The software furthermore had to include a set of integrational analyses combining track data of heterogeneous kinds, also allowing for easy extension to other types of analyses. Lastly, the statistics behind the analyses needed to be sound and preferentially be based on biologically relevant models.

C) To illustrate the usability of the system developed under aim B by practical applications on heterogeneous track data. Of particular focus was grand-scale integrational aspects, pushing the computational boundaries of the system as far as practically possible.

As is evident in the aims described above, the aims of this study were focused towards the development of classifications, data representation and
software tools of a general nature. Although the project was originally formulated in terms of integrating specific types of heterogeneous data to form a model of chromatin, the computational background of the author of this thesis quickly led to a refocus towards the development of more general functionality for analysis of genomic tracks.
Chapter 3

Summary of the papers

The aims of the study can be related to the three papers included in the thesis in the following way:

Paper I (published dec. 2010) contributed to the basic development of data representation (aim A) to the extent needed for developing the Genomic HyperBrowser, a software system for integrational data analysis (aim B). Also, several examples of integrated track analysis were presented (aim C).

Paper II (published dec. 2011) extended the data representations developed in Paper I, also introducing a new, unified file format named GTrack (aim A).

Paper III (published jul. 2011) extended the system presented in Paper I with new functionality for grand-scale integrational analysis, providing a thorough example of exploratory data analysis on transcription factor association with disease phenotypes (aim C).

More detailed summaries of the three papers are presented in the following.

3.1 Paper I

Paper I presented the first public version of a web server for general statistical analysis of genomic datasets, the Genomic HyperBrowser. The paper combined: a) a structured methodology for representing and classifying datasets and biological investigations; b) a range of novel descriptive statistics and hypothesis tests; c) an implemented framework for handling data and doing statistical calculations on them; d) a graphical user interface for specifying datasets, analyses and analysis parameters, in addition to pre-
senting the results of the analyses to the user. The aim was thus to both provide a structured methodology and also a usable system for analyzing many kinds of genomic annotation tracks.

The paper introduced the concept of *track types*, modeling the elements of genomic tracks as mathematical objects positioned on a common genome reference line. In *Paper I*, five track types were introduced, but this number was later extended to fifteen in *Paper II*, where the track types also were defined and discussed more closely (changing the terminology somewhat). In *Paper I*, the focus was on hypothesis tests on particular relations between exactly two tracks, *e.g.* whether the segments of two tracks overlap. The corresponding pair of track types (segment-segment in the previous example) was then used to classify each hypothesis test. This classification thus defined a landscape of possible investigation opportunities on the relation of two tracks (all combinations of two track types), some of which were filled by the novel hypothesis tests of *Paper I*, and some left unfilled.

*Paper I* further elucidated common pitfalls and challenges for these kinds of genomic investigations by showing and discussing a set of example analyses. First, the relation between particular histone modifications and gene expression was considered, as investigated by visual inspection in [195]. It was shown that this investigation could be modeled by a relation between tracks of type *points* (histone occupancy) and *valued segments* (genes with expression values). A Kendall’s tau correlation between the number of modified histones in regions around the TSS of genes and the corresponding gene expression values were calculated for 21 histone modifications and 4 region sizes, along with p-values. This illustrated the enhancement of visual analysis with genome-wide hypothesis tests.

Next, local hotspots of murine leukemia virus [109] integration were found, more specifically regions where MLV virus integrated inside 2-kb flanking regions of promoters, more than expected by chance. Some challenges of interpreting results from such local analyses were illustrated with the re-analysis of the repressive histone modification H3K27me3 in mouse chromosome 17, as studied in [196]. A significant overlap was found between H3K27me3-modified histones and SINE repeats on the global level, but not locally along the genome. This result seemed to be an artifact of the fact that the frequency of modified histones changed considerably along the genome. Globally, the frequency distribution was not preserved in the null model, incorrectly leading to a significant result. Locally, however, the frequency of modified histones was relatively constant, thus leading to the loss of significance, a more accurate result. The effect of different preservation and randomization rules for the null model was also studied for this example. It was shown that unrealistically simple null models may lead to false
positives, and that the examination of results obtained for a set of different null models often may contribute important information.

The last example illustrated the inclusion of confounder tracks to an analysis of the melting stability of the DNA double helix. Using data from *Saccharomyces cerevisiae*, it was found that melting fork probabilities \[197\] were higher than expected at the exon boundaries. However, the algorithm for predicting melting forks are highly dependent on the GC content, and it is also known that the GC content is higher inside exons than outside. The analysis was thus repeated, but this time using a null model where the exon boundaries where randomized according to an *intensity track*, in essence limiting the randomization of exon boundaries to positions with similar GC content. The differences in melting fork probabilities were, in the last analysis, not found to be statistically significant.

The Genomic HyperBrowser was developed in order to help reduce the time needed from the inception of a genomic analysis project to the finished results. Often the researcher implements *ad hoc* functionality for the analyses, which takes precious time and has the potential for introducing bugs. By using this system, many common analyses can be done out-of-the-box, and it is easier to write custom extensions of the system when needed, rather than implementing functionality from the ground up. Also, the system aims to provide the researcher with a toolbox and a methodology that makes him aware of analysis assumptions and other choices that may have large consequences for the results.
3.2 Paper II

Paper II further develops the concept of track types introduced in Paper I. It was argued that the different file formats for track data that exist have been needed partly because of underlying informational differences in their structures. The concept of *genomic tracks* is precisely defined, and four *core informational properties* of common types of tracks are delineated: *gaps* between positions, *lengths* of regions, *values* associated to track elements, and *interconnections* between them. From these four properties, fifteen track types were defined, one track type for each combination of the four core properties being defined or not. Figure 3.1 illustrates the one-to-one relation between core informational properties and the track types. The geometrical properties of the different track types are illustrated in Figure 3.2.

![Diagram of Track Types](image)

Figure 3.1: Four-dimensional matrix mapping the relations of the fifteen track types defined in Paper II. Each dimension represents the exclusion (0) or inclusion (1) of one of the four core informational properties: gaps, lengths, values and interconnections. The track type abbreviations in the top-left box are: Genome Partition (GP), Points (P) and Segments (S); in the bottom-left box: Function (F), Step Function (SF), Valued Points (VP) and Valued Segments (VS); in the top-right box: Linked Base Pairs (LBP), Linked Genome Partition (LGP), Linked Points (LP) and Linked Segments (LS); and in the bottom-right box: Linked Function (LF), Linked Step Function (LSF), Linked Valued Points (LVP) and Linked Valued Segments (LVS). The track types with white background (with gaps) are the *sparse* track types, while the ones with grey background (without gaps) are the *dense* track types. See Figure 3.2 for a geometric illustration of the track types. Figure reprinted from Paper II.
The formal model of genomic tracks was defined mathematically: the genomic coordinates forms a discrete metric space, with points defined as single coordinates and segments defined as a subset of 2 or more continuous coordinates. It was further argued that all pairwise combinations of track types define different sets of questions that are meaningful to ask on the particular combination of datasets. The track types were then used to catalogue existing generic file formats for track-type data. The XML-based BioXSD 1.0 format was found to be easily extensible to support all track types, but no existing tabular formats could be easily extended in the similar way. It was thus decided to develop a new tabular format GTrack 1.0 in addition to updating the BioXSD format to version 1.1, in both cases providing support for all track types.

The main concepts behind the GTrack 1.0 specification were laid out as summarized below:

- **A column specification line** (starting with “###”) specifies the names and order of the columns in the file. Four core reserved columns (corresponding to the four core informational properties) and four non-core reserved columns are defined. The track type thus corresponds directly to a combination of the four core columns being defined in the GTrack file. GTrack supports any number of custom columns.

- **Bounding region specification lines** (starting with “#####”) specifies the domain of the track, i.e. the regions over which the track is defined.

- **Header lines** (starting with “##”) contains metadata information on the structural characteristics of the GTrack file. A set of reserved header variables is defined together with default values. GTrack allows the definition of any number of custom header variables.

- GTrack allows structured extensions of the format by specifying *GTrack subtypes* for particular subsets of the complete specification for use in particular tools or research applications. Fully compliant parsers provide automatic validation of GTrack files according to the specified subtype.

Furthermore, the enhancements of BioXSD format 1.1 over the previous version 1.0 were presented. As the BioXSD 1.1 format was developed solely by the second author of Paper II, the details are outside the scope of this PhD thesis and will not be discussed here. At the end of the paper, GTrack-related web tools were presented, along with an overview of the binary storage scheme used internally in the Genomic HyperBrowser.
Figure 3.2: Illustration of the geometric properties of the fifteen track types defined in Paper II. The base line is a genome, or a sequence, on which the tracks are defined. Vertical lines represent positions, while horizontal lines represent the lengths of the track elements. Gaps are thus illustrated by any empty areas between the elements. Values are represented by the height of the vertical lines. Interconnections are represented by arrows, the thickness of which correspond to the weight of the corresponding edge. Figure reprinted from Paper II.
3.3 Paper III

As transcription factors regulate gene expression, they represent potential drug targets. **Paper III** presents an example of large-scale heatmap-based analysis of multiple diseases versus multiple transcription factors, the *differential disease regulome*. In the main “regulome”, a set of predicted target genes for different TF motifs [198] was related to a disease-associated gene lists from the Phenopedia catalog [199]. The main heatmap is created using a scheme pinpointing the deviations of binding events from the average across the set of all diseases-associated gene lists. As the over-/under-representation is dependent on the total set of diseases selected, the heatmap is called differential, *i.e.* one finds differences as compared to all diseases. An alternative scheme, where over-/under-representation is not dependent on the total set of diseases is also implemented. A web interface based on the Google Maps engine visualizes the clustered heatmap with colors representing the Z-score values of over-/under-representation, and marks representing significance.

Using the track type concepts developed in Paper II, the differential disease regulome is an example of the pairwise analysis of two tracks, one of type *valued points* and the other of type *valued segments*, in both cases with categories as values. This representation is informationally identical to a situation of multiple analysis between a set of tracks of type *points* versus a set of tracks of type *segments*. For each pair-wise combination of a point and a segment track, the overlap is calculated genome-wide and used as inputs to the statistical analysis.

The main regulome was scrutinized for interesting clusters, and two particular immune-related clusters was selected for further analysis, the one with motifs related to the NF-κB and Rel transcription factors, the other one related to the IRF family, Stat1 and Cux1 TFs. Similar clusters were also found in an alternative heatmap relating TF binding to gene ontology associations. The gene lists were scrutinized for both clusters in both regulome types, in order to pinpoint which genes were most responsible for the associations. A similar cluster was also found in another ‘regulome’ comparing histone modification ChIP-seq peaks co-localization with TF binding sites.

The particular datasets used for creating the regulome heatmaps of **Paper III** were mainly meant as examples. The main point of the paper was to present a generic large-scale methodology that could be used to combine a variety of genome-wide tracks in a range of settings.
Chapter 4

Discussion

Few directly measurable results have been produced in the present work. The focus has instead been on the development of methodologies, both theoretically, and implemented in software. The biological examples in Paper I function mostly as illustrations of general use cases of the software, focusing specifically on interpretational issues. Paper II is a purely methodological effort. Paper III combines methodology with the integrated analysis of several particular pairs of data sources, resulting in resources that can be used for generating hypotheses. The actual “regulomes” presented in Paper III are, however, more to be considered as proofs of concept, to be improved when better data is available, rather than definite results in themselves. It is the structured integrational aspects of the study that are the most important.

This thesis is thus mostly of a structural nature. Common types of datasets have been scrutinized in order to understand their basic properties. Data analysis techniques have been conceptualized in the same manner, with a focus on the geometric properties of the data and the possible operations that follow from such representations. Another structural aspect relates to the development of system architecture of the Genomic HyperBrowser (Paper I). A principled focus on generality and the loose coupling of the basic modules of the systems have steadily allowed easy addition of new types of functionality not envisioned at the start of the project. A third type of structure is manifest in the development of the GTrack file format (Paper II). A focus on allowing specificity alongside unification has lead to the creation of a set of rules and dependencies that in turn has allowed the use of the format in a range of settings. Lastly, the massive heatmap resources of Paper III were the result of highly structured global-scale analysis of transcription factor binding sites and phenotype-associated gene lists.

This chapter will discuss the results of this study in the following order:
1) the application of the concept of “track types” to common dataset types and data analysis techniques representing the state of the field; 2) the confirmatory and exploratory aspects of the developed methodologies; 3) the biological relevance of the results; 4) different aspects of the user experience of the developed software tools; 5) issues related to the implementation of the software; and 6) comparison of the developed methodologies with existing efforts. The chapter concludes with a discussion of future perspectives.

4.1 Categorizing tracks and data analysis methods by track type

The concept of track types was gradually developed throughout the project period. At the beginning, only three track types were considered: points, segments, and function. When examining the informational aspect of these, it was soon realized that they differed considerably. Points were defined by positions (later changed to gaps between positions), segments were defined by positions and lengths, while functions were defined by value alone. The technical problem of representing data of these three track types in a uniform manner in the prototype code of the Genomic HyperBrowser was basically solved by defining a three-array structure, consisting of start and end positions, and values, respectively. While pondering upon this data structure it was discovered that, logically, seven track types could be represented by the combinatorial use of the three arrays. Two of these, then called marked points and marked segments (later changed to valued points and valued segments), were included with the discussion of Paper I. All of the seven track types were found to describe different types of common datasets. Also, scrutinizing all possible pair-wise combinations of track types proved to be a powerful approach from which a range of relevant, geometrically defined questions were described. Hypothesis tests were designed for a subset of the pair-wise track type combinations, mainly developed by the statisticians associated with the project, and implemented in the system. An overview of implemented analyses can be found in Table A.1. Later, plans were made for a dedicated PhD project focused on the analysis of genomic tracks according to the three-dimensional chromatin structure, making use of tracks created with technologies such as Hi-C and ChIA-PET (see section 1.1.2.5). This required the addition of support for linking track elements. The array-based implementation was thus complemented with an id and edge array pair for describing each link as a pair of from-to IDs. In addition, a weight array was added for edge weights. The addition of linked storage increased the number of track types to fifteen, as illustrated in Figure 3.2. This was
the number of final track types defined in **Paper II**.

The introduction to this thesis provides an overview of typical track-based datasets and data analysis tools and techniques. No common nomenclature for denoting the data models exists. Instead, a range of different terms are used, *e.g.* regions, points, locations, intervals, segments, features, segmentations, annotations, or genomedata. The fifteen track types have been created as a suggestion for a common nomenclature and unified model of genomic track data. To illustrate the power of the track type approach, the data types and methods presented in the introduction have been classified according to track type. Table 4.1 shows an overview of typical tracks, as reviewed in sections 1.1.2 and 1.2.2.3, categorized by track type. In some of the datasets, several values and/or metadata fields are associated with the base pair positions. In those cases, the most important value, or values, have been selected. Note that GTrack supports any number of extra metadata fields. In table 4.2, all tools and methodologies reviewed in section 1.3 have been categorized according to a single or a pair of track types. In some cases, the tools support several different analyses, in which case the most advanced have been chosen for illustrative purposes. The tables illustrate the power of the track type approach for representing a range of heterogeneous genomic tracks, in addition to providing a rich framework for denoting, understanding, and representing analysis methodologies for heterogeneous genomic track data. Note that all datasets in the tables are representable by the GTrack file format and the implemented binary storage scheme presented in **Paper II**.
Table 4.1: Overview of typical tracks, by technology, as reviewed in sections 1.1.2 and 1.2.2.3. The tracks are categorized by corresponding track type. The fifteen track types are illustrated in Figure 3.2. For each linked or valued track type, the corresponding value or weight type is indicated, along with a description of typical content. Other encodings than the ones exemplified here are possible. Note that categories can be used to combine several tracks into one, with the category indicating the name of the track a particular element belongs to.

<table>
<thead>
<tr>
<th>Type of dataset</th>
<th>Track type</th>
<th>Value type</th>
<th>Weight type</th>
<th>Value / edge / weight description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Microarray</td>
<td>Valued segments</td>
<td>Number / vector of numbers</td>
<td></td>
<td>Intensity / fold change (for one or several samples)</td>
</tr>
<tr>
<td>Sequence reads</td>
<td>Segments</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Paired-end sequence reads</td>
<td>Linked segments</td>
<td></td>
<td></td>
<td>Edges represents pairing of reads</td>
</tr>
<tr>
<td>RNA-seq / ChIP-seq signal</td>
<td>Step function</td>
<td>Number</td>
<td></td>
<td>Number of reads</td>
</tr>
<tr>
<td>ChIP-seq peaks</td>
<td>Valued segments</td>
<td>Number</td>
<td></td>
<td>Signal value</td>
</tr>
<tr>
<td>DNase I hypersensitivity sites</td>
<td>Valued segments</td>
<td>Number</td>
<td></td>
<td>Signal value</td>
</tr>
<tr>
<td>ChIA-PET</td>
<td>Linked segments</td>
<td>Number</td>
<td></td>
<td>Ligation frequencies</td>
</tr>
<tr>
<td>Hi-C</td>
<td>Linked genome partition</td>
<td>Number</td>
<td></td>
<td>Ligation frequencies</td>
</tr>
<tr>
<td>Gene transcripts</td>
<td>Linked segments</td>
<td>None</td>
<td></td>
<td>Edges represent part-of relationship between exons and gene transcripts</td>
</tr>
<tr>
<td>Repeating elements</td>
<td>Valued segments</td>
<td>Category</td>
<td></td>
<td>Categories indicate repeat type</td>
</tr>
<tr>
<td>DNA methylation</td>
<td>Valued points</td>
<td>Number</td>
<td></td>
<td>Percent of reads showing methylation</td>
</tr>
<tr>
<td>Histone modifications</td>
<td>Valued segments</td>
<td>Number</td>
<td></td>
<td>Segments represent nucleosome position; values are ChIP-seq signal values</td>
</tr>
<tr>
<td>Nucleosome positions</td>
<td>Step function</td>
<td>Number</td>
<td></td>
<td>Signal enrichment</td>
</tr>
<tr>
<td>TRE predictions (e.g. ChromHMM)</td>
<td>Step function</td>
<td>Category</td>
<td></td>
<td>Categories indicate type of TRE</td>
</tr>
<tr>
<td>SNPs in individual</td>
<td>Valued points</td>
<td>Character</td>
<td></td>
<td>Nucleotide variant</td>
</tr>
</tbody>
</table>

Continued on next page
<table>
<thead>
<tr>
<th>Type of dataset</th>
<th>Track type</th>
<th>Value type</th>
<th>Weight type</th>
<th>Value / edge / weight description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>List of characters / numbers</td>
<td>Binary</td>
<td>Nucleotide variants / allele frequencies</td>
</tr>
<tr>
<td>SNPs in population</td>
<td>Valued points</td>
<td>List of characters</td>
<td></td>
<td>The inserted sequence</td>
</tr>
<tr>
<td>Insertions</td>
<td>Valued points</td>
<td>List of character</td>
<td></td>
<td>Values represent inserted sequences, if any; edges connect two breakends; weights indicate the direction that the joined sequence continues in</td>
</tr>
<tr>
<td>Deletions</td>
<td>Segments</td>
<td></td>
<td></td>
<td>Copy number</td>
</tr>
<tr>
<td>Reversions</td>
<td>Segments</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Translocations</td>
<td>Linked valued points</td>
<td>List of characters</td>
<td>Binary</td>
<td>True if nucleotide is a G or C, else False</td>
</tr>
<tr>
<td>Copy number variation</td>
<td>Valued segments</td>
<td>Number</td>
<td></td>
<td>Number of CpGs in each island</td>
</tr>
<tr>
<td>Virus integration sites</td>
<td>Points</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GC content</td>
<td>Function</td>
<td>Binary</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CpG islands</td>
<td>Valued segments</td>
<td>Number</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DNA melting temperature</td>
<td>Function</td>
<td>Number</td>
<td></td>
<td>Prediction of melting temperature at each position</td>
</tr>
</tbody>
</table>
Table 4.2: Overview of tools and methodologies for data analysis, as reviewed in section 1.3, categorized according to a single or a pair of track types (as illustrated in Figure 3.2). For each valued track type, the corresponding value type is indicated, along with a description of typical content. Note that categories can be used to combine several tracks into one, with the category indicating the name of the track.

<table>
<thead>
<tr>
<th>Tool / method</th>
<th>Refs</th>
<th>Analysis regions</th>
<th>Track 1 type</th>
<th>Track 2 type</th>
<th>Value type</th>
<th>Value description</th>
</tr>
</thead>
<tbody>
<tr>
<td>seqMiner (heatmap)</td>
<td>[142]</td>
<td>User bins</td>
<td>Valued segments</td>
<td>Category</td>
<td>Track names</td>
<td></td>
</tr>
<tr>
<td>Cistrome (heatmap)</td>
<td>[143]</td>
<td>User bins</td>
<td>Valued segments</td>
<td>Category</td>
<td>Track names</td>
<td></td>
</tr>
<tr>
<td>Spark</td>
<td>[144]</td>
<td>User bins</td>
<td>Valued segments</td>
<td>Category</td>
<td>Track names</td>
<td></td>
</tr>
<tr>
<td>ChromaSig</td>
<td>[145]</td>
<td>Genome-wide</td>
<td>Valued segments</td>
<td>Category</td>
<td>Track names</td>
<td></td>
</tr>
<tr>
<td>HMMSeg</td>
<td>[146]</td>
<td>Genome-wide</td>
<td>Valued segments</td>
<td>Category</td>
<td>Track names</td>
<td></td>
</tr>
<tr>
<td>Spatial clustering</td>
<td>[147]</td>
<td>Genome-wide</td>
<td>Segments</td>
<td>Segments</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ChromHMM</td>
<td>[148]</td>
<td>Genome-wide</td>
<td>Segments</td>
<td>Segments</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Segway</td>
<td>[150]</td>
<td>Genome-wide</td>
<td>Segments</td>
<td>Segments</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SOM method</td>
<td>[152]</td>
<td>Genome-wide</td>
<td>Segments</td>
<td>Segments</td>
<td></td>
<td></td>
</tr>
<tr>
<td>seqMiner (dotplot)</td>
<td>[142]</td>
<td>User bins</td>
<td>Valued segments</td>
<td>Category</td>
<td>Track names</td>
<td></td>
</tr>
<tr>
<td>Cistrome (correlation)</td>
<td>[143]</td>
<td>User bins</td>
<td>Valued segments</td>
<td>Category</td>
<td>Track names</td>
<td></td>
</tr>
<tr>
<td>BEDTools</td>
<td>[155]</td>
<td>Genome-wide</td>
<td>Segments</td>
<td>Segments</td>
<td></td>
<td></td>
</tr>
<tr>
<td>GenomicTools</td>
<td>[156]</td>
<td>Genome-wide</td>
<td>Segments</td>
<td>Segments</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BEDOPS</td>
<td>[157]</td>
<td>Genome-wide</td>
<td>Segments</td>
<td>Segments</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MULTOVL</td>
<td>[158]</td>
<td>User bins or genome-wide</td>
<td>Valued segments</td>
<td>Category</td>
<td>Track names</td>
<td></td>
</tr>
<tr>
<td>SegTools</td>
<td>[159]</td>
<td>Genome-wide</td>
<td>Valued segments</td>
<td>Valued segments</td>
<td>Category</td>
<td>Labels for both segmentations</td>
</tr>
<tr>
<td>Mutation rate co-variation</td>
<td>[160]</td>
<td>User bins or genome-wide</td>
<td>Step function</td>
<td>Vector of numbers</td>
<td>Counts in 1-Mb windows for a range of tracks</td>
<td></td>
</tr>
<tr>
<td>GREAT</td>
<td>[161]</td>
<td>Genome-wide</td>
<td>Segments</td>
<td>Valued segments</td>
<td>Category</td>
<td>Genes of track 2 are categorized according to Gene Ontology terms</td>
</tr>
</tbody>
</table>

Continued on next page
<table>
<thead>
<tr>
<th>Tool / method</th>
<th>Refs</th>
<th>Analysis regions</th>
<th>Track 1 type</th>
<th>Track 2 type</th>
<th>Value type</th>
<th>Value description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cistrome (CEAS)</td>
<td>[143]</td>
<td>Genome-wide</td>
<td>Segments</td>
<td>Step function</td>
<td>Category</td>
<td>Segmentation labels (exons, introns...)</td>
</tr>
<tr>
<td>EpiExplorer</td>
<td>[164]</td>
<td>Genome-wide</td>
<td>Segments</td>
<td>Valued segments</td>
<td>Category</td>
<td>Names of tracks in system</td>
</tr>
<tr>
<td>GenomeRunner</td>
<td>[175]</td>
<td>Genome-wide</td>
<td>Segments</td>
<td>Valued segments</td>
<td>Category</td>
<td>Names of tracks in system</td>
</tr>
<tr>
<td>PinkThing</td>
<td>[178]</td>
<td>Genome-wide</td>
<td>Valued segments</td>
<td>Step function</td>
<td>T1: binary;</td>
<td>T1: case or control; T2: segmentation</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>T2: category</td>
<td>labels (exons, introns...)</td>
</tr>
<tr>
<td>EpiGRAPH</td>
<td>[179]</td>
<td>Genome-wide</td>
<td>Valued segments</td>
<td>Valued segments</td>
<td>T1: binary;</td>
<td>T1: case or control; T2: names of</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>T2: category</td>
<td>tracks in system</td>
</tr>
<tr>
<td>EpiCenter</td>
<td>[180]</td>
<td>User bins or</td>
<td>Segments</td>
<td>Segments</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>genome-wide</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GenometriCorr</td>
<td>[181]</td>
<td>Genome-wide</td>
<td>Points / segments</td>
<td>Points / segments</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ENCODE subsampling</td>
<td>[183]</td>
<td>Genome-wide</td>
<td>Any non-linked</td>
<td>Any non-linked</td>
<td>Depends on</td>
<td>test</td>
</tr>
<tr>
<td>GSC</td>
<td>[183, 186]</td>
<td>Genome-wide</td>
<td>Segments</td>
<td>Segments</td>
<td></td>
<td></td>
</tr>
<tr>
<td>UL, FL, and FL(r)</td>
<td>[187]</td>
<td>Genome-wide</td>
<td>Valued points</td>
<td></td>
<td>Category</td>
<td>Names of transcription factors</td>
</tr>
<tr>
<td>Hypergeo. and $\chi^2$ tests,</td>
<td>[188]</td>
<td>Genome-wide</td>
<td>Segments</td>
<td>Segments</td>
<td></td>
<td></td>
</tr>
<tr>
<td>and log-linear model</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Hybrid resampling</td>
<td>[189]</td>
<td>Genome-wide</td>
<td>Segments</td>
<td>Segments</td>
<td></td>
<td></td>
</tr>
<tr>
<td>“Exact p-value”</td>
<td>[190]</td>
<td>User-defined</td>
<td>Segments</td>
<td>Segments</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Multiv. Hawkes process</td>
<td>[191]</td>
<td>Genome-wide</td>
<td>Valued points</td>
<td></td>
<td>Category</td>
<td>Names of transcription factors/TREs</td>
</tr>
</tbody>
</table>
4.2 Confirmatory or exploratory data analysis?

Paper I and Paper III focus on respectively confirmatory and exploratory data analysis, even though both papers make use of hypothesis tests. This section will discuss the choice of hypothesis tests for confirmatory analysis in Paper I, and further compare the confirmatory approach of Paper I with the exploratory aspects of Paper III.

The main statistical focus of the Genomic HyperBrowser (Paper I) has been on confirmatory data analysis through the development of hypothesis tests. The major question to ask for such a system is then quite obvious: is hypothesis testing really a good approach for querying relations between genomic track data? The basic issue is really a general one: should statistical hypothesis testing be commonly used as a part of the hypothetico-deductive method (see section 1.3.2) at all? This is an old debate, but still relevant. One aspect of the debate regards the difficulties in interpreting the p-value itself. In a study in 2002, a set of incorrect statements on the interpretation of p-values were presented to students, lecturers and professors of psychology, and lecturers in statistics in six German universities, asking them to mark the statements as true or false [121]. Respectively 100%, 90% and 80% of the different groups endorsed at least one of the false statements, which were all erring in the direction of “wishful thinking”. As there is traditionally little focus on the integration of statistics courses in the education of biology students [200], there is no reason to believe that biologists are any better at interpreting p-values correctly. Common misapprehensions include interpreting a p-value as the probability of the null hypothesis (i.e. that the observations are caused by chance), the probability of the alternative hypothesis, or the probability that replication of the study will cause significant results. The p-value is less informative than either of these misapprehensions, namely, the probability of the observations or something more extreme, given that the null hypothesis is true. Even if interpreted correctly, a p-value leading to rejection of the null hypothesis does not necessarily mean that the theory behind the alternative hypothesis is strengthened, as there obviously may be different explanations for the alternative hypothesis. [201–203]

In Paper I the p-values are, to the knowledge of this author, interpreted correctly. However, as the Genomic HyperBrowser is meant as a tool for helping researchers in biology to assess hypotheses on relations of genomic tracks, it may be a major issue if the users are prone to misinterpreting the results. One may argue that it is the responsibility of the users to interpret
the results correctly, as a software tool cannot, and should not, do all the thinking for the user. There is, on the other hand, a major element of expert knowledge incorporated with the HyperBrowser, as evident in the phrasing of questions, hypotheses and answers in natural language, as well as in the automatic selection of an appropriate hypothesis test based on the combination of a user-selected question and a null model. Some precautions against misinterpretations have, however, been taken, such as comments on the results page describing e.g. that “low p-values are evidence against Ho”.

A particular difficulty with hypothesis testing is represented by tests of so-called “nil hypotheses”, i.e. a null hypotheses of zero difference between two groups. In this case, the p-value decreases with sample size. Two groups will in practice always differ for the observation in question at some decimal point. As the sample size increases, ever smaller differences will be statistically significant. This was a difficulty with several of the analyses implemented in the Genomic HyperBrowser. A particular problematic example of sample size-dependent p-values was encountered with the question of whether two function tracks were correlated. As each function track in the human genome consists of over 3 billion numbers, a null hypothesis of zero correlation would in practice always be rejected. A partial solution for reducing the sample size was implemented, but such a solution does not really fix the underlying problem. The basic issue is that a p-value does not say anything about the size of the effect, which should always be considered together with the assessment of statistical significance. A rejection of a nil hypothesis can really only say something about the direction of the relation, e.g. that A is larger than B, from which one usually does not learn very much. Again, in the words of Tukey:

The physical scientists have learned much by storing up amounts, not just directions. If, for example, elasticity had been confined to “When you pull on it, it gets longer!,” Hooke’s law, the elastic limit, plasticity, and many other important topics could not have appeared. [201]

Hypothesis testing was chosen as the main focus of Paper I by the collaborating statisticians before this author started with the PhD project. Alternative methodological principles could also have been chosen and still provided similar functionality. Use of confidence intervals instead of p-values could potentially have been used to alleviate some of the problems discussed above, while at the same time allowing the estimation of unknown distributions by Monte-Carlo simulations [201, 204]. It is, however, not trivial to see how Monte-Carlo based confidence intervals could be implemented in the
current setting. Another possibility is represented by Bayesian approaches [205].

Methodologies for confirmatory analysis are highly dependent on the biological validity of the underlying assumptions. This may also hold true for exploratory analyses, but one may allow more leeway in the validity of the assumptions in such cases, as other methods must still, in the end, be used for validation. In **Paper III**, p-values were used to signify over- or under-representation of transcription factor binding near genes associated to a particular disease. Despite this, the main approach was an explorative one. The p-values were derived from null model distributions that were adequately approximated by the Z-values measuring deviation of the signals from the expected values. Thus, significance may be regarded simply as representing a direct thresholding of the corresponding Z-values. The Z-values are displayed as colors in the heatmaps, in addition to being used for clustering. As the “regulome” heatmaps already display deviation from the null model, the addition of significance marks does not really provide the user with much additional information other than a (rather arbitrarily selected) threshold. The p-values should then be considered exploratively. In addition, there is no multiple testing correction. One reason for this is that just a fraction of all TF-disease combinations are usually interesting for the user. Correcting for all combinations would then in almost all cases lead to a too conservative estimation of the p-values, probably removing most, if not all, significance in the heatmap. One result of the lack of multiple testing is that, if one considers more than just a single TF-disease combination, the p-values cannot be directly interpreted in a confirmatory setting.

**Paper III** is explicit in that the results are to be considered as hypothesis-generating. **Paper I**, on the other hand, is explicit in the assumption that, given a null model that gives biological sense, the user may expect to be able to use the system to test biological hypotheses. However, as all bioinformaticians are aware, no null model is ever fully valid, and one should strive for experimental validation of all results based on computations. Journals routinely reject papers that do not provide satisfactory experimental validation of the findings. Thus, in the light of the full scientific method as described in the hypothetico-deductive method, one may argue that the Genomic Hyper-Browser and the other “confirmatory tools” described in section 1.3.1 are not able to confirm biological findings by themselves. All results should, in this view, be validated in the lab. The “confirmatory” tools do, however, provide (explicit or implicit) theoretical null models that are tested formally on the data, and that failing to reject null hypotheses in will at least work as a filter for removing the “worst” hypotheses before experimental validation. As such, the theoretical role of computational biology is downplayed. In prac-
tice, though, removal of hypotheses in silico may still greatly reduce the cost of wet lab experiments. It is the view of this author that confirmatory analysis using computational biology tools will continue to need confirmation by wet lab experimental designs for the foreseeable future.

4.3 Biological relevance

As touched upon in the preceding section (4.2) it is imperative to ask whether, and to what extent the developed methodologies are biologically relevant. The questions and related hypothesis tests developed in Paper I were identified from pondering the possible geometric relations between pairs of track of different track types. The mathematical relations came first, biological relevance later. It is still an open question whether the hypotheses developed were the most biologically relevant ones. Function tracks, for one, are still not very common, so the hypothesis tests on function tracks seems, in hindsight, to not be as relevant as segment-based tests. The biological questions chosen for Paper I were also chosen for illustrative purposes more than representing important research questions. For instance, the use of MLV virus integration sites as an example of local analysis was not well grounded, especially as the number of data points was too low for this to make statistical sense. More work in the direction of biologically relevant examples and research questions could perhaps have increased the impact of the paper. The software and the methodology may have found more use by researchers if their application had been shown to produce novel and intriguing biological results, more than what was shown in the biological examples presented in the paper.

A particular problem concerns the biological relevance of the null models. As discussed in Paper I, the null model should as much as possible reflect relevant biological characteristics of the dataset. The paper does not, however, go into much detail on the mapping of null models to biology, leaving this to the user of the system to decide. As a valid null model is imperative for producing valid results, this question is of the utmost importance. Finding an appropriate null model for a particular research question is, however, very difficult. A paper with further discussions of the implications of Monte Carlo null model assumptions has been written by some of the co-authors of Paper I and is currently submitted to a statistical journal. The paper also explores some different criteria for choosing appropriate assumptions.

In hindsight, a significant omission of Paper I is the lack of detailed description of the subsampling method of Bickel et. al [183]. The paper describing the thorough statistical treatment of the method was published
in parallel with Paper I in Dec. 2010, and could therefore not have been referenced. The major ideas of the method had, however, been previously published as a part of the supplementary material of the pilot ENCODE paper [80] in 2007. We were aware of this method and described it in Paper I, but referenced the wrong article ([78] instead of [80]). Also, the description was rather superficial. A solution to the problem of specifying biologically relevant null models could have been to implement the subsampling methods of Bickel et al. These subsampling methods circumvent the need for exact definitions of the null model elegantly by subsampling from piecewise stationary regions of the data. The subsampling methods are not, however, able to take confounding tracks into account, as does the intensity track functionality implemented in the Genomic HyperBrowser.

A more particular critique of the implemented null models is that most of them are based on randomization over the complete domain of the track, typically the complete genome. However, some areas of the genome may be, for biological reasons, depleted for the track elements in question. If the null model does not take such depletion (or enrichment, for that matter) into account, the test may result in false positives. Null models based on intensity tracks may help, but this is only available for a handful of questions. Further developments of the intensity track concept, e.g. to support segments in addition to points, will help. Today, the centromeres are removed from the null model randomization by default. The additional removal of e.g. repeating elements could improve the situation.

Paper III goes to some length to describe the biological relevance of the developed regulome heatmaps. The main immune-related clusters are, for instance, discussed in some detail, referencing literature that provides support for the TF-disease relations found. The paper is, however, vague when concerning the biological implications of the finding. This vagueness is deliberate, of course. The main goal of the paper was to describe a methodology, not to generate new hypotheses or provide important biological findings. It would have been advantageous if the paper would present novel and interesting TF-disease relations with experimental validation. The regulome methodology was, however, originally meant to be included as an additional example in Paper I, but was later expanded to a separate paper. Experimental validation was outside of the scope of the project. The paper could, on the other hand, have discussed the biological premises more carefully. One premise of the relevance of methodology is the possibility of using transcription factors as drug targets. Even though examples of drugs inhibiting transcription factors exist, such targeting remains a challenging endeavour and further research is required [206]. It is also not given that a transcription factor that is overrepresented as a regulator of many genes as-
associated to a particular disease constitutes a good drug target. The complexity of the regulatory and signalling pathways may undermine this premise.

**Paper III** presents two different types of heatmaps based upon two different null hypotheses. The biological relevance of the null hypotheses is, however, not discussed. Simply explained, the first heatmap type shows over- and under-representation as compared to the total matrix of row-column combinations, while the other heatmap type shows over- and under-representation as compared to the rows only. In the case of the TF-disease regulomes, the first heatmap type shows which TF-disease combinations that stand out compared to the others, *i.e.* that have TF-gene associations that are specific as compared to the other diseases and TFs. The second heatmap type shows TF-disease combinations that stand out as compared with TFs only, possibly showing regulatory activity of genes that are common for many diseases. Interpretation of the different heatmaps is not discussed in any detail, but left to the user. Even though the differential aspect of the first null hypothesis makes it possible to say something about a relation being specific for a particular disease or sets of diseases, it is very hard to say anything precise about this, as the z-values are dependent on all the other diseases, many of which may be arbitrarily added to the analysis. One may envision other null hypotheses that make use of other backgrounds. For instance, one may envision a heatmap generated for a particular cell type, using the other cell types as the control set. Another option would be to adjust the relevance of the disease-associated genes by prediction scores, which is not done presently. A more thorough treatment of the choices of null hypotheses would have improved the paper.

The datasets used in **Paper III** are mostly predictive and thus probably quite noisy. The original heatmap datasets were found to be too noisy, *i.e.* the combination of predicted transcription factor binding sites (TFBS) from UCSC ("TFBS conserved") with gene associations predicted from literature with the PubGene system [207]. The TFBS predictions were replaced with gene binding predictions based on machine learning [198], while the PubGene predictions were replaced with manually curated gene associations from Phenopedia [199]. The signal/noise ratio of the new datasets were not, however, investigated further. It seems natural to assume that the TF binding predictions are not optimal, as they are fundamentally based on DNA motifs, as represented in position weight matrices (PWMs). Using ChIP-seq datasets would probably be an improvement, but such datasets were not at the time available in the same number as they currently are, after the completion of the main phase of the ENCODE project [79]. An obvious alternative to the gene ontology dataset used in the alternative heatmaps of **Paper III** is to download and use the gene ontology database themselves, rather than
basing the associations on predictions from the PubGene system.

A problem with the TF bindings that were used is that the machine learning method directly predicts the relation between each TF (or PWM) and a list of genes, without predicting the actual binding sites. The main regulome thus did not illustrate the sequence-level opportunities of the methodology properly. Relations between TF binding and other datasets, such as histone modifications, are dependent on positional binding information. It was thus also not possible to map such relations using the heatmap methodology using the main TF binding dataset. More recent datasets would alleviate this problem, such as TFBS predictions based on the CENTIPEDE software [208], experimental in vitro binding profiles [209], or in vivo, cellspecific binding data from ChIP-seq experiments [210]. Better rules or predictions for linking regulatory regions to genes may also increase the signals. At the very least, the approach used in GREAT [161] could be implemented. Lastly, the thresholds used for the generation of the regulomes, such as the use of the top 1000 predicted genes relations for each TF, were rather arbitrarily chosen. The effects of changing such parameters, as well as clustering parameters, were not discussed in detail in the paper.

4.4 User experience

Much work has gone into the design and development of the user interface of the Genomic HyperBrowser, which contains all the tools described in all three papers in this thesis. The system was meant to be easy to use, but users have, however, complained that the interface is rather complicated. The paper on the recently published PinkThing system describes that the Genomic HyperBrowser “has a complex interface that takes time to learn to use efficiently” [178]. This is, of course, in comparison to the interface of the PinkThing system, which is a much simpler system overall. Still, such critiques have some validity.

The following will discuss the tool “Analyze genomic tracks”, which is the main tool described in Paper I, containing the main interface to the statistical analysis system. One of the basic ideas was to design the system as a sort of “expert system”, based on a series of steps: 1) the user provides one or two datasets; 2) based on the track types of the datasets, the system presents a set of questions that may be asked; 3) the user selects the appropriate question; 4) the system presents a set of parameters relevant for the question, including the selection of a null model; 5) the user chooses parameters and starts the analysis; and 6) the system presents the results in a simple, answer-like manner, in addition to providing the underlying results as tables and graphs.
An advantage of this system is that the user does not need to know which exact hypothesis test that should be run, only the assumptions he makes on the data. Also, only relevant choices are displayed at each step. The main problem with this approach is that all of the choices are hidden from the user at the beginning. Some of the analyses are only presented to the user if the right combination of tracks is selected. The user has thus no way of knowing which questions are possible. Such a list of questions could have been presented somewhere in the system, but this was not done for reasons of maintainability; the list would soon have been outdated. In hindsight, this may have been a wrong choice. A recent article on the system presents such a list of questions, alleviating the problem somewhat ([211], included in appendix A as table A.1). Some work has also gone into developing a simpler, “wizard”-based interface for beginner users, as an alternative to the more advanced interface which is currently the only choice.

One other reason that users may find the user interface difficult is that it is designed to be completely general with respect to the type (biologically speaking) of the selected datasets, and the context of the analysis. This is quite uncommon for bioinformatics tools, which are usually made with a focus on particular datasets or research questions. This problem may be alleviated with the development of specific tools focused at a particular application. One example of such a tool is the tool for the analysis of spatial colocalization of track elements in 3D [212]. A tool for analyzing GWAS data in relation to, among other things, cell types, is also under development.

When initially designing the system, it was thought that a repository of genomic tracks on the server would be important. After observing users, it now seems that the tracks in the repository are not used very much. Users typically upload several custom datasets that they want to compare. Based on this knowledge, the user interface could have been redesigned with a more focus on user-uploaded tracks. The new “wizard”-like interface described above will have such a focus.

There are at least four problematic issues with the current track repository: 1) there is no search option for finding tracks; 2) the tracks are seldom if not at all updated as the source improves; 3) track metadata is lacking for many tracks; and 4) many common tracks are missing. A search option would not be very difficult to implement and is a job done once. Also, a system for accessing datasets available at the UCSC server [176] has been implemented but not yet released as a production-quality feature. This feature would alleviate the three other issues somewhat, but would only support segment or point tracks. The main problem with keeping a local repository of tracks is that this requires quite some maintenance effort, which is a difficult thing to prioritize in a research-based setting. The right approach to track
maintenance seems not to have been found yet.

Perhaps the most confusing aspect of the user interface is the selection of the null models. The reason for this is obvious: there is no easy connection between the biological mechanism one is studying and the correct null model. Paper I does not address this difficulty in a very detailed manner. In essence, the paper only states that several null models are implemented, that different null models may give profoundly different results, and that looking at the results from different null models may provide valuable information. This does not help the user very much in the choice of null models. Many biologists are biased towards using systems where one just “pushes a button” and receives the results. On the other hand: as the appropriate choice of null model is essential for accurate results, one may argue that the choice of null model must be done by the user. The system could, however, have been better in informing the user on the details of the different null models, possibly evaluating how well the null model fits the data.

After carrying out an analysis, the results need to be interpreted. In section 4.2, possible misinterpretations of the p-value was described. An extended problem occurs as a consequence of the system calculating p-values both globally and for the local analysis regions. These p-values may give different conclusions, as shown in the H3K27me3 versus SINE repeats example in Paper I. The user is still not properly informed on how to interpret such results. In order for the system to provide better help to the users, such situations would need to be studied in a systematic manner, which has not yet been done.

The previous discussion has focused on the user interface of the main tool presented in Paper I. This main analysis functionality is augmented by a range of tools for data preparation, data customization and other forms of analysis (see appendix A for an overview). Also Paper II and Paper III present graphical web-based tools. Most of the tools have been created using a general system for dynamic tool specification, developed by us. By the use of this framework, functionality for dynamically updating the parameter lists and contents, as well as functionality for dynamic parameter validation, has been made possible. One example of such dynamic behavior is found in the tool “Create GTrack file from unstructured tabular data” presented in Paper II. The tool allows the conversion of any segment or point-based tabular data to the GTrack format. The GUI displays the first lines of the selected tabular file in a table, automatically updating the column names depending on the user selections. The list of available tools is not in any way complete. New tools are developed and added as they are needed by internal or external users of the system. One particular tool that has yet to be developed is a tool for editing the header lines of a GTrack file. Such a
tool was planned as part of Paper II, but was postponed because of time limits.

Paper III described a user interface which is conceptually different from the ones described in the other papers. Because of the large size of the heatmaps, the Google Maps engine [213] was used to enable the display of parts of the heatmap in a web browser. Additional functionality for pinpointing the results related to single disease-TF combinations, or clusters of diseases and TFs, was implemented using this API. Clicking on such markers presents the gene associations and statistics underlying the relations. Also, a search functionality was added for easier traversal. The heatmap interface has been found a little difficult for some first-time users, especially when trying to describe the borders of the clusters. More help texts and usage tutorials could have been added. The current interface is also focused on the use cases where the user is a priori interested in specific diseases and/or TFs. The only way users may search the massive heatmaps for interesting findings is by manual scrolling. One may envision functionality that automatically discerns relations and clusters of relations from the heatmap depending on deviance from the null model, as well as checking the literature for novel findings. The massive size of the heatmaps available through the interface has, however, worked well as an eye-catcher in order to generate interest in the system. Of particular note is the lack of functionality for generating interactive heatmaps for any user-defined pair of categorical tracks. This was planned, but never properly implemented.

One last point related to the user interface is the issue of reproducibility. The scientific method is dependent upon the ability for other researchers to replicate studies. Modern biological publications, however, often makes use of computational analyses that are poorly described and often, in practice, unrepeatable. A recent study found that, out of 50 randomly selected papers from 2011 making use of the Burrows-Wheeler Aligner for Illumina reads alignment, only 7 provided all the necessary details for reproducibility [214]. As the Genomic HyperBrowser is built upon the Galaxy framework [139–141], which has been built specifically with a focus on reproducibility, much functionality for increasing reproducibility is provided “for free”. The HyperBrowser system does not, however, completely follow the standards for Galaxy tools, as the system required more advanced user interface than what was provided by Galaxy. Even though results of analyses are stored, and can be shared and re-run, HyperBrowser tools cannot be included in Galaxy workflows. Also, the tracks in the repository are only referenced by name. Changing the names or contents of repository tracks may thus invalidate prior analyses.
4.5 Implementational issues

As described in Paper I a range of structural decisions were made for the system architecture of the Genomic HyperBrowser. One of the design goals was to support dynamic loading of tracks, analyses, and parameters without requiring redundant storage. For instance, the storage of all analyses and their connections to track types in a static table would soon be out of sync with the code. In contrast to such a solution, the list of relevant analyses is based upon small dynamic runs of all analyses in the system, dynamically removing the ones that are not relevant or are not working. This system has worked very well in practice, disregarding some issues with debugging.

Another goal for the system architecture was for the code to be structured as much as possible as loosely coupled modules, in order to increase robustness and re-use. This was successful to some extent, even though parts of the code ended up quite intermingled. The architecture of the statistical modules was particularly effective. The source code exhibits abundant reuse of statistical modules, as intended. Also, the separation of different parts of the code was quite successful, as witnessed e.g. with the reimplemention of the data preprocessor and storage modules after the GTrack file format had been fully specified (Paper II). This major refactoring effort required minimal change in the code for the statistical modules. However, the preprocessor code was, and still is, in itself quite intermingled, providing an exception to the rule of loosely coupled modules.

Throughout the development process, a focus has been to try to balance performance with speed of development. The Genomic HyperBrowser was mainly written in Python [215], which is a high-level programming language that provides fast and flexible software development. This was matched with writing speed-critical parts of the code in Numpy [216], a numerical library for fast vector operations written in C. This has worked well in practice.

Of particular notice is two components that have been written with performance in mind: the code for binary representation of track data and an algorithm for calculating counts used when creating the the regulome heatmaps. As described in section 4.1, the internal representation of track data makes use of four core numerical arrays stored on disk: start, end, val, and edges. The fifteen track types correspond directly to all possible combinations of these four arrays being defined for a track. Three non-core reserved arrays were also defined (strand, id, and weights), in addition to the support for storing any columns on disk as arrays of strings. The separate indexing structure is created in order to provide efficient random-access to the data. The code also handles tracks with elements that overlap within the track by storing two version of the arrays in parallel: one with overlapping
elements and one where the overlapping elements have been merged. This array-based implementation of binary data storage has allowed the use of Numpy operations for efficient computation throughout the system, a design that has been working very well. The step of parsing input text files to populate the binary array structures is, on the other hand, rather slow. As this is only done once for each track, the performance of this code has not been prioritized. For certain types of tracks, e.g. function tracks, the slow speed constitute a real bottleneck. Of note is also the realization that parsing GTrack files into such binary formats was found to be more complex than first envisioned when specifying the format for Paper II. Rewriting the Genomic HyperBrowser code to support the full GTrack specification took around half a year of focused development for one person. This implementation was carried out after Paper II was published. However, the preprocessor code was fully rewritten, as well as the code for converting between the different formats, adding support for interconnections throughout the system. This rewrite has allowed the implementation of 3D-type analyses, which is the focus of another PhD project in the research group [212]. Features of the GTrack format that is still not completely implemented in the HyperBrowser system is the support of lists as values or weights (because of limitations with the Numpy library), as well as the support for more than one genome in a file. Another shortcoming of the code is that it still does not support storage of tracks where overlapping elements have been merged separately for each category. Lastly, it should be noted that missing data values are currently only handled in a very basic way. Improvements in this direction could improve the robustness of the analyses. The code for binary representation of track data has been extracted to a separate python library, named GTrackCore, for use by other tools and systems [217].

The main heatmap of Paper III relates 446 transcription factors with 1010 diseases, thus containing the results of around 450 000 genome-wide comparisons. The first attempt at generating such a massive heatmap (though with different input tracks) were based on iteration over all pair-wise comparisons. This attempt ran for over 14 days on the HyperBrowser server before crashing, never to be finished. An improved algorithm was then developed in order to reduce the running time. The algorithm is presented in appendix C. The two input tracks are categorical tracks of type valued points and valued segments respectively. The algorithm is based upon the generation of a massive boolean matrix with all categories as rows and all base pairs as columns. Each cell is set to true if the corresponding base pair is covered by a track element of the particular category. Based on this matrix, a smaller matrix of track 1 categories vs track 2 categories is created, containing the counts of points inside segments of all combinations of categories.
As shown in code listing 1, the actual implementation exchanges most of the for-loops with Numpy vector operations, providing fast performance within the Python code. As the boolean matrix created by the algorithm may be very large (i.e. $\sim 340$ GB for chromosome 1 of the human genome), the calculation of the matrix is split into smaller bins. The resulting matrices, one for each bin, are later added together to generate the final matrix. The running time of the main regulome was reduced to $\sim 3 - 4$ hours after the algorithm was implemented, exemplifying the power inherent in the combination of Python for quick development, and the Numpy library, for fast performance.

Even though performance has been a goal, some parts of the code have remained very slow, limiting their practical use. One example of code with poor performance is tools and functionality based the sliding-window calculations. Fixing such problems is typically only a matter of focusing some development time on the matter. Another option, which has not been tested much, is the compilation of the code using Cython [218], which can be used to convert Python code to compilable C-code.

A recurring issue when dealing with bioinformatics tools is that maintenance and further development of software after publication is usually not prioritized. The minimal requirements for maintenance of scientific software ought to be to keep the software running and to continuously patch the code to fix any bugs that are discovered. A study from 2008, however, showed that 14% of computational biology resources published in the Web server issue of Nucleic Acids Research in 2004 had ceased to work [219]. Ideally, scientific software should, in addition, be continuously expanded and improved after publication, not only by fixing bugs but also by adding new functionality. It is after publication that the real-world requirements of the users will appear. Most research grants do not, however, include the costs of software maintenance. Also, universities may be skeptical to releasing free software to the research community, instead wanting to gain revenues by licensing the software to commercial agents [220]. The Genomic HyperBrowser project is in the fortunate position of being developed in a bioinformatics core facility setting, where the software is used for providing service to end users. Maintenance and further developments is thus economically possible within this organization. However, as with all scientific tools, further development must be grounded in the continued need of the functionality.
4.6 Comparison with existing methodologies

When this PhD project was started, most of the methodologies described in section 1.3 was yet to be developed. As the Genomic HyperBrowser (HB) was meant to be a general-purpose analysis system, most of the tools and methodologies described are relevant for comparison purposes. In order to focus the discussion, however, only the most similar tools will be discussed. In the case of Paper I, the relevant tools and methodologies are mainly the ones developed for confirmatory pair-wise track analysis. The different implementations of simple pairwise overlap tests (section 1.3.2.1) are all improved by the Genomic HyperBrowser by its ability to specify more complex null models. The transcription factor co-occurrence problem (section 1.3.2.4) has not been explicitly solved in the HB setting, and the system hence cannot be compared directly with the other solutions to this problem. What is clear, however, is that most of the proposed null models may easily be incorporated with the HyperBrowser system. The closest competitor is the GenometriCorr system [181], which provides four generic hypothesis tests in the same vein as the HB system. Both the absolute distance test and the projection test have been independently implemented in the HB, the former with some differences. The Jaccard test was considered for implementation, but not found expressive enough. Lastly, the relative distance test has not been implemented in HB, but could be added quite simply. The main difference between GenometriCorr and the Genomic HyperBrowser is that the former only provides uniform null models, while the latter provides a large set of null models, including support for confounding tracks by the use of intensity track randomization. GenometriCorr is implemented in R, which is the tool of choice for most statisticians. Biologists, however, usually prefer web-based tools. The Genomic HyperBrowser, being a web-based tool, is thus more focused toward biologist users than GenometriCorr.

The other software system that competes directly with the Genomic HyperBrowser is the Genome Structure Correction (GSC) software implementing the subsampling approaches of Bickel et al. [183] for the overlap test. The main advantage of the subsampling method is that no explicit specification of a null model is required. According to the paper of Bickel et al., the subsampling approach provides a better estimate to the true distribution than uniform randomization. The Genomic HyperBrowser, on the other hand, provides support for confounding tracks by the use of intensity tracks. Intensity track randomization is, however, not available for the overlap test. Even though only the overlap test has been implemented in the GSC software, the generality of the subsampling methods echoes that of the Monte Carlo approach on the HyperBrowser. The GSC tool is only command-line
based, but a Galaxy [139–141] wrapper exists. Further comparisons of different Monte Carlo based null models with the sumsampling methods of Bickel et al. would provide be needed in order to decide upon whether the subsampling methods should be included with the Genomic HyperBrowser.

**Paper II** describes both the GTrack 1.0 as well as the BioXSD 1.1 formats, but only the former format is regarded a part of this thesis. The paper compares the GTrack format with other common formats, a discussion that will not be repeated here. One of the characteristics of the GTrack format is the support for flexible columns. At least two other formats exist that support flexible columns, but **Paper II** did not explicitly compare these to the GTrack format. The first format is the Galaxy-specific [139–141] “interval” format. The format supports six flexible columns: chromosome, start, end, strand, name, and score, in any order and intermingled with any number of custom columns, similarly to the GTrack format. However, the Galaxy interval format stores the column mapping as metadata in the Galaxy framework, and not in the textual file representation itself. It is thus not a standalone file format and cannot be exported for use in other tools, and will thus not fulfill the same needs as the GTrack format. The other format supporting flexible columns is the bigBed binary format [221]. The bigBed format supports the specification of flexible columns by the use of an additional AutoSQL configuration file. The AutoSQL specification supports explicit type declarations for all columns, not only the core columns as in GTrack. Also, bigBed supports transmitting parts of the file at a time to servers, a feature not supported by GTrack or the related binary representation. One advantage of the GTrack format, on the other hand, is that all metainformation on the columns is stored as headers in the text file itself, not requiring an additional AutoSQL file. Another advantage is the possibility of storing datasets of all 15 track types including 3D-type data. The bigBed and the Galaxy interval formats only support the track types “points”, “segments”, “valued points”, and “valued segments”. Also, GTrack supports the definition of bounding regions specifying the domain of the genome where the track is defined. The Genomic HyperBrowser was built upon array representations of the data in order to support flexible data analysis in all scales. The bigBed binary format, on the other hand, is designed for efficient visualization of parts of the content, or summaries of the data in different zoom levels. The design goals are different, and bigBed would not have fit as the internal data structure of the Genomic HyperBrowser. The combination of a BED-like tabular file and an AutoSQL could, however, have been a valid alternative to GTrack as input to the system. This would, on the other hand, require two input files, which would be cumbersome and fit poorly with the Galaxy framework.

The decision to develop the GTrack format raised a particularly difficult
issue: trying to solve the problem of there being too many standards by developing yet another standard is a rather contradictory thing to do. Changing the adoption of file format standards is a complicated endeavor, even if the format is superior to the existing standards. In this case, the advantages of GTrack may not be convincing enough for researchers to change their habit. On the other hand, as no file format standards exist for 3D-type data, this particular niche seems to be “up for grabs”. It cannot be a disadvantage that the GTrack format in addition can represent most other types of tracks.

The methodology of Paper III has not, to our knowledge, any near competitors. One single row of the heatmap, i.e. one transcription factor, bear some similarity to a single analysis with the GREAT software [161]. With GREAT, a set of regions, i.e. transcription factor binding sites in this case, is linked to nearby genes, which are then used to search for enriched gene ontology terms. With the regulome methodology, such analyses can be carried out simultaneously on a range of TFs. The obvious advantage of this approach is that one may find transcription factors that are related in their regulatory functionality, thus clustering together. The recently published PRISM resource [162] did, in essence, carry out a GREAT analysis on the predicted binding sites of 332 TF binding motifs, resulting in 2543 functional predictions. The only conceptual difference from the GO-based differential regulome was that PRISM did not present the complete analysis as a clustered heatmap, but rather as a searchable web resource. Comparing the results of PRISM with the methodology of Paper III would be interesting, but would probably lead to accepting PRISM as the better solution based on better TFBS predictions, better handling of proximal/distal binding, and better datasets for functional annotations. The intuitive power of clustered visual analysis for providing a global overview is, however, evident when comparing the outputs of the two methods. Other methods that are related to the methodology of Paper III is methods comparing a single track against a set of other tracks, such as EpiExplorer [164] and GenomeRunner [175]. The regulome heatmap functionality can be thought of as expanding such methods to a many-vs.-many genome-wide comparison scale.

4.7 Future perspectives

Proposing a nomenclature for track types, and developing an analysis framework and a file format does not in itself make researchers make use of them. On the contrary, experience from this project has shown that pushing solution through to researchers probably requires hard work. Although the Genomic HyperBrowser has been used in several research projects initiated
by other research groups [222–225], the usage of the system still has a way
to go to reach critical mass. The GTrack format has, to our knowledge, not
yet been used outside of the HyperBrowser system, and neither of Paper II
and Paper III have been cited by any papers written by people outside our
research group. This should not, perhaps, come as any surprise. The ambi-
tions for this PhD project have, however, been high, and further “marketing”
of the developed software and format is a priority for the immediate future.
There may be several reasons for the lack of uptake in the research com-
munity: 1) the users do not know of the solutions or the benefits have not been
sufficiently advocated, 2) the methodologies do not provide enough novelty
as compared with previous solutions, 3) the solutions do not have backing
from internationally recognized collaborations/groups, 4) the software may
be too hard to use, or 5) the software may not be relevant for the specific
scenarios faced by the users. A range of different plans have been proposed
to attempt to improve the situation:

- Make more of a presence in relevant conferences, in addition to pub-
  lishing more articles referencing the system.

- Improve the user experience with a wizard-based graphical user inter-
  face.

- Develop more domain-specific tools such as a GWAS analysis tool us-
  ing ENCODE data, and a tool for providing more detailed properties
  of tracks in a simple manner.

- Investigate the inclusion of the subsampling methods of Bickel et al.
  [183] in order to simplify the process of specifying assumptions

- Update the differential disease regulome Paper III with novel datasets,
such as ChIP-seq and DNase hypersensitivity sites from the ENCODE
project [79], and GWAS sites from the NHGRI GWAS Catalog [226].

- Refactor the Genomic HyperBrowser such that the system can be inte-
  grated as a tool suite in the Galaxy Tool Shed [227]

- Incorporate parallelization code as developed in a master project with
  the production system, radically improving performance.

- Further develop the GTrackCore library [217] in three main directions:
  1) refactor internal structure to support the PyTables library [228] for
     improved indexing, analysis capabilities, compression, and Galaxy in-
     tegration as a single binary file, 2) develop operations and analysis ca-
     pabilities to make the library more useful for tool developers, and 3)
possibly integrate the library with the Galaxy framework, assuming that the Galaxy developers are interested in this. The basic idea is that the adoption of the GTrack format is dependent on a simple and powerful library for use by tool developers.

Several manuscripts of research papers building on the contents of this thesis have been written by other researchers connected to the project. These include:

- A paper presenting basic methodology for analyzing 3D co-localization of genomic elements has recently been published [212]. Further developments of the 3D functionality is planned for a new paper, launching a 3D version of the Genomic HyperBrowser, making use of GTrack as the format of choice for representing linked data.

- A paper on analyzing more than two tracks using the Genomic HyperBrowser [229].

- A paper presenting methodology and software for clustering genomic tracks [230].

- A paper discussing implications of Monte Carlo null model assumptions and exploring some different criteria for choosing appropriate assumptions [231].

- As mentioned above, a project on the cell-specific analysis of GWAS datasets in its beginnings, possibly leading to a separate paper.

In addition to this, the further development of the intensity track functionality is a possibility for a separate paper, but no immediate plans have been made in this direction.
Chapter 5

Conclusions

The technological developments in molecular biology over the last 50 years have brought with them a gradual shift of focus from genes and proteins towards the biological activity in non-coding parts of the genome. Together with this shift has followed an increased need for data representation based on the tracks metaphor, with elements positioned along a reference genome. Next-generation sequencing techniques have accelerated this development significantly. At the start of this PhD thesis, tools and methodologies for downstream confirmatory and exploratory analysis of track-type data were in high demand. Presently, the set of available tools have increased somewhat, but are mostly still specifically focused on certain functionalities. The Genomic HyperBrowser was developed as a general, web-based system for many kinds of data analysis of genomic tracks, focusing specifically on pairwise confirmatory analyses. In connection to this system, common types of heterogeneous track datasets were categorized into fifteen different track types. A novel file format, GTrack, was developed to support datasets of any of the fifteen track types, including 3D-type tracks. Lastly, a heatmap-based exploratory methodology was developed for probing, among other things, transcription factor regulation of common diseases.

The recent flood of ENCODE papers and datasets are a strong reminder that tools and methodologies for analyzing track-type data is in higher demand than ever. It is the opinion of this author that the groundwork presented in this thesis provides a solid methodological and implementational basis for integrated data analysis of heterogeneous track-type datasets. The novel tools and methodologies compare favorably to other existing solutions, at the same time proving the generality needed to allow a range of possibilities for further developments into areas of scientific importance. A move towards developing novel analysis tools for specific data types and/or research
questions would be beneficial, both for ease of use and for biological relevance. A particular focus should be on applications that are easily extended with experimental validation, as all model-based computational analysis of biological relations in the strictest sense are exploratory. Distribution of the developed software in forms that allow easy use and integration with other tools will probably help the research community to adopt the GTrack format, the associated track type concepts, and the data analysis tools developed as part of this PhD project.
References


40. Sanger F, Air GM, Barrell BG, et al. Nucleotide sequence of bacterio-
41. Venter JC, Adams MD, Myers EW, et al. The sequence of the human
43. Nagaraj SH, Gasser RB, and Ranganathan S. A hitchhiker’s guide to
expressed sequence tag (EST) analysis. Briefings in bioinformatics
2006;8:6–21.
44. Shendure J and Ji H. Next-generation DNA sequencing. Nature Biotech-
nology 2008;26:1135–1145.
45. Morey M, Fernández-Marmiesse A, Castiñeiras D, Fraga JM, Couce
ML, and Cocho JA. A glimpse into past, present, and future DNA se-
47. Park PJ. ChIP-seq: advantages and challenges of a maturing technolo-
48. Pepke S, Wold B, and Mortazavi A. Computation for ChIP-seq and
49. Boyle AP, Davis S, Shulha HP, et al. High-Resolution Mapping and
Characterization of Open Chromatin across the Genome. Cell 2008;
132:311–322.
DNA interactions in vivo by digital genomic footprinting. Nature meth-
ods 2009;6:283–289.
51. Dekker J, Rippe K, Dekker M, and Kleckner N. Capturing chromo-
52. Wit E de and Laat W de. A decade of 3C technologies: insights into
53. Wolinsky H. The thousand-dollar genome. Genetic brinkmanship or
54. Rossbach M and Lecea MGM de. Translational genomics in personal-
ized medicine - scientific challenges en route to clinical practice. The
HUGO Journal 2012.


59. BioMed Central copyright and license agreement. URL: http://www.biomedcentral.com/about/license.


87. So much ”junk” DNA in our genome.


125. Tukey JW. We need both exploratory and confirmatory. The American Statistician 1980;34:23–25.


REFERENCES


154. Creative Commons Attribution License. URL: http://creativecommons.org/licenses/by/2.0/.


186. ENCODE Data Analysis Center Software. URL: http://www.encodeproject.org.


217. GTrackCore source code repository. URL: https://github.com/1000/gtrackcore.


## Errata

**Errors corrected from the original manuscript**

<table>
<thead>
<tr>
<th>Page / line / footnote</th>
<th>Original text</th>
<th>Corrected text</th>
</tr>
</thead>
<tbody>
<tr>
<td>iii / 20 /</td>
<td>had yet to be published</td>
<td>had yet been published</td>
</tr>
<tr>
<td>xii / 28 /</td>
<td>modified hi chemical alterations</td>
<td>modified by chemical alterations</td>
</tr>
<tr>
<td>xiii / 1 /</td>
<td>performed on computer</td>
<td>performed with a computer</td>
</tr>
<tr>
<td>2 / / 2</td>
<td>opinion of the latters</td>
<td>opinion of the latter</td>
</tr>
<tr>
<td>4 / 8 /</td>
<td>glsminoaacid</td>
<td>amino acid</td>
</tr>
<tr>
<td>5 / 1 /</td>
<td>the directly transcription</td>
<td>the direct transcription</td>
</tr>
<tr>
<td>6 / 12 /</td>
<td>exons in 1977.</td>
<td>exons in 1977 (see Figure 1.3A).</td>
</tr>
<tr>
<td>6 / 19 /</td>
<td>binding [14])</td>
<td>binding [14], see Figure 1.3B)</td>
</tr>
<tr>
<td>12 / 30 /</td>
<td>one of which are</td>
<td>one of which is</td>
</tr>
<tr>
<td>13 / 7</td>
<td>mitochondrial</td>
<td>mitochondrial</td>
</tr>
<tr>
<td>15 / 21 /</td>
<td>amplification method,x</td>
<td>amplification method,</td>
</tr>
<tr>
<td>16 / 27 /</td>
<td>Conceptually</td>
<td>Conceptually</td>
</tr>
<tr>
<td>24 / 6 /</td>
<td>particular data representation.</td>
<td>particular data representations.</td>
</tr>
<tr>
<td>26 / 5 /</td>
<td>(GWAS)was found</td>
<td>(GWAS) was found</td>
</tr>
<tr>
<td>29 / 17 /</td>
<td>emphi.e.</td>
<td>i.e.</td>
</tr>
<tr>
<td>30 / 12 /</td>
<td>reference genome.[3]</td>
<td>reference genome. [3]</td>
</tr>
<tr>
<td>31 / 22 /</td>
<td>pilot regions [103],[14]</td>
<td>pilot regions [103]. [14]</td>
</tr>
<tr>
<td>32 / 20 /</td>
<td>contribution of rare mutations are</td>
<td>contribution of rare mutations is</td>
</tr>
<tr>
<td>33 / 18 /</td>
<td>dataset is contains</td>
<td>dataset contains</td>
</tr>
<tr>
<td>33 / 21 /</td>
<td>dataset is constitutes</td>
<td>dataset is constituted</td>
</tr>
<tr>
<td>35 / 13 /</td>
<td>track,e.g.</td>
<td>track, e.g.</td>
</tr>
<tr>
<td>35 / 40 /</td>
<td>regions involved transcriptional regulation</td>
<td>regions involved in transcriptional regulation</td>
</tr>
<tr>
<td>36 / 13 /</td>
<td>sequence of number</td>
<td>sequence of numbers</td>
</tr>
</tbody>
</table>

*Continued on next page*
Errors in Paper II

Equation (2) should have looked like this:

\[ S(a, b) = \{ s \in \mathbb{N} \mid a \leq s \leq b \land b > a \} \]

And similarly for equation (3):

\[ P(a) = \{ p \in \mathbb{N} \mid p = a \} \]

Thanks to master student Kristoffer Waløen for pointing out this.
The Genomic HyperBrowser: inferential genomics at the sequence level

Geir K Sandve1, Sveinung Gundersen2, Halfdan Rydbeck1,3,5, Ingrid K Glad6, Lars Holden3, Marit Holden3, Knut Liestøl1,5, Trevor Clancy2, Egil Ferkingstad3, Morten Johansen6, Vegard Nygaard6, Eivind Tøstesen6, Arnoldo Frigessi3,7, Eivind Hovig1,2,3,6*

Abstract

The immense increase in the generation of genomic scale data poses an unmet analytical challenge, due to a lack of established methodology with the required flexibility and power. We propose a first principled approach to statistical analysis of sequence-level genomic information. We provide a growing collection of generic biological investigations that query pairwise relations between tracks, represented as mathematical objects, along the genome. The Genomic HyperBrowser implements the approach and is available at http://hyperbrowser.uio.no.

Rationale

The combination of high-throughput molecular techniques and deep DNA sequencing is now generating detailed genome-wide information at an unprecedented scale. As complete human genomic information at the detail of the ENCODE project [1] is being made available for the full genome, it is becoming possible to query relations between many organizational and informational elements embedded in the DNA code. These elements can often best be understood as acting in concert in a complex genomic setting, and research into functional information typically involves integrational aspects. The knowledge that may be derived from such analyses is, however, presently only harvested to a small degree. As is typical in the early phase of a new field, research is performed using a multitude of techniques and assumptions, without adhering to any established principled approaches. This makes it more difficult to compare, reproduce and realize the full implications of the various findings.

The available toolbox for generic genome scale annotation comparison is presently relatively small. Among the more prominent tools are those embedded within the genome browsers, or associated with them, such as Galaxy [2], BioMart [3], EpiGRAPH [4] and UCSC Cancer Genomics Browser [5]. BioMart at this point mostly offers flexible export of user-defined tracks and regions. Galaxy provides a richer, text-centric suite of operations. EpiGraph presents a solid set of statistical routines focused on analysis of user-defined case-control regions. The recently introduced UCSC Cancer Genomics Browser visualizes clinical omics data, as well as providing patient-centric statistical analyses.

We have developed novel statistical methodology and a robust software system for comparative analysis of sequence-level genomic data, enabling integrative systems biology, at the intersection of genomics, computational science and statistics. We focus on inferential investigations, where two genomic annotations, or tracks, are compared in order to find significant deviation from null-model behavior. Tracks may be defined by the researcher or extracted from the sizable library provided with the system. The system is open-ended, facilitating extensions by the user community.

Results

Overview

Our system is based on an abstract representation of generic genomic elements as mathematical objects. Hypotheses of interest are translated into mathematical relations. Concepts of randomization and track structure preservation are used to build complex problem-specific null models of the relation between two tracks. Formal inference is performed at a global or local scale, taking confounder tracks into account when necessary (Figure 1).

* Correspondence: ehovig@ifi.uio.no
1 Department of Informatics, University of Oslo, Blindern, 0316 Oslo, Norway
Full list of author information is available at the end of the article

© 2010 Sandve et al; licensee BioMed Central Ltd. This is an open access article distributed under the terms of the Creative Commons Attribution License (http://creativecommons.org/licenses/by/2.0), which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.
Abstract representation of genomic elements

A genome annotation track is a collection of objects of a specific genomic feature, such as genes, with base-pair-specific locations from the start of chromosome 1 to the end of chromosome Y. Tracks vary in biological content, but also in the form of the information they contain. A track representing genes contains positional information that can be reduced to ‘segments’ (intervals of base pairs) along the genome. A track of SNPs can be reduced to points (single base pairs) on the genome. The expression values of a gene, or the alleles of a SNP, are non-positional information parts and are attributed as ‘marks’ (numerical or categorical) to the corresponding positional objects, that is, segments or points. Finally, a track of DNA melting assigns a temperature to each base pair, describing a ‘function’ on the genome. We thus define five genomic types: unmarked points (UP), marked points (MP), unmarked segments (US), marked segments (MS), and functions (F).

Figure 1 Flow diagram of the mathematics of genomic tracks. Genomic tracks are represented as geometric objects on the line defined by the base pairs of the genome sequence: (unmarked (UP) or marked (MP)) points, (unmarked (US) or marked (MS)) segments, and functions (F). The biologist identifies the two tracks to be compared, and the Genomic HyperBrowser detects their type. The biological question of interest is stated in terms of mathematical relations between the types of the two tracks. The relevant questions are proposed by the system. The biologist then selects the question and needs to specify the null hypothesis. For this purpose she is called to decide about what structures are preserved in each track, and how to randomize the rest. Thereafter, the Genomic HyperBrowser identifies the relevant test statistics, and computes actual \( P \)-values, either exactly or by Monte Carlo testing. Results are then reported, both for a global analysis, answering the question on the whole genome (or area of study), and for a local analysis. Here, the area is divided into bins, and the answer is given per bin. \( P \)-values, test-statistic, and effect sizes are reported, as tables and graphics. Significance is reported when found, after correction for multiple testing.
marked segments (MS) and functions (F). These five types completely represent every one-dimensional geometry with marks.

Catalogue of investigations
We translate biological hypotheses of interest into a study of mathematical relations between genomic tracks, leading to a large collection of possible generic investigations.

Consider the relation between histone modifications and gene expression, as investigated by visual inspection in [6] (Figure S1 in Additional file 1). The question is whether the number of nucleosomes with a given histone modification (represented as type UP), counted in a region around the transcription start site (TSS) of a gene, correlates with the expression of the gene. The second track is represented as marked segments (MS). This study of histone modifications and gene expressions can then be phrased as a generic investigation between a pair of tracks (T1, T2) of type UP and MS: are the number of T1 points inside T2 segments correlated with T2 marks? Figure 2 shows the results when repeating this analysis for all histone modifications studied in [6], and different regions around the TSS. See Section 1 in Additional file 1 for a more detailed example investigation, analyzing the genome coverage by different gene definitions.

In the context of the catalogue of investigations, the genomic types are minimal models of information content. In the above example, nucleosome modifications are only used for counting, and thus considered unmarked points (UP), even though they are typically represented in the file system as marked or unmarked segments. As the gene-related properties of interest are the genome segments in which the nucleosomes are counted, as well as the corresponding gene expression values (marks), T2 is of the type marked segments (MS). The choice of genomic type clarifies the content of a track, and also restricts which analyses are appropriate. Investigations regarding the length of the elements of a track are, for instance, relevant for genes, but not for SNPs and DNA melting temperatures.

The five genomic types lead to 15 unordered pairs (T1, T2) of track type combinations, with each combination defining a specific set of relevant analyses. For instance, the UP-US combination defines several investigations of potential interest: are the T1 points falling inside the T2 segments more than expected by chance? Do the points accumulate more at the borders of the segments, instead of being spread evenly within? Do the points fall closer to the segments than expected? A growing collection of abstract mathematical versions of biological questions is provided. We have currently implemented 13 different analyses, filling 8 of the 15 possible combinations of track types (see Additional file 2 for mathematical details). Note that information reduction of a track to a simpler type (for example, segments to points) may open up additional analytical opportunities, and are handled dynamically by the system - for example, by treating segments as their middle points.

Global and local inference
A global analysis investigates if a certain relation between two tracks is found in a domain as a whole. A local analysis is based on partitioning the domain into smaller units, called bins, and performing the analysis in each unit separately. Local analysis can be used to investigate if and where two tracks display significant concordant or discordant behavior, and thus be used to generate hypotheses on the existence of biological mechanisms explaining such perturbations. Local investigations may also be used to examine global results in more detail. The length of each bin defines the scale of the analysis. Inference is then based on the computation of P-values, locally in each bin, or globally, under the null model.

To illustrate the value of local analysis, we consider viral integration events in the human genome. These may result in disease and may also be a consequence of retroviral gene therapy. Derse et al. [7] examined integration for six types of retroviruses, with different viral integrases, thus having different integration sites (type UP). Using these data, we asked whether there are hotspots of integration inside 2-kb flanking regions of predicted promoters (type US), that is, whether and where the points are falling inside the segments more than expected by chance. Figure 3 displays the hotspots as calculated P-values in bins across the genome, using the subset of murine leukemia virus (MLV) sites. We find locations of increased integration, thus generating hypotheses on the role of integration site sequences and their context.

Local analysis may be used to avoid drawing incorrect conclusions from global investigations. Consider the repressive histone modification H3K27me3 as studied in [8]. Data from ChIP-chip experiments on mouse chromosome 17 were analyzed, finding that H3K27me3 falls in domains that are enriched in short interspersed nuclear element (SINE) and depleted in long interspersed nuclear element (LINE) repeats. Using the line of enquiry raised in [8], we asked whether H3K27me3 regions (type US) significantly overlap with SINE repeats (type US), but here using formal statistical testing at the base pair level. The chosen null model only allows local rearrangements of genomic elements (for more detail, see next section). This preserves local biological structure, but allows for some controlled level of randomness.

Performing this test globally on the whole chromosome 17 leads to rejection of the null hypothesis ($P = 10^{-4}$),
in line with [8]. However, a local analysis leads to a deeper understanding. At a 5-Mbp scale, no significant findings were obtained in any of the 19 bins (10% false discovery rate (FDR)-corrected). The frequency of H3K27me3 segments varies considerably along chromosome 17 (Figure S2 in Additional file 1), which may cause the observed discrepancy between local and global results.

Precise specification of null models
A crucial aspect of an investigation is the precise formalization of the null model, which should reflect the combination of stochastic and selective events that constitutes the evolution behind the observed genomic feature.

Consider again the example of H3K27me3 versus repeating elements. In the chosen null model, we preserved the repeat segments exactly, but permuted the positions of the H3K27me3 segments, while preserving segment and intersegment lengths. We then computed the total overlap between the segments, and used a Monte Carlo test to quantify the departure from the null model. The effect of using alternative null models is shown in Table 1. The null model examined in the first column, which does not preserve the dependency between neighboring base pairs, produces lower P-values. Unrealistically simple null models may thus lead to false positives. In fact, two simulated independent tracks may appear to have a significant association if their individual characteristics are not appropriately modeled (Section 2 in Additional file 1). In this example, the choice between the biologically more reasonable null models is difficult. The two other columns of Table 1 include models that preserve more of the biological structure. The fact that these models do not lead to clear rejection of the null hypotheses suggests that we in this case lack strong evidence against the null hypothesis. Thus, examining the results obtained for a set of different null models may often contribute important information. The null model should reflect biological realism, but also allow sufficient variation to permit the construction of tests. A set of simulated synthetic tracks is provided as an aid for assessing appropriate null models (Additional file 3).

The Genomic HyperBrowser allows the user to define an appropriate null model by specifying (a) a preservation...
rule for each track, and (b) a stochastic process, describing how the non-preserved elements should be randomized. Preservation fixes elements or characteristics of a track as present in the data. For each genomic type, we have developed a hierarchy of less and less strict preservation rules, starting from preserving the entire track exactly (Section 3 in Additional file 1). For example, these preservation options for unmarked segments can be assumed: (i) preserve all, as in data; (ii) preserve segments and intervals between segments, in number and length, but not their ordering; (iii) preserve only the segments, in number and length, but not their position; (iv) preserve only the number of base pairs in segments, not segment position or number. Depending
The relation between two tracks of interest may often be modulated by a third track. Such a third track may act as a confounder, leading, if ignored, to dubious conclusions on the relation between the two tracks of interest.

Table 1 Significant bins of the overlap test between H3K27me3 segments and SINE repeats under various null models

<table>
<thead>
<tr>
<th>Tracks to randomize</th>
<th>Preserve total number of base pairs covered</th>
<th>Preserve segment lengths, but randomize position</th>
<th>Preserve segment and intersegment lengths, but randomize positions</th>
</tr>
</thead>
<tbody>
<tr>
<td>H3K27me3</td>
<td>10/19</td>
<td>1/19</td>
<td>0/19</td>
</tr>
<tr>
<td>SINE</td>
<td>10/19</td>
<td>5/19</td>
<td>4/19</td>
</tr>
<tr>
<td>H3K27me3 and SINE</td>
<td>10/19</td>
<td>5/19</td>
<td>4/19</td>
</tr>
</tbody>
</table>

The number of significant bins of the overlap test between H3K27me3 segments and SINE repeats under different preservation and randomization rules for the null model. The test was performed in 19 bins on mouse chromosome 17, with the MEFB1 cell line. (Use of the MEFF cell line gave similar results; Table S2 in Additional file 1). In this case, less preservation of biological structure leads to smaller P-values. Also, randomizing the SINE track gave smaller P-values than randomizing the H3K27me3 track (or both).

Confounder tracks

The relation between two tracks of interest may often be modulated by a third track. Such a third track may act as a confounder, leading, if ignored, to dubious conclusions on the relation between the two tracks of interest.

Consider the relation of coding regions to the melting stability of the DNA double helix. Melting forks have been found to coincide with exon boundaries [11-15]. Although few studies have reported statistical measures of such correlation [11], the correlation is confirmed by a straightforward investigation. Tracks (type F) representing the probabilities of melting fork locations [16] in *Saccharomyces cerevisiae*, were compared to tracks containing all exon boundaries (Figure 4). We asked if the melting fork probabilities (P) were higher than expected at the exon boundaries (E) than elsewhere. In the null model, the function was conserved, while points were uniformly randomized in each chromosome. Monte Carlo testing was carried out on the chromosomes separately, giving P-values <0.0005 (Table S3 in Additional file 1). In the absence of a confounder, it is thus tempting to conclude that there is an interesting relation between DNA melting and coding regions, for which functional implications have been previously discussed [15,17,18].

An alternative view is that the GC content, being higher inside exons than outside, contains information about exon location that is simply carried over, or decoded, by a melting analysis, thus acting as a confounder. We have developed a methodology to investigate such situations further. Non-preserved elements of a null model can be randomized according to a non-homogeneous Poisson process with a base-pair-varying intensity, which can depend on a third (or several) modulating genomic tracks [19,20]. We have defined an algebra for the construction of intensities, where tracks are combined, to allow rich and flexible constructions of randomness (see Materials and methods).

To investigate the influence of GC content on the exon-melting relation, we first generated a pair of custom tracks (type F), assigning to each base the value given by the GC content in the 100-bp left and right flanking regions, respectively, weighted by a linearly decreasing function. These two functions were used, together with the exon boundary track, to create an intensity curve proportional to the probability of exon points, given GC content (see Materials and methods). When performing the same analysis as before, but now using the null model based on this intensity curve (rather than assuming uniformity), a significant relationship was found in only one yeast chromosome (Table S3 in Additional file 1). In conclusion, there is a melting-exon relationship in yeast, but it may simply be a consequence of differences in GC content at the exon boundaries (high GC inside, low GC outside), which may exist for biological reasons not involving melting fork locations.

Resolving complexity: system architecture

The Genomic HyperBrowser is an integrated, open-source system for genome analysis. It is continually evolving, supporting 28 different analyses for significance testing, as well as 62 different descriptive
statistics. The system currently hosts 184,500 tracks. Most of these represent literature-based information, previously mostly utilized in network-based approaches [21]. As natural language based text mining allows for the identification of a wide variety of biological entities, we have generated tracks representing genomic locations associated with terms for the complete gene ontology tree, all Medical Subject Heading (MeSH) terms, chemicals, and anatomy.

The system is implemented in Python [22], a high-level programming language that allows fast and robust software development. A main weakness of Python compared to languages like C++ is its slower performance. Thus, a two-level architecture has been designed. At the highest level, Python objects and logic have been used extensively to provide the required flexibility. At the base-pair level, data are handled as low-level vectors, combining near-optimal storage with efficient indexing, allowing the use of vector operations to ensure speed. Interoperability with standard file formats in the field [23] is provided by parallel storage of original file formats and preprocessed vector representations. To reduce the memory footprint of analyses on genomewide data, an iterative divide-and-conquer algorithm is automatically carried out when applicable. A further speedup is achieved by memoizing intermediate results to disk, automatically retrieving them when needed for the same or different analyses on the same track(s) at any subsequent time, by any user.

The system provides a web-based user interface with a low entry point. However, the complex interdependencies between the large body of available tracks, a number of syntactically different analyses, and a range of choices for constructing null models, all pose challenges to the concepts of simplicity and ease of use. In order to simplify the task of making choices, a step-wise approach has been implemented, displaying only the relevant options at each stage. This guided approach hides unnecessary complexities from the researcher, while confronting her with important design choices as needed. We rely on a dynamic system to infer appropriate options, aiding maintenance. The list of selectable tracks is based on scans of available files on disk. The list of relevant questions is based on short runs of all implemented analyses, using a minimal part of the actual data from the selected tracks. For each analysis, a set of relevant options is defined. The dynamics of the system also provides automatic removal of analyses that fail to run, enhancing system robustness.

Allowing extensibility along with efficiency and system dynamics is a challenge. The complexities of the software solutions are hidden in the backbone of the system, simplifying coding of statistical modules. Each module declares the data types it supports and which results are needed from other modules. The backbone automatically checks whether the selected tracks meet the requirements, and if so, makes sure the intermediate computations are carried out in correct order. Redundant computations are avoided through the use of a RAM-based memoization scheme. The system also provides a component-based framework for Monte Carlo tests, where any test statistic can be combined with any relevant randomization algorithm, simplifying development. In addition, a framework for writing unit and integration tests [24] is included. Further details on the system architecture are provided in Section 4 in Additional file 1.

Step-by-step guide to HyperBrowser analysis
One of the main goals of the Genomic HyperBrowser is to facilitate sophisticated statistical analyses. A range of textual guides and screencasts are available in the help section at the web page, demonstrating execution of various analyses, how to work with private data, and more. To give an impression of the user experience, we here provide a step-by-step guide to the analysis of broad local enrichment (BLOC) segments versus SINE repeats, as discussed in the section on 'Precise specification of null models'.

First, we open ‘hyperbrowser.uio.no’ in a web browser and we select the ‘Perform analysis’ tool under ‘The Genomic HyperBrowser’ in the left-hand menu. We select the mouse genome (mm8) and continue to select tracks of interest. As the first track, we select ‘Chromatin’-‘Histone modifications’-‘BLOC segments’-’MEFB1’. These are the BLOC segments according to the algorithm of Pauler et al. [8] for the MEFB1 cell line. As the second track, we select ‘Sequence’-‘Repeating elements’-’SINE’. Now that both tracks have been selected, a list of relevant investigations is presented in the interface (that is, investigations that are compatible with the genomic types of the two tracks: US versus US). We select the question of ‘Overlap?’ in the ‘Hypothesis testing’ category, and the options relevant for this analysis are subsequently displayed in the interface. The different choices for ‘Null model’ will produce the various numbers in Table 1 (six different choices are directly available from the list. The other variants can be achieved by reversing the selection order of the tracks). The original BLOC paper [8] focused on chromosome 17. We want to perform a local analysis along this chromosome, avoiding the first three megabases that are centromeric. Under ‘Region and scale’ we thus choose to ‘Compare in’ a custom specified region, writing ‘chr17:3m-’ as ‘Region of the genome’ and writing ‘5 m’ (5 megabases) as ‘Bin size’. Clicking the ‘Start analysis’ button will then perform an appropriate statistical test according to the selected null model assumption, and output textual and graphical
results to a new Galaxy history element. Figure 5a shows the user interface covering all selections above and Figure 5b shows the answer page that results from this analysis.

This example assumed the BLOC segments were already in the system. If not, they could simply be uploaded to the Galaxy history and then selected in the first track menu as `\texttt{\textasciitilde\textasciitilde From history (bed, wig) \textasciitilde\textasciitilde [your BLOC history element]}`. For information on how to use the Galaxy system, we refer to the Galaxy web site [25].

**Discussion**

The current leap in high-throughput sequencing technology is opening the way for a range of genome-wide annotations beyond the presently abundant gene-centric data. Not least, chromatin-related data are becoming increasingly important for understanding higher-level organization and regulation of the genome [26].

As is typical for a subfield that has not reached maturation, analysis of new massive sequence-level data is performed on a per-project basis. For instance, a paper on the ENCODE project describes how inference can be done by Monte Carlo testing, sampling bins for one of the real tracks at random genome locations under the null hypothesis [1]. Independently, a newer study of histone modifications instead permuted bins of data for one of the tracks [27]. Although genomic visualization tools have been available for several years, few generic tools exist for inference at the sequence level.

The following aspects distinguish our work from currently available systems. First, we focus on genomic information of a sequential nature, that is, with specific base-pair locations on a genome, and thus not restricted to only genes. Second, it focuses on the comparison of pairs of genomic tracks, possibly taking others into account through the concept of intensity tracks. Third, all comparisons are performed using formal statistical testing. Fourth, we provide analyses on any scale, from genome-wide studies to miniature investigations on particular loci. Fifth, we offer flexible choices of null models for exploration and choice where relevant. Finally, we provide a user interface where the user describes the data and the null models, while the system based on this chooses the appropriate statistical test. Comparing this to the EpiGRAPH and Galaxy frameworks, which we believe are the closest existing systems, we find that both require substantial technical expertise when choosing the correct analysis and options. EpiGRAPH is focused on a specific type of scenario that, according to our cataloguing, amounts to the comparison of unmarked points or segments versus categorically marked segments (with mark being case or control). Galaxy provides a simple user interface, is rich in tools for manipulating and analyzing datasets of diverse formats, but has little support for formal statistical testing. Note also that our system is tightly connected to Galaxy and can make use of all the tools provided within Galaxy.

We provide tools for abstraction and cataloguing of what we believe are typical questions of broad interest.
The abstractions of genomic data, the proposing of prototype investigations, and the careful attention given to null models simplifies statistical inference for a range of possible research topics. Our approach invites researchers to build relevant null models in a controlled manner, so that specific biological assumptions can be realistically represented by preservation, randomness and intensity based confounders. In addition, time used for repetitive tasks like file parsing and calculation of descriptive statistics may be significantly reduced.

Our system is highly extensible. The software is open source, inviting the community to add new investigations and tools. Attention has been given to component-based coding and simple interfaces, facilitating extensions of the system.

The highly specialized nature of many research investigations poses a major challenge for a generic system such as the one presented here. Even though a range of analyses and options are provided, chances are that at a given level of complexity, functionality beyond what is provided by a generic system will be needed. Still, the time and effort used to reach such a point may be shortened considerably, and it should in many cases be possible to meet demands through custom extensions.

Genomic mechanisms commonly involve more than two tracks, and the current focus on pair-wise interrogations is limiting. Our methodology allows the incorporation of additional tracks through the concept of an intensity track that modulates the null hypothesis, acting as a confounder. However, the investigation of genuine multi-track interactions is not yet possible within the system, as complex modeling and testing of multiple dependencies will be required.

Attention should be given to the trade-off between fine resolution and lack of precision. When large bins are considered, there may be too little homogeneity, while small bins may contain too little data. There is also an unresolved trade-off relating to preservation of tracks in null-hypotheses construction: too little preservation may give unrealistically small P-values, while too strong preservation may give too limited randomness.

On a more specific note, a set of tissue-specific analytical options would be beneficial with respect to many types of experimental data - for example, chromatin, expression and also gene subset tracks. Such options are now under development.

Novel sequencing technologies are instrumental in realizing the personalized genomes [28], and with them the task of identifying phenotype-associated information contained in each genome. An imminent challenge in understanding cellular organization is that of the three dimensions of the genome. While a number of genomes have been sequenced, and a number of important cellular elements have been mapped on a linear scale, the mapping of the three-dimensional organization of the DNA and chromatin in the nucleus is still only in its beginnings. Consequently, the impact of this organization on cell regulation is still largely unresolved. However, the advent of methods like Hi-C [29] permits detailed maps of three-dimensional DNA interactions to be combined with coarser methods of mapping of other elements. It appears that looking simultaneously at multiple scales seems important for understanding the dynamics of different functional aspects, from chromosomal domains down to the nucleosome scale. The need for taking multiple scales into account has recently been emphasized in both theoretical and analytical settings [30,31]. Consequently, statistical genomics needs to consider several scales when proper analytical routines are developed. Our approach is open to three-dimensional extensions, where the bins, which are flexibly selected in the system, will become three-dimensional volumes, and local comparison will be within each volume. What appears much more complex is the level of dependence of such volumes. But as the three-dimensional organization of the genome will become increasingly known, appropriate volume topologies will be possible, so that neighboring volumes representing three-dimensional contiguity may be used as a basis for statistical tests.

Conclusions

By introducing a generic methodology to genome analysis, we find that a range of genomic data sets can be represented by the same mathematical objects, and that a small set of such objects suffice to describe the bulk of current data sets. Similarly, a range of biological investigations can be reduced to similar statistical analyses. The need for precise control of assumptions and other parameters can furthermore be met by generic concepts such as preservation and randomization, local analysis (binning) and confounder tracks.

Applying these ideas on a sample set of genomic investigations underlines that the generic concepts fit naturally to concrete analyses, and that such a generic treatment may expose vagueness of biological conclusions or expose unforeseen issues. A re-analysis of the relation between BLOC segments of histone modification and SINE repeats shows that conclusions regarding direct overlap at the base-pair level depends on the randomizations used in the significance analysis. Using biologically reasonable null models, the correspondence between BLOC segments and SINE repeats appears not to be due to overlap at the base-pair level, but rather seems to be due to local variation in intensities of both tracks. This does not directly oppose the original conclusions, but brings further insight into the nature of the relation. Similarly, an analysis of the relation between DNA melting and exon location confirms the
conclusion from previous studies that exon boundaries coincide with gradients of melting temperature. However, taking GC content into account as a possible confounder, the analysis does not suggest a direct functional relation between melting and exons. Instead, it suggests that the association is due to the relationship of both exons and melting tracks to GC content.

We believe the generic concepts and challenges identified by our work will trigger community efforts to improve genome analysis methodology. The Genomic HyperBrowser demonstrates the feasibility of applying our approach to large-scale genomic datasets, providing a concrete basis for further research and development in inferential genomics. We thus consider the solutions presented here more like a start than an end of this important endeavor.

Materials and methods

Statistical methods

A track is defined over the whole genome or only in parts of it, masking away the rest. In a local analysis, statistical tests are performed in each bin with sufficient sample size. Resizing of bins allows for localization of events (similarities, differences, and so on, between the two tracks) with flexible precision. Preservation rules lead to conditional P-values that are not necessarily ordered, even if the preservation mechanism is incremental. Statistical tests have been tried on simulated data, also when model assumptions are not completely fulfilled. Standard Monte Carlo requires deciding on the number of Monte Carlo samples. We suggest at least two to five times the number of tests, in order to allow for FDR adjustment. Additionally, we adopt sequential Monte Carlo, where the algorithm continues sampling until the observed statistic has been exceeded a given number of times (say 20) [9]. This gives better estimates of small P-values with overall reduced computations. Intensity tracks are used to define non-standard null hypothesis. Several strategies for building intensity curves are described in Section 3 in Additional file 1. Intensity curves allow performing randomizations that mimic another track (or a combination of tracks), useful to account for confounding effects. For unmarked points, the intensity curve can be any regular function $\lambda_0(b)$ where b is the position along, say, a chromosome. If $\lambda_0(b) = c$ (constant), points are uniformly distributed. As another example, $\lambda_0(b)$ can be a kernel density estimate based on the track of observed points. In general, the intensity $\lambda_0(b)$ may depend on several different tracks $g_1, g_2, ..., g_k$ through a function $s$, so that $\lambda_0(b) = s(g_1(b), g_2(b), ..., g_k(b))$, for example, $\lambda_0(b) = c + \Sigma g_i(b)$. An important case that requires a special choice of intensity track is when the comparison between two tracks $T_1$ and $T_2$ might be confounded by a third, confounder, track $T_3$. This is discussed in further detail in Section 5 in Additional file 1 for the melting-exon example, where each track depends on a function of the GC content.

Software system

The Genomic HyperBrowser [30] is implemented in Python [22], version 2.7. It runs as a stand-alone application tightly connected to the Galaxy framework [2], using the version dated 2010-10-04. The user interface is based on Mako templates for Python [32], version 0.2.5, and Javascript library Jquery [33], version 1.4.2. The software uses NumPy [34], version 1.5.1rc1, for disk based vector mapping and fast vector operations. R [35], version 2.10.1, is used for plotting and basic statistical routines, using the RPy API [36], version 1.0.3. The software is open source and freely available, using GPL [37] version 3, and can be downloaded from [30]. The Genomic HyperBrowser runs on a dedicated Linux server, with large computations offloaded to the Titan cluster [38].

Biological example: histone modifications versus gene expression

Raw histone modification data [39] were preprocessed using the NPS (Nucleosome Positioning from Sequencing) software [40], using peak detection, leading to nucleosome positioning information as short segments, treated as unmarked points (UP). Raw microarray expression values [41] were used to represent gene expression, in line with [6]. Direct comparison of the expression levels of individual probes is not generally justified. As Barski et al. [6] compares sets of 1,000 genes each, the direct comparison of values between groups may be justified by noise averaging (although not discussed in [6]). Using Kendall’s rank correlation test, a similar reduction of error is obtained. Detailed correlation values for the different histone modifications are given in Table S1 in Additional file 1. The distribution of histone modifications relative to TSS is given for two different modifications in Figure S4 in Additional file 1.

Biological example: histone modifications versus repeating elements

ChiP-seq data on histone modification [39,42] were preprocessed using the SICER software [43], which returns clusters of neighboring nucleosomes as islands unlikely to have appeared by chance, using an appropriate random background model. These clusters are treated as unmarked segments (US). The ChiP-chip data of H3K27me3 positions were obtained directly from Pauler et al. [8], and were preprocessed by them using their BLOCs software, which returns broad local enrichments, also treated as unmarked segments (US). Detailed overlap results between repeats and different histone modification sources are given in Table S2 in Additional file 1.
Biological example: exons versus DNA melting

The melting fork probability tracks \( P_L(x) \) and \( P_R(x) \) used in this study were obtained using the Poland-Scheraga model [44]. To make the correction for GC content, a pair of GC-based function tracks, \( L(x) \) and \( R(x) \), were created using a moving window approach. Let \( E_L \) (\( E_R \)) be the left (right) exon boundaries. For testing the melting-exon relation in tracks \( (E_L, P_L) \), an intensity track was created based on \( L(x), R(x) \) and \( E_L \) (and similarly for tracks \( (E_R, P_R) \)). See Section 5 in Additional file 1 for more details.

Additional material

Additional file 1: Supplementary material. Miscellaneous supplementary material: gene coverage example. On the importance of realistic null models. On mathematics of genomic tracks. On system architecture. On Exon DNA melting example. Supplementary figures and tables.

Additional file 2: Statistical tests. Detailed description of the statistical tests implemented in the software system.

Additional file 3: Supplementary note on simulation. Description of basic algorithms for simulating synthetic tracks, used to assess statistical tests.

Abbreviations

BLOC: broad local enrichment; bp: base pair; F: function; FDR: false discovery rate; kb: kilo base pairs; LP: long interspersed nuclear element; Mbp: mega base pairs; MP: marked point; MS: marked segment; SINE: short interspersed nuclear element; SNP: single-nucleotide polymorphism; TSS: transcription start site; UP: unmarked point; US: unmarked segment.

Acknowledgements

We gratefully acknowledge CHIP-chip data provision from Florian M Pauer, and helpful comments on the manuscript from Magnus Lie Hetland, Sylvia Richardson and Håvard Rue. Guro Nilsen is acknowledged for some plotting functions, and Peter Wiedswang for administrative assistance. We thank the Scientific Computing Group at USIT for providing friendly and helpful assistance on system administration. We also thank PubGene, Inc. for kind assistance in the development of literature tracks. Additional funding was kindly provided by ENBIO, UiO and Heise Sar-Ott. This work was performed in association with ‘Statistics for Innovation’, a Centre for Research-Based Innovation funded by the Research Council of Norway.

Author details

1Department of Informatics, University of Oslo, Blindern, 0316 Oslo, Norway.
2Department of Tumor Biology, The Norwegian Radium Hospital, Oslo University Hospital, Montebello, 0310 Oslo, Norway.
3Statistics For Innovation, Norwegian Computing Center, 0314 Oslo, Norway. 4Department of Mathematics, University of Oslo, Blindern, 0316 Oslo, Norway. 5Centre for Cancer Biomedicine, The Norwegian Radium Hospital, Oslo University Hospital, Montebello, 0310 Oslo, Norway. 6Institute for Medical Informatics, The Norwegian Radium Hospital, Oslo University Hospital, Montebello, 0310 Oslo, Norway. 7Department of Biostatistics, Institute of Basic Medical Sciences, University of Oslo, Blindern, 0317 Oslo, Norway.

Authors’ contributions

GKS, AF and EH conceived the approach, GKS, SG and MJ developed the software, GKS, SG, HR, TC, VN and EH developed novel track types, IKG, LH, MH, KL, EF and AF developed the statistical concepts, GKS, SG and HR tested and validated the system, and GKS, SG, HR, ET and EH developed the biological examples. All authors participated in the manuscript development, and read and approved the final manuscript.

Competing interests

Elvind Hovig is a shareholder of PubGene, Inc. All other authors declare that they have no competing interests.

Received: 27 August 2010 Revised: 8 December 2010 Accepted: 23 December 2010 Published: 23 December 2010

References

22. Python Reference Manual. [http://docs.python.org/release/2.5.2/ref/ref/ref.html](http://docs.python.org/release/2.5.2/ref/ref/ref.html)
25. Galaxy. [http://main.g2.bx.psu.edu/](http://main.g2.bx.psu.edu/)


28. 1000 Genomes. [http://www.1000genomes.org/]


33. jQuery. [http://jquery.com].


36. RPY a robust Python interface to the R Programming Language. [http://rpy.sf.net].


38. Titan. [http://www.notur.no/hardware/titan/].


Cite this article as: Sandve et al: The Genomic HyperBrowser: inferential genomics at the sequence level. Genome Biology 2010, 11:R121.
Identifying elemental genomic track types and representing them uniformly

Sveinung Gundersen1, Matúš Kaláš2,3, Osman Abul4, Arnoldo Frigessi5,6, Eivind Hovig1,7,8 and Geir Kjetil Sandve8*

Abstract

Background: With the recent advances and availability of various high-throughput sequencing technologies, data on many molecular aspects, such as gene regulation, chromatin dynamics, and the three-dimensional organization of DNA, are rapidly being generated in an increasing number of laboratories. The variation in biological context, and the increasingly dispersed mode of data generation, imply a need for precise, interoperable and flexible representations of genomic features through formats that are easy to parse. A host of alternative formats are currently available and in use, complicating analysis and tool development. The issue of whether and how the multitude of formats reflects varying underlying characteristics of data has to our knowledge not previously been systematically treated.

Results: We here identify intrinsic distinctions between genomic features, and argue that the distinctions imply that a certain variation in the representation of features as genomic tracks is warranted. Four core informational properties of tracks are discussed: gaps, lengths, values and interconnections. From this we delineate fifteen generic track types. Based on the track type distinctions, we characterize major existing representational formats and find that the track types are not adequately supported by any single format. We also find, in contrast to the XML formats, that none of the existing tabular formats are conveniently extendable to support all track types. We thus propose two unified formats for track data, an improved XML format, BioXSD 1.1, and a new tabular format, GTrack 1.0.

Conclusions: The defined track types are shown to capture relevant distinctions between genomic annotation tracks, resulting in varying representational needs and analysis possibilities. The proposed formats, GTrack 1.0 and BioXSD 1.1, cater to the identified track distinctions and emphasize preciseness, flexibility and parsing convenience.

Background

Recent ChIP and high-throughput sequencing technologies are currently generating functional annotations at unprecedented speed and resolution. The availability of detailed protein binding locations, DNA methylation, histone modifications, DNA variations of individuals, and more for different tissues and conditions, provides the basis for a plethora of representational formats of genome wide data. Adding to this, new technologies for assessing the three-dimensional structure of the DNA, such as Hi-C [1], introduce the concepts of distance measures between different parts of a genome, opening up a whole new set of representational complexity.

Several efforts have been attempted at defining general formats for the textual representation of genome annotation data. One such format is the General Feature Format (GFF), currently in version 3 [2]. Other generic formats are provided in connection to the UCSC Genome Browser [3], the Browser Extensible Data format (BED), bedGraph and WIG, among others. One reason for the different formats is that different properties are required, often in order to support information related to specific domains, technologies or experimental methods. Consider for instance the BED15 format by UCSC. This is an extension of the BED format, adding 3 columns in order to represent microarray expression data [4]. Other examples are the Gene Transfer Format (GTF) [5] for gene tracks and the Genome Variation Format (GVF) [6] for DNA variant files, both based on the GFF format.
Another reason behind the proliferation of formats seems to be an issue of practicality. Certain types of genome annotations, or genomic tracks, are more efficiently and elegantly represented by certain data formats. Consider a track of DNA melting temperatures, i.e. an algorithmic prediction of the denaturation temperature for each base pair of the genome, e.g. [7]. Representing such a track in the Wiggle format (WIG) would take around 20 GB for the human genome. The exact same information could be represented in the bedGraph format, but the file size would then expand to around 100 GB. In this case, the file would contain much redundant information, such as repeated chromosome declarations, and start and end positions that are always increased by one for each line. The help pages at the UCSC Genome Browser explicitly recommend the WIG format for “dense, continuous data” and bedGraph for “continuous data that is sparse or contains elements of varying size” [8]. From this it seems that, at an abstract level, there may exist fundamental distinctions between track data, such that warrants the use of particular textual formats. We are, however, not aware of any systematic discussion of such distinctions in the literature.

Expanding on this notion of systematic distinctions between track data, it seems that such distinctions also warrant differences in which analyses are applicable. It is for instance meaningful to ask whether SNPs fall inside exons, but it is not meaningful to ask whether SNPs fall inside melting temperature. Conversely, one can ask whether SNP locations have high melting temperatures, but not whether SNPs have high exons. This indicates that there may be some form of abstract grammar, where each track defines a set of informational properties, and each analysis only makes sense on certain sets of informational properties for the tracks in question.

In this paper, we start with a clarification of basic nomenclature. We then discuss how the presence of different core informational properties of a track can be used to delineate fifteen different types of tracks at an abstract level. The fifteen track types encompass most existing data formats, in addition to open up for data sets making use of cross-positional linking, e.g. data sets based on the three-dimensional structure of DNA. We continue by reviewing common, generic formats, in tabular, XML-based, or binary form, and discuss how they fit with the proposed track types. This is followed up with the proposal of a new tabular format and an updated XML format for track data. These formats build closely on previous ones, but obey the distinctions between types of tracks. Finally, we discuss supporting tools for the proposed formats, including a code base supporting the storage of tracks in efficient binary format, illustrating how the formats can be pragmatically applied in high-speed analyses.

**Results and Discussion**

**Definitions**

A reference genome may be abstracted as a line-based coordinate system. To build on this powerful metaphor, we use the term **genomic track** (or, in short, **track**), as used by the UCSC Genome Browser [3]) to refer to a series of data units positioned on such a line. The basic informational unit is called a **track element**, that is, a unit of data with associated genomic coordinates that may or may not be explicitly specified. A track element is to be thought of as a mathematical or implementational abstraction, in tabular formats typically represented as a single data line. Although the concept of genomic tracks is most useful for describing data that refer to a single reference genome, the meaning carries easily over to datasets referring to multiple reference genomes, or to contigs or scaffolds of partially assembled genomes.

We further define a **genome feature** as a track element or set of track elements comprising a biological unit, e.g. a specific gene, of a certain feature type, e.g. genes. The term **biological unit** is to be understood broadly and should also include experimental results, algorithmic predictions and similar concepts, such as defined under **sequence feature** in the Sequence Ontology [9]. Note that a feature, e.g. a gene, may be composed of several track elements, e.g. representing the exons of that gene. Often, a complete genome annotation, i.e. features of many feature types connected to a genome, are collected into a single file. This complicates the comparison of different feature types, creating the need for filtering such a file for the appropriate feature types prior to analysis. On the other hand, restricting a track to contain only a single feature type may reduce the information. For example, the connection between genes and their exons is lost if the two feature types are stored as separate tracks. We thus define a **genomic track** more specifically as set of track elements of one or several feature types, defined over an appropriate genome-scale coordinate system, where the set of feature types constitutes a pragmatic unit for analysis. A genomic track is then, in our view, defined in relation to an analytical purpose, whether explicitly defined or only suggested; this, in contrast to a data file used mainly for storage, which should be considered more as a flat file database.

**Core informational properties of tracks**

A genomic track consists of a set of track elements and, for each element, describes a set of properties, such as an identifier, a quality score or the method used. The positional information of a track element is obligatory
for any genomic track and can be interpreted generically across tracks. The position of a track element is often encoded as a pair of start and end coordinates. However, when looking at genomic tracks from the perspective of information content, we find it fruitful to identify the positional information equivalently as the \textit{lengths} of the track elements and the \textit{gaps} between them, both measured in base pairs. As the positional information is essential and generic, we refer to gaps and lengths as \textit{core informational properties} of the track.

A genomic track may also carry a main value associated with each track element, for instance the measured expression of a gene or the copy number of a genomic region. We thus include \textit{values} among the core informational properties. This main value can be a number (e.g. the expression of a gene), a binary value (e.g. if the element is considered case or control), a category (e.g. the feature type), a character (e.g. the allele variant of a SNP), or a list of values (e.g. gene expression for a set of patients).

Lastly, a track element may be connected to other track elements located at different locations on the genome. This is critical for three-dimensional tracks, as locations that seem far apart when the DNA is unwound, could still be co-located in the nucleus. The corresponding core informational property of a track is then \textit{interconnections}. The interconnections, or edges, are either directed or undirected, possibly with an attached weight value.

\textbf{Fifteen genomic track types}

All four core informational properties (\textit{gaps, lengths, values}, and \textit{interconnections}) will not always be defined for a track. Consider, for instance, a track of viral insertion points on a genome. As it makes no sense to talk about the length of an insertion point, such a track will not have the lengths property defined. Similarly, a track of single nucleotide polymorphisms (SNPs) will only contain elements that refer to single discrete positions on the genome. The track elements will, however, have associated values denoting the respective alleles. Consider also the DNA melting map, a track where a temperature value is assigned to every base pair of the genome [7]. As temperature values, \textit{i.e.} track elements, are defined for every consecutive position of the genome, there is never any gaps between the elements. Also, the elements refer to single base pairs and have no lengths. Thus, a track of DNA melting will have neither the \textit{lengths} nor the \textit{gaps} property defined, only the \textit{values} property (denoting temperature).

Four core properties, being defined or not, gives \(2^4 = 16\) distinct combinations. Assuming that a genomic track always consists of track elements with the same core properties, we can distinguish tracks on the basis of which combination of core properties are defined. For one of the sixteen combinations, no core properties are defined. It is thus of no interest, hence reducing the set to fifteen combinations.

Looking closely at the fifteen combinations, an interesting pattern appears. Figure 1 shows an illustration of the informational contents of each combination. As every combination denotes a particular geometric configuration, strikingly distinct from the others, we refer to tracks of the different combinations as having different \textit{track types}. The concept of dividing genomic tracks into track types was partially introduced in [10], but has now been expanded from five to fifteen track types.

Looking at the top left of Figure 1 and going downward, we start at the base case where the only core informational property is the gaps between the track elements. In this case, each track element represents an exact base pair location on the genome, denoting \textit{e.g.} viral insertion sites. We call this track type \textit{Points (P)}. Adding informative values to this case, \textit{e.g.} associating SNPs with allele frequencies, we get the track type \textit{Valued Points (VP)}. In the next two cases, the lengths property is added, resulting in the track types \textit{Segments (S)} and \textit{Valued Segments (VS)}. Segments are probably the most common track type of existing tracks, representing common features such as genes or exons. Valued segments could, for instance, denote genes with associated expression levels.

Moving on, we remove the \textit{values} and \textit{gaps} properties, leaving only \textit{lengths}. Such tracks consist of segments covering all base pairs of the genome, \textit{i.e.} a partition of the genome into potentially unequal pieces. Hence, the track type is called \textit{Genome Partition (GP)}. Basic examples of this track type are the partition of a genome into chromosomes or cytobands. Adding a value to each part of a partition creates a \textit{Step Function (SF)}, covering the whole genome with values. Basic examples of such tracks are tracks denoting results of tiling microarrays, providing that any gaps or overlaps between the tiles are ignored. Removing the \textit{lengths} core property, the step function track is transformed into a track of type \textit{Function (F)}, where every base pair has an associated value. Examples of function tracks are tracks with close dependency on the genome sequence, such as GC content tracks, or predictions of melting temperatures, as outlined above. We call the seven track types outlined here \textit{basic track types}.

The fourth core informational property, \textit{interconnections}, can be envisioned as an orthogonal extension to the previous discussion. Adding interconnections, or edges, to the seven track types previously outlined (first column in Figure 1) defines linked versions of the same track types, \textit{e.g.} \textit{Linked Segments (LS)} or \textit{Linked Step Function (LSF)} (second column of Figure 1). Although
Figure 1 Illustration of the geometric properties of the fifteen track types. The base line is a genome, or a sequence, on which the tracks are defined. Vertical lines represent positions, while horizontal lines represent the lengths of the track elements. Gaps are thus illustrated by any empty areas between the elements. Values are represented by the height of the vertical lines. Interconnections are represented by arrows, the thickness of which correspond to the weight of the edge.
tracks that include interconnections are presently in little use, enough datasets exist to warrant the definition of all the linked track types, at least for completeness. For example, the recent Hi-C dataset of Dekker et al. [4] partitions the genome into 1 Mbp regions (for the genome-wide case), where each pair of regions has an associated proximity value. This dataset is then of type \textit{Linked Genome Partition (LGP)}, where every region has a weighted edge to all other regions. More traditionally, one could envision a gene/protein pathway being represented as gene segments, perhaps also with associated expression data, being linked together with directed edges representing associations (binding, activation, inhibition, etc.). This would be of type \textit{Linked Valued Segments (LVS)}. Note that a track type is considered linked if at least some track elements are interconnected.

To complete the picture, a last track type needs to be defined. If only the \textit{interconnections} core property is defined, track elements do not have gaps between them, lengths, or values. All base pairs are then track elements, with each base pair connected to other base pairs by edges, hence the name \textit{Linked Base Pairs (LBP)}. Thinking in term of graphs, all base pairs will thus be nodes, although not all nodes need to have any edges. This, in contrast to the track type \textit{linked points}, which limits the nodes to a specified set of points. The track type \textit{Linked Function (LF)} is similar to \textit{linked base pairs}, only adding an associated value to each base pair (node). The \textit{linked base pairs} track type is mostly suggestive at this point, but at least theoretically, this would be the track type of the perfect three-dimensional track, mapping the distance between all base pairs of a genome. Another example of a track of this kind is the representation of a randomization of a genome, with each edge representing the positional relocation of a base pair. We refer to the eight linked track types as the \textit{extended track types}. Figure 2 shows an overview of the relations between the fifteen track types and the combination of core informational properties defined.

---

**Figure 2** Four-dimensional matrix mapping the relations of the fifteen track types. Each dimension represents the exclusion (0) or inclusion (1) of one of the four core informational properties: gaps, lengths, values and interconnections. The track type abbreviations in the top-left box are: \textit{Genome Partition (GP), Points (P) and Segments (S)}, in the bottom-left box: \textit{Function (F), Step Function (SF), Valued Points (VP) and Valued Segments (VS)}, in the top-right box: \textit{Linked Base Pairs (LBP), Linked Genome Partition (LGP), Linked Points (LP) and Linked Segments (LS)}, and in the bottom-right box: \textit{Linked Function (LF), Linked Step Function (LSF), Linked Valued Points (LVP) and Linked Valued Segments (LVS)}. The track types with white background (with gaps) are the \textit{sparse} track types, while the ones with grey background (without gaps) are the \textit{dense} track types. See Figure 1 for a geometric illustration of the track types.
Formal model of genomic tracks

Formally, we base the discussion of track types on a specific mathematical model of genomic tracks. We treat the genomic coordinates as forming a discrete metric space on the natural numbers, defined by the discrete metric d:

\[ d(a, b) = |a - b| + 1, \quad a, b \in \mathbb{N} \]  

(1)

The genomic coordinates in the model are thus isolated points. A segment or interval starting at a position a and ending at b is defined as the subset S of natural numbers where:

\[ S(a, b) = \{ s \in S \mid a \leq s \leq b \land b > a \} \]

(2)

The length of a segment is defined by the metric d, and is equal to the number of elements in the set. The length of the segment S(1, 3) = {1, 2, 3} is thus d(1, 3) = 1 - 3 + 1 = 3 = |S(1, 3)|. Transferred to the biological domain, the length of a segment is the number of base pairs covered by the segment. The end position of a segment must be larger than the start position. We thus exclude segments of length 1 from the model, as such segments would be exactly equal to a point, e.g. the set of a single number:

\[ P(a) = \{ p \in P \mid p = a \} \]

(3)

From the set notation follows that a point P can be precisely defined as falling inside a segment S if and only if P ⊆ S. Two segments, on the other hand, may partially overlap. A function is precisely defined as a mathematical function from genomic coordinates to corresponding values, e.g. \( f: \mathbb{N} \rightarrow \mathbb{R} \). A step function is similarly a function from disjoint intervals covering the entire domain to corresponding values.

Analysis dependency on track types

As each of the fifteen track types implies a set of core informational properties, a track type also poses a limit to which analyses are appropriate for a track. It makes sense to calculate the base pair coverage of a track of genes (type: segments), but not for a track of SNPs (type: valued points), which should instead be counted. This logic also carries on to analyses applied to more than one track. Consider, for the sake of simplicity, only five of the fifteen track types. If we select two tracks, each of one of these five types, we get 15 combinations, provided that the order of the tracks is not important. Each of these combinations could then define a set of appropriate analyses. Table 1 provides analysis examples for many of the pairwise combinations of the five track types points, segments, function, valued points, and valued segments. Although assigned to a single combination of track types, an analysis may often be meaningful for a set of such combinations. For instance, asking whether the points of one track are located inside the borders of the segments of another track (points vs segments) will trivially also give meaning where one or both of the tracks has associated values (e.g. valued points vs valued segments). Also, it could give meaning to ask whether small segments of one track are located inside the borders of the segments of another track (e.g. for the segments vs segments combination). The correspondence between the track types and possible analyses are at the core of the idea of track types. Although storing data sets as efficiently as possible is an important aspect, the bioinformatics field is currently lagging more in terms of general understanding and standards for analyzing data sets in meaningful ways. It is our hope that the definition of track types will help in this regard.

Existing representational formats

Existing formats for representing genomic tracks can broadly be divided into three groups: textual formats, binary formats, and XML formats. Often textual and binary formats are closely connected, such as the SAM and BAM formats for read alignments [11]. This duality is due to the different advantages of the two forms. Textual formats are often humanly readable and simpler to parse and manipulate than their binary alternatives. The binary formats, on the other hand, are more compact and more efficient to use, often incorporating indexing schemes for fast random access to data. XML formats aim to bridge this gap by defining data structures that can exist in both textual and binary forms. Note that we limit the discussion to formats that aim at being general,

<table>
<thead>
<tr>
<th>Points</th>
<th>Segments</th>
<th>Function</th>
<th>Valued Points</th>
<th>Valued Segments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Different frequencies?</td>
<td>Located inside?</td>
<td>Higher values at locations?</td>
<td>Located in highly valued segments?</td>
<td></td>
</tr>
<tr>
<td>Segments</td>
<td>Overlap?</td>
<td>Higher values inside?</td>
<td>Categories differentially located in targets?</td>
<td></td>
</tr>
<tr>
<td>Function</td>
<td>Correlated?</td>
<td>Valued Points</td>
<td>Nearby values similar?</td>
<td></td>
</tr>
<tr>
<td>Value Points</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Valued Segments</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Examples of analyses for different combinations of track types (using only five of the fifteen defined track types). Note that many of these analyses are valid for several (though not all) combinations, and are assigned to what we consider the most typical combination for the analysis. All these analyses are carefully described significance tests [10], available online at the Genomic HyperBrowser [28].
in one form or another, thus excluding formats that are special to a particular technology or platform.

The large majority of formats for genomic data are textual, and the large majority of the textual data formats are tabular, that is, they consist of tab-separated columns. Three of the most common tabular formats are Generic Feature Format (GFF) [2], Browser Extensible Data format (BED) [4] and Wiggle Track Format (WIG) [8]. Figure 3 shows an overview of these three tabular formats, with example files.

A main reason for the popularity of tabular formats is that they are inherently simple to create and read, both manually and by computers. This has been a major asset in the field of bioinformatics because of the widespread use of both ad hoc scripting and WYSIWYG editing in spreadsheet software (such as Microsoft Excel). Still, the abundance of different formats, together with the increased complexity of particular formats, creates practical problems when e.g. creating new tools.

XML formats represent a way of letting go of the entire process of custom and explicit parsing of files. In particular when an XML format is specified by a dedicated XML Schema (abbreviated XSD, from XML Schema Definition), the data included in an XML document can be automatically transformed into convenient runtime data objects. XML formats are much used in connection with Web services, XML databases, or serializations of object models, but there have so far been only a few XML formats used for exchanging sequence-data feature. The Distributed Annotation System [12]
uses the DASGFF XML format, which is similar to the tabular GFF. Web services for feature prediction at CBS [13] have been using a common XSD-based output format that has been inspired by GFF. Numerous Web services and databases define their own XML formats for annotation data, such as the UniProt XML [14] or the ELMdb Web service [15]. BioXSD version 1.0 has defined a format for sequence features that is expressive enough to be able to substitute the majority of other feature formats [16]. The main disadvantages of using XML for genome-scale annotations have been the verbosity of the textual serialization of XML data and the large memory usage of most of the libraries parsing XML. The recent W3C standard for highly optimized binary representation of XML - the Efficient XML Interchange (EXI) format [17] - promises to solve these problems.

Binary formats are often used internally in software systems, and not necessarily provided as public formats. Some exceptions to this are the aforementioned BAM, as well as the bigBed and bigWig formats [18]. The last two formats are binary versions of the BED and WIG format, respectively, providing efficient storage and indexing capabilities, allowing users to store large tracks on their own computers, while a server requests only the parts needed for analysis or visualization. Another binary format is the USeq Compressed Binary format [19] focusing on tight compression of tabular data files of different types, while keeping them in an indexed structure.

As Figure 3 illustrates, different formats support different combinations of the core informational properties, and hence, different track types. Table 2 provides an overview of which of the basic track types are supported by existing tabular, binary and XML formats.

<table>
<thead>
<tr>
<th>Format</th>
<th>Ref.</th>
<th>Data</th>
<th>Repr.</th>
<th>P</th>
<th>S</th>
<th>VP</th>
<th>VS</th>
<th>SF</th>
<th>F</th>
<th>L</th>
<th>Strand</th>
<th>#Cols</th>
<th>Value type</th>
</tr>
</thead>
<tbody>
<tr>
<td>BED/bigBed</td>
<td>[4]</td>
<td>General</td>
<td>Tab/Bin.</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>2</td>
<td>✓</td>
<td>✓</td>
<td>3-12 Int(0-1000)/string</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BED15</td>
<td>[4]</td>
<td>Microarray</td>
<td>Tab.</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>2</td>
<td>✓</td>
<td>✓</td>
<td>15 List of floats</td>
<td></td>
<td></td>
</tr>
<tr>
<td>bedGraph</td>
<td>[4]</td>
<td>General</td>
<td>Tab.</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>4</td>
<td>✓</td>
<td>✓</td>
<td>2 Float</td>
<td></td>
<td></td>
</tr>
<tr>
<td>WIG/bigWig (fixedStep)</td>
<td>[8]</td>
<td>General</td>
<td>Tab/Bin.</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>1</td>
<td>✓</td>
<td>✓</td>
<td>2 Float</td>
<td></td>
<td></td>
</tr>
<tr>
<td>WIG/bigWig (variableStep)</td>
<td>[8]</td>
<td>General</td>
<td>Tab/Bin.</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>2</td>
<td>✓</td>
<td>✓</td>
<td>2 Float</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CNT</td>
<td>[36]</td>
<td>Copy number</td>
<td>Tab.</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>4</td>
<td>✓</td>
<td>✓</td>
<td>7 String</td>
<td></td>
<td></td>
</tr>
<tr>
<td>VCF</td>
<td>[37]</td>
<td>Variation</td>
<td>Tab.</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>≥8</td>
<td>✓</td>
<td>✓</td>
<td>9 String</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PSL</td>
<td>[4]</td>
<td>Alignment</td>
<td>Tab.</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>21</td>
<td>✓</td>
<td>✓</td>
<td>11 Int</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SAM/BAM</td>
<td>[38]</td>
<td>Alignment</td>
<td>Tab/Bin.</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>11</td>
<td>✓</td>
<td>✓</td>
<td>11 Int/strings</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BioHDF</td>
<td>[39]</td>
<td>Alignment</td>
<td>Bin.</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>11</td>
<td>✓</td>
<td>✓</td>
<td>11 Int/strings</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MAF</td>
<td>[4]</td>
<td>Multiple Alignment</td>
<td>Tab.</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>2-7</td>
<td>✓</td>
<td>✓</td>
<td>11 Float/string</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FASTA</td>
<td>[40]</td>
<td>Sequence</td>
<td>Text</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>N/A</td>
<td>✓</td>
<td>✓</td>
<td>2 Float/string</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DAS XML</td>
<td>[12]</td>
<td>General</td>
<td>XML</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>2</td>
<td>✓</td>
<td>✓</td>
<td>N/A Float</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BioXSD 1.0</td>
<td>[16]</td>
<td>General</td>
<td>XML</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>11</td>
<td>✓</td>
<td>✓</td>
<td>N/A Float</td>
<td></td>
<td></td>
</tr>
<tr>
<td>USeq</td>
<td>[19]</td>
<td>General</td>
<td>Bin.</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>N/A</td>
<td>✓</td>
<td>✓</td>
<td>N/A Float/string</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Genomedata</td>
<td>[41]</td>
<td>General</td>
<td>Bin.</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>N/A</td>
<td>✓</td>
<td>✓</td>
<td>N/A Float/char</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The track type abbreviations are as follows: Points (P), Segments (S), Valued Points (VP), Valued Segments (VS), Genome Partition (GP), Step Function (SF), and Function (F). L refers to any of the linked track types. The table also denotes whether the format supports specification of strand, the number of columns of the tabular formats, and the type of the dominant value, if any.

1 Points are specified using both start and end values. There is no way of specifying that a file contains only points.

2 Only a special case of linked segments is supported, namely part-of relationships, such as en exon being a part of a gene.

3 The chosen value type refers to what may be considered the main score column of the format. The format also includes a configurable column containing values that may be extracted by specialized parsers.

4 We limit the bigBed format to the standard BED columns for simplicity, as the bigBed format is highly customizable through the use of AutoSQL configurations.

5 The float values represent a set of gene expression values from microarray experiments.

6 The values represent the possible alleles at a SNP position. Also, the allele frequencies and quality scores are reported and could be used as values.

7 E.g. the number of bases that match/do not match.

8 E.g. the mapping quality or the aligned sequence itself.

9 Links to alignments in other genomes.

10 There is no way of specifying that a record contains only points or only segments.

11 No weights are supported in BioXSD 1.0.

12 Numerical values are always signed double precision floats (8 bytes). A limited set of other value types is also allowed (e.g. sequence variation and alignments).
covered by some common formats. As each of the different groups of formats (tabular, XML, and binary) has advantages in distinct scenarios and communities, one would ideally like to select three formats that cover all track types, one from each group. Unfortunately, no common formats do. One option would be to extend an existing format to support all track types. A main reason for such an extension would be to be able to make use of the plethora of tools and parsers already available. In the case of XML formats, the existing BioXSD 1.0 format was found to be easily extensible to support all track types. In the case of tabular formats, however, the only major format to support extensions is GFF, through the attribute column. However, using GFF to represent e.g. tracks of type function would be highly impractical. Each base pair would then be represented by a data line of nine columns, wasting considerable amounts of space. The remaining option is then to create a new tabular format. In order for the introduction of a new format to be justified, such a format should have the potential to replace at least some of the existing formats, in addition to having the extensibility required to meet future needs when new types of data appear. As binary formats are often not independent formats, but typically linked to tabular ones, we will not focus on such formats here. We thus present a pair of general formats aware of all track types, one of which is tabular and the other based on XML. The tabular format, GTrack 1.0, is a new format that builds closely on the BED and WIG formats, while adding support for extensions in a similar fashion as in GFF. The XML format is a successor of the existing BioXSD 1.0 format. Besides catering to a broader user base, presenting "track type"-compliant formats of both kinds illustrates that the fundamental concepts of track type are independent of implementation. The primary goals for the formats are to support all track types systematically, to allow custom extensions, and to provide efficient storage, while at the same time focusing on simple parsing and manipulation of files.

**GTrack: Type-aware tabular format**

We here introduce a new tabular track format: the GTrack format, short for both “Genomic Track” and “Generic Track”. The GTrack format supports all fifteen previously defined track types, illustrated in Figures 1 and 2. A GTrack file includes a column specification line, specifying the names of all the columns in the file. Each track type has a one-to-one correspondence to a combination of core columns being present in the column specification line, as detailed in Table 3. The four core informational properties are represented by the four core reserved columns in such a way that the existence of each core column (start, end, value, and edges) corresponds to a core property being defined (gaps, lengths, values, and interconnections, respectively):

- Gaps are implicitly represented by the start column, i.e. it holds the start coordinate of a track element and thus marks the end of any preceding gap.
- For sparse track types, i.e. track types with gaps, length is implicitly represented by the difference between start and end columns. For dense track types (without gaps), there is no start column. The length is then the difference between the previous end position and the current. Deriving length from the end position, rather than the start position, is preferable, as a parser in the opposite case would have to read the subsequent line before concluding on the length of the current track element. The existence of the end column thus corresponds directly to the track elements having the length property.
- Although several columns in a data set may contain values of potential interest, one of these columns will typically provide a main value used in processing or analysis according to a given purpose. This focus is specified by the value column.
- The edges column contains, for each track element, a comma-separated list of id’s of other track elements which are interconnected with the element in question, in addition to values associated to the edges, e.g. weights or edge types
- A GTrack file may contain several columns containing values or edges. Users may then switch between them by simply editing the column specification line.

The edges column requires that the non-core reserved column id is present, containing a unique identifier for each track element. Three other non-core columns are specified in the GTrack format: genome, seqid and strand (see Table 3). The titles of the eight reserved columns are reserved words in the column specification line. They may appear in any order, and any number of additional columns may be specified. Figure 4 shows six example GTrack files, five of which are conversions of the example files in Figure 3. The example files illustrate the variation stemming from the different column specification lines (starting with the characters "###").

When creating the GTrack format, we have emphasized simplicity, both for creation, manual reading and automated parsing of the format. We have identified three principles towards simplicity: independence of data lines, overview of structural characteristics and equally sized lines.

The principle of independent data lines states that it should be possible to interpret each data line in a tabular format independently of its location in the file. This
is a principle followed in many common formats, e.g. GFF [2] or BED [4]. Following this principle gives several advantages. First, when creating or manipulating a file, keeping data lines independent allows the filtering and sorting of data lines while still keeping all the relevant information. Second, keeping a track element on a single line makes it easier to read for the human eye. Third, independent data lines reduce the need of automatic parsers to hold state information. The GTrack format follows the principle of independent data lines with two exceptions. First, data lines of dense track types are dependent on their positions in the file. Second, the GTrack format allows (and, in the case of dense track types, requires) the specification of bounding regions around each block of values. A bounding region specification line defines the domain of the following track elements, i.e. the region where we have information about the features modeled by the track elements. It is recommended that tracks mask out regions of a genome where nothing is known (such as centromeres or assembly gaps) using bounding regions, rather than just omitting track elements or specifying 0-values, as the difference is important for many analyses. Bounding regions unfortunately require parsers to store state information. See Figure 4A, 4C1, 4C2 and 4D for examples of bounding region specification lines (starting with the characters ‘####’).

The principle of including an overview of structural characteristics means that a track file should start with a set of configurable options that describe the structure of the data lines, in an easily readable manner. Note that many of these characteristics will, by nature, include redundant information, i.e. that could have been collected from the data lines themselves. There are several reasons for explicitly stating such characteristics. First, it gives the human reader a simple overview of the type of data stored in the file, without having to scrutinize the actual data. Second, it allows the creator of a track to validate that the file is structured in the way intended (for this purpose, we also provide a web-based validator tool [20]). Third, inclusion of structural characteristics allows parsers to be restrictive on which kind of structures to support. A quick script can then, for instance, read the header and check whether the track type is segments with no overlapping elements, failing explicitly if the header does not match this requirement. The script can then assume that the remaining file follows the asserted structure, safely ignoring the non-relevant

---

Table 3 Overview of the reserved columns in the GTrack format and their associations to track type

<table>
<thead>
<tr>
<th>Core property:</th>
<th>Gaps</th>
<th>Lengths</th>
<th>Values</th>
<th>Inter.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type of column:</td>
<td>genome</td>
<td>seqid</td>
<td>start</td>
<td>end</td>
</tr>
<tr>
<td>Track type</td>
<td>N</td>
<td>N</td>
<td>C</td>
<td>C</td>
</tr>
<tr>
<td>Linked Points (LP)</td>
<td>?</td>
<td>!</td>
<td>✓</td>
<td>.</td>
</tr>
<tr>
<td>Linked Segments (LS)</td>
<td>?</td>
<td>!</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Linked Genome Partition (LGP)</td>
<td>?</td>
<td>!</td>
<td>.</td>
<td>✓</td>
</tr>
<tr>
<td>Linked Valued Points (LVP)</td>
<td>?</td>
<td>!</td>
<td>✓</td>
<td>.</td>
</tr>
<tr>
<td>Linked Valued Segments (LVS)</td>
<td>?</td>
<td>!</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Linked Step Function (LSF)</td>
<td>?</td>
<td>!</td>
<td>.</td>
<td>✓</td>
</tr>
<tr>
<td>Linked Function (LF)</td>
<td>?</td>
<td>!</td>
<td>.</td>
<td>✓</td>
</tr>
<tr>
<td>Linked Base Pairs (LBP)</td>
<td>?</td>
<td>!</td>
<td>.</td>
<td>✓</td>
</tr>
</tbody>
</table>

C Core reserved column (defines track type)
N Non-core reserved column (reserved, but does not define track type)
✓ Column is mandatory
? Column is optional
. Column is not allowed
! Property must be present, either as a column or in a bounding region specification
1 The length is the difference between the end and the start position, or, if the start column is not present, the difference between the current end position and the previous.
2 The non-core reserved column id is required when the edges column is present.
generality of the GTrack specification. In the GTrack format, the structural characteristics are specified in header lines, starting with the characters "##". Table 4 contains an overview of all GTrack header variables. Note that header lines are optional when their values are equal to the default values. We also provide the "Expand GTrack headers" tool, which generates a GTrack file with full headers based on a supplied, incomplete GTrack file, further simplifying the process of generating header lines.

The principle of equally sized lines states that all data lines contain the same number of columns, i.e.
that all attributes have a value. Columns that do not contain information are marked with a period character. There are several advantages for this solution compared to the solution used in the GFF format, where the last column may contain a list of attributes in the format tag = value, allowing the attribute list to differ for each line. First, having equal size columns allows validation that all data lines are complete, or at least that the creator of the track has considered all attributes for all track elements. With a variable size attribute column, there is no way to check that all attributes have been considered. Second, parsing attribute lists as in the GFF format is more cumbersome, as the parser will not in advance know which attributes may appear in the file. Third, not having to repeat attribute names for all lines saves some space. Fourth, and most importantly, having the same number of columns in each data line keeps the interface of the format coherently organized, with attributes as columns and track elements as rows. As the GTrack format supports custom columns, it can completely replace the attribute solution of the GFF format.

In addition to simplicity, the GTrack format aims at being highly extensible and inter-operable. First, the ability to define columns in any order and number, provides ample options for extensibility, in addition to simplifying conversion. In many cases, converting another tabular format to GTrack is as simple as adding a column specification line. Note that basic, three-column BED files are directly compatible with the GTrack format, without the need for any modifications. Also, both 0- and 1-based indexing, in addition to the end position being inclusive or exclusive, are included in the GTrack specification, further simplifying conversion. Second, GTrack includes a strategy for making structured extensions of the format, namely the specification of subtypes. Four subtype header lines are available (see Table 4), specifying the name and version of a subtype, the URL of the subtype specification, and the strictness of adherence required by the subtype. The idea is that research

<table>
<thead>
<tr>
<th>Header variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GTrack version</td>
<td>Version of the GTrack specification used</td>
</tr>
<tr>
<td>Track type</td>
<td>Track type of the GTrack file</td>
</tr>
<tr>
<td>Value type</td>
<td>The kind of content accepted in the value column</td>
</tr>
<tr>
<td>Value dimension</td>
<td>The dimension of the content in the value column</td>
</tr>
<tr>
<td>Undirected edges</td>
<td>Whether all edges are undirected</td>
</tr>
<tr>
<td>Edge weights</td>
<td>Whether the edges have weights</td>
</tr>
<tr>
<td>Edge weight type</td>
<td>The kind of content accepted as edge weights</td>
</tr>
<tr>
<td>Edge weight dimension</td>
<td>The dimension of the edge weights</td>
</tr>
<tr>
<td>Uninterrupted data lines</td>
<td>Whether it is guaranteed that the data lines are not interrupted by bounding region specification lines or comments</td>
</tr>
<tr>
<td>Sorted elements</td>
<td>Whether it is guaranteed that all bounding regions and track elements come in sorted order</td>
</tr>
<tr>
<td>No overlapping elements</td>
<td>Whether it is guaranteed that no two track elements overlap</td>
</tr>
<tr>
<td>Circular elements</td>
<td>Whether any track elements or bounding regions cross the coordinate borders of a circular sequence</td>
</tr>
<tr>
<td>1-indexed</td>
<td>Whether the coordinates start at 1 (0 if false)</td>
</tr>
<tr>
<td>End inclusive</td>
<td>Whether the coordinates specified in the end column is included in intervals</td>
</tr>
<tr>
<td>*Value column</td>
<td>The name of the column to be used for as the ‘value’ column</td>
</tr>
<tr>
<td>*Edges column</td>
<td>The name of the column to be used for as the ‘edges’ column</td>
</tr>
<tr>
<td>*Fixed length</td>
<td>Fixed length of all track elements</td>
</tr>
<tr>
<td>*Fixed gap size</td>
<td>Fixed-size gaps between all neighboring track elements</td>
</tr>
<tr>
<td>*Fixed-size data lines</td>
<td>Whether each data line has an exact size in terms of number of characters</td>
</tr>
<tr>
<td>*Data line size</td>
<td>The size of each data line in terms of number of characters</td>
</tr>
<tr>
<td>*GTrack subtype</td>
<td>The name of the subtype of the GTrack format specification used for the file</td>
</tr>
<tr>
<td>*Subtype version</td>
<td>The version of the GTrack subtype</td>
</tr>
<tr>
<td>*Subtype URL</td>
<td>URL to a GTrack file used as a specification/model for the GTrack subtype</td>
</tr>
<tr>
<td>*Subtype adherence</td>
<td>Regulates the way a GTrack file may override the subtype specification</td>
</tr>
</tbody>
</table>

All header variables not specified in a GTrack file retain their default values.

* Defined in the extended part of the GTrack specification. See the GTrack specification (Additional file 1) for more details.
communities can define their own tabular formats, making use of a subset of the GTrack specification. Such formats could for instance be replacements of existing formats, or formats that are honed to specific technologies or tools. The header variable "subtype URL" points to a GTrack file that can be used as model for the subtype, and is intended to be read by automatic parsers. Figure 5A shows an example of such a subtype specification file, based on the example GTrack file in Figure 4A. Specifying subtype models allows the reduction of a complete GTrack header down to a minimum of one line, as shown in Figure 5B. It is our belief that allowing extensions of the GTrack format via subtypes caters for a range of future extensions, while ensuring backward compatibility. Subtypes can be defined in a range of settings, from project specific, ad hoc solutions, to the specification of generic formats. Further examples of GTrack subtypes are described in the GTrack specification (Additional file 1). A set of standard GTrack subtypes are available online [20] (including subtypes corresponding to the example files in Figure 4).

BioXSD 1.1: Enhanced and optimized XML format

BioXSD has been developed as a universal XML format for the basic types of bioinformatics data that is in particular suitable to be used with Web services [16]. It models common types of data for which a specialized XML Schema (XSD) has not been widely adopted: biomolecular sequences, alignments, sequence feature records, and references to ontologies and data resources. The BioXSD schema defines formats of data but not formats of particular XML documents, by defining XSD types but no global XML elements. BioXSD types can thus be used according to applications’ needs in applications’ own XSDs such as those in WSDL files of Web services.

BioXSD 1.0 type AnnotatedSequence can represent annotations of a biomolecular sequence or genome with any types of positioned or non-positioned features, which can be combined in one record. Although the textual serialization of XML is in general more verbose than a tabular format, already the BioXSD 1.0 has included a number of optimizations compared to traditional feature formats like GFF or BED, thanks to the tree-like structure of XML. These have been mainly:

- not repeating the reference to a sequence in every feature occurrence
- not repeating the type of feature in every feature occurrence
- representing multi-segment and multi-point feature occurrences in one feature-occurrence element

The goal of BioXSD version 1.1 has been to further improve the expressiveness of the BioXSD formats and at the same time focus on optimizations of the data size. The successor of BioXSD 1.0 AnnotatedSequence is BioXSD 1.1 type FeatureRecord. BioXSD 1.1 in general allows more types of sequence positions, distinguishing them in the same way as the tabular GTrack format. Sparse positions are segments, points (actual points or insertions), and outer positions. Dense positions have been added: dense points (function) marked-up by <nextPoint/> empty elements; and dense partition or step function marked by <nextPartition max="..."/> elements including the border position where each interval ends. However in contrast to GTrack, the different types of positions can still be freely combined within a FeatureRecord. The representation of all types of sequence positions have been refactored, simplified, and optimized. Another crucial set of optimizations allows specification of the ontologies, databases, and computational

---

**Figure 5 GTrack subtype example**

A) An ad hoc GTrack subtype specification based on the example GTrack file in Figure 4A, which is a conversion from the GFF file in Figure 3A. This and other GTrack subtypes are available from the GTrack website [20]. B) A minimal GTrack header, parsable by fully compliant GTrack parsers. Note that the "Expand GTrack headers" tool, available from the GTrack website [20], can be used to expand headers of GTrack files using subtypes, in order for such files to be used in simpler parsers that do not support the subtype functionality.
tools of interest in a condensed way for a list of feature annotations, so that they do not have to be repeated. Detailed contents of the BioXSD feature record are listed in Table 5. Examples of data represented in BioXSD 1.1 format are available at [21-25].

There is one slight difference in how the GTrack and BioXSD deal with focus of feature records. GTrack defines one operational focus of a concrete dataset. That is the reason why it allows to specify only one type of track locations and only one value column and one edges column at a time, although other values and edges may still be “hidden” in out-of-focus columns. BioXSD on the other hand allows combining features, types of track positions, values, and interconnections freely without any operational focus. Thus, if a tool consuming BioXSD feature data demands it, a particular operational focus of the data must be supplied by the user.

Compared to other generic sequence-feature formats, BioXSD allows defining complex, structured meanings of annotations, as well as complex feature data and metadata, or relations. This would not be conveniently possible in a tabular format and takes advantage of the XML. BioXSD types can freely be combined and included within documents, files, or applications’ inputs and outputs. They can easily be combined with other XML formats defined in other XSDs, can be extended just like classes in an object-oriented programming language, or further restricted using built-in XSD mechanisms. BioXSD can be validated and parsed by ordinary XML/XSD-handling frameworks.

Table 5 The allowed content of a BioXSD FeatureRecord

<table>
<thead>
<tr>
<th>BioXSD description of feature type</th>
<th>May further contain</th>
</tr>
</thead>
<tbody>
<tr>
<td>Notes</td>
<td></td>
</tr>
<tr>
<td>BioXSD description of feature type</td>
<td></td>
</tr>
<tr>
<td>Name</td>
<td>1</td>
</tr>
<tr>
<td>Ontology concepts</td>
<td>1</td>
</tr>
<tr>
<td>Synonyms</td>
<td></td>
</tr>
<tr>
<td>Textual note</td>
<td></td>
</tr>
<tr>
<td>References</td>
<td>to database entries, databases, ontology concepts, other feature types</td>
</tr>
<tr>
<td>More specific type of feature</td>
<td>type of relationship with the referenced object</td>
</tr>
<tr>
<td>More generic class of feature types</td>
<td>name and/or concepts, synonyms, database entries</td>
</tr>
<tr>
<td>BioXSD feature occurrence</td>
<td></td>
</tr>
<tr>
<td>Position</td>
<td>segments, points, positions outside of the actual sequence or feature occurrence, dense points (function) and dense partition or step function</td>
</tr>
<tr>
<td>Scores (values)</td>
<td>double-precision signed floats (8 bytes), or any well-formatted strings</td>
</tr>
<tr>
<td>Evidence</td>
<td>unit, index, type of score, note, position, provenance metadata</td>
</tr>
<tr>
<td>Name</td>
<td>references to databases, tools, and citations; scores, verdict, reliability, provenance metadata</td>
</tr>
<tr>
<td>Note</td>
<td></td>
</tr>
<tr>
<td>Alignments</td>
<td></td>
</tr>
<tr>
<td>Sequence variation</td>
<td></td>
</tr>
<tr>
<td>Frame</td>
<td></td>
</tr>
<tr>
<td>CDS phase</td>
<td></td>
</tr>
<tr>
<td>References</td>
<td>to ontology concepts, database entries, other feature occurrences (interconnections)</td>
</tr>
<tr>
<td></td>
<td>type of relationship with the referenced object, scores of the relationship (weights of edges)</td>
</tr>
</tbody>
</table>

1 At least one of these two is mandatory.
2 By any ontology concept, referred to by a concept URI, identifier, or term; or by a custom term if no ontology concept is available.
3 Points are bases/residues or insertions between them.
4 For example if annotating the position of a regulatory element of a coding sequence, or relations between genes or protein domains.
5 Positions can form multi-segment subsequences, multi-point tuples, and can be combined within feature occurrences according to users’ needs. The positions are always 1-based. The feature occurrence may apply to the whole sequence (being a non-positioned sequence property).

* Added in BioXSD version 1.1.
It has, however, been problematic to use XML formats for highly voluminous data such as whole-genome annotations. The textual serialization of XML is more verbose compared to a textual tabular format, and even more compared to a bespoke binary format. Many basic XML-handling tools have high runtime demands for computer memory, making parsing of huge XML documents impossible. All these problems are hopefully going to be solved thanks to the recent and long-expected Efficient XML Interchange (EXI) standard by the World Wide Web Consortium [17], together with its growing family of supporting libraries, and tools for streamed XSLT transformations and random-access XPath and XQuery queries. EXI defines the way any XML data or document should be serialized in a standard binary format that will be many times smaller and faster to access than the textual XML. There is no need to develop one’s own bespoke binary encodings and parsers when using EXI, and the data can be programmatically handled transparently, with the same look and feel as the ordinary XML.

Availability of specifications and supporting tools
The BioXSD 1.1 XML Schema is available at [26]. BioXSD data can be validated by all the main XML validation tools, and consumed and produced programmatically by the bulk of the common XML/XSD-handling libraries. Further information and documentation are available at [27].

A complete specification of the GTrack format version 1.0 is attached as Additional file 1 and is also available from the GTrack website [20]. The website also contains supporting tools for the GTrack format, connected to the Genomic HyperBrowser [10,28]. Table 6 contains an overview of all GTrack-related tools available as webtools.

The GTrack format is maintained by Sveinung Gundersen and the BioXSD format is maintained by Matúš Kalaš. Both formats are licensed under the Creative Commons Attribution-NoDerivs 3.0 Unported License [29].

The Genomic HyperBrowser [10,28] is built on top of the Galaxy framework [30,31] and provides a large set of statistical investigations tailored for the specific track types of supplied tracks. In order for such analyses to be efficient, the system uses a binary storage scheme internally. In this scheme, the core informational columns are stored as C vectors directly written to disk. The vector files are then accessed using the NumPy package [32] for Python [33], allowing very efficient vector computations. A linear index of the files is built in order to allow random access to the data. This binary representation is stored in parallel to the files in their original format, and updated automatically as the original files are updated. The implementation is open source and available as part of the HyperBrowser code base under the GPL license, version 3 [34]. As an alternative, the recently published Tabix tool [35] provides fast access to tabular data in compressed form, and works with GTrack files of types Points and Segments, and their derivatives.

Conclusions
By systematic analysis of informational properties of genomic tracks, we delineated fifteen distinct types of tracks. These track types shed light on the variability of track representations, suggesting that the differences between formats is not only due to preferences and conventions, but also to fundamental differences in the information inherent in different tracks. Furthermore, discerning the informational properties of a track allows the nature of the track to be precisely conveyed, as well as clarifying what represents meaningful analyses on a given track.

The identification of core informational properties of tracks, as well as a broad survey of various practicalities concerning existing formats, created a basis for the specification of a new format for genomic data: the GTrack format. By allowing precise interpretation, simple parsing, as well as relatively straightforward conversion to several existing formats, we believe that the introduction of this "yet another format" will actually help streamline

---

Table 6 Overview of the webtools available from the GTrack website [20]

<table>
<thead>
<tr>
<th>GTrack supporting tools</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Show GTrack specification</td>
<td>Displays a HTML version of the GTrack specification</td>
</tr>
<tr>
<td>Validate GTrack file</td>
<td>Checks whether a GTrack file complies with the specification</td>
</tr>
<tr>
<td>Convert tabular file to GTrack</td>
<td>Converts any tabular file to GTrack</td>
</tr>
<tr>
<td>Convert file to/from GTrack</td>
<td>Converts to and from common tabular formats (GFF, BED, WIG, bedGraph)</td>
</tr>
<tr>
<td>Expand GTrack headers</td>
<td>Expands partially completed GTrack headers based on data contents</td>
</tr>
<tr>
<td>Standardize GTrack file</td>
<td>Converts a GTrack file to track type “linked valued segments” using the default indexing scheme</td>
</tr>
<tr>
<td>Sort GTrack file</td>
<td>Sorts a GTrack file (including bounding regions)</td>
</tr>
<tr>
<td>Complement GTrack columns</td>
<td>Complements the columns of a GTrack file based on another GTrack file</td>
</tr>
</tbody>
</table>

All tools are implemented as part of the Genomic HyperBrowser [10,28] and available under the GPL license, version 3 [34].
data representation in the field. Finally, by coordinating the GTrack format with an enhanced and optimized version 1.1 of the BioXSD format, this also aids in unifying tabular and XML-based track representation, while keeping the specific advantages of the two.

Additional material

Additional file 1: GTrack specification. Specification document of GTrack 1.0.

Abbreviations

BAM: Binary Alignment/Map format; BED: Browser Extensible Data format; ChiP-seq: Chromatin Immunoprecipitation sequencing; EXI: Efficient XML Interchange; F: function; GFF: General Feature Format; GTF: Gene Transfer Format; GVF: Genome Variation Format; GP: genome partition; P: points; LBP: linked base pairs; LF: linked function; LGP: linked genome partition; LP: linked points; LS: linked segments; LSF: linked step function; LVP: linked valued points; LVS: linked valued segments; S: segments; SAM: Sequence Alignment/Map format; SF: step function; SNP: single nucleotide polymorphisms; URL: Uniform resource identifier; URL: Uniform resource locator; VP: valued points; VS: valued segments; WIG: Wiggle format; WSDL: Web Service Definition Language; W5/Si/ WG: what you see is what you get; XML: Extensible Markup Language; XSD: XLM Schema Definition.

Acknowledgements and funding

Funding was kindly provided by EMBIO, FUGE, UiO, Helse Sør-Øst, and Statistics for Innovation, a Centre for Research-Based Innovation funded by the Research Council of Norway. We thank Kai Trengereid for crucial work in developing the GTrack-related tools, and Inge Jonassen for valuable input on the BioXSD format. We would also like to acknowledge the excellent review work provided by the peer reviewers. These reviews have contributed significantly to the content of this paper.

Author details

1Department of Tumor Biology, The Norwegian Radium Hospital, Oslo University Hospital, Montebello, 0310 Oslo, Norway. 2Computational Biology Unit, Univ Computing, Thormoåhsengate 55, 5008 Bergen, Norway. 3Department of Informatics, University of Bergen, Thormoåhsengate 55, 5008 Bergen, Norway. 4Department of Computer Science, University of Bergen, Thormoåhsengate 55, 5008 Bergen, Norway. 5Department of Medical Informatics, The Norwegian Radium Hospital, Oslo University Hospital, Montebello, 0310 Oslo, Norway. 6Department of Informatics, University of Oslo, Blindern, 0316 Oslo, Norway.

Authors’ contributions

SG, AF, EH and GKS conceived and developed the ideas on track type distinctions. SG, MK, OA and GKS developed the GTrack specification. SG and GKS wrote the main parts of the paper. MK wrote the parts on XML-based track representation and developed BioXSD 1.1. SG and GKS were involved with the development of GTrack-related tools. All authors read and approved the final manuscript.

Received: 11 May 2011 Accepted: 30 December 2011 Published: 30 December 2011

References

4. UCSC genome browser data formats. [http://genome.ucsc.edu/FAQ/FAQformat.html].
13. Web services provided by the Center for Biological Sequence analysis (CBS), Technical University of Denmark. [http://www.cbs.dtu.dk/wa/].
20. GTrack. [http://www.gtrack.no/].
28. The genomic hyperbrowser. [http://hyperbrowser.uio.no/].
29. Creative commons Attribution-NoDerivs 3.0 Unported license (CC BY-ND 3.0). [http://creativecommons.org/licenses/by-nd/3.0/].
34. GNU General Public License, version 3. [http://www.gnu.org/copyleft/gpl.html].

Page 16 of 17
http://www.biomedcentral.com/1471-2105/12/494
Gundersen et al. BMC Bioinformatics 2011, 12:494
Submit your next manuscript to BioMed Central and take full advantage of:

- Convenient online submission
- Thorough peer review
- No space constraints or color figure charges
- Immediate publication on acceptance
- Inclusion in PubMed, CAS, Scopus and Google Scholar
- Research which is freely available for redistribution

Submit your manuscript at www.biomedcentral.com/submit
The differential disease regulome

Geir K Sandve1, Sveinung Gundersen2†, Halfdan Rydbeck1,3,4, Ingrid K Glad5, Lars Holden3, Marit Holden3, Knut Liestøl1,4, Trevor Clancy2, Finn Drabløs6, Egil Ferkingstad3, Morten Johansen7, Vegard Nygaard8, Eivind Tøstesen7, Arnoldo Frigessi3,8 and Eivind Hovig1,2,3,7*

Abstract

Background: Transcription factors in disease-relevant pathways represent potential drug targets, by impacting a distinct set of pathways that may be modulated through gene regulation. The influence of transcription factors is typically studied on a per disease basis, and no current resources provide a global overview of the relations between transcription factors and disease. Furthermore, existing pipelines for related large-scale analysis are tailored for particular sources of input data, and there is a need for generic methodology for integrating complementary sources of genomic information.

Results: We here present a large-scale analysis of multiple diseases versus multiple transcription factors, with a global map of over-and under-representation of 446 transcription factors in 1010 diseases. This map, referred to as the differential disease regulome, provides a first global statistical overview of the complex interrelationships between diseases, genes and controlling elements. The map is visualized using the Google map engine, due to its very large size, and provides a range of detailed information in a dynamic presentation format. The analysis is achieved through a novel methodology that performs a pairwise, genome-wide comparison on the cartesian product of two distinct sets of annotation tracks, e.g. all combinations of one disease and one TF. The methodology was also used to extend with maps using alternative data sets related to transcription and disease, as well as data sets related to Gene Ontology classification and histone modifications. We provide a web-based interface that allows users to generate other custom maps, which could be based on precisely specified subsets of transcription factors and diseases, or, in general, on any categorical genome annotation tracks as they are improved or become available.

Conclusion: We have created a first resource that provides a global overview of the complex relations between transcription factors and disease. As the accuracy of the disease regulome depends mainly on the quality of the input data, forthcoming ChIP-seq based binding data for many TFs will provide improved maps. We further believe our approach to genome analysis could allow an advance from the current typical situation of one-time integrative efforts to reproducible and upgradable integrative analysis. The differential disease regulome and its associated methodology is available at http://hyperbrowser.uio.no.

Background

Knowledge of the molecular biology of the cell is rapidly being gained, providing increasing detail of the cellular signalling systems, as well as better mapping of the various parts of cell regulation. Among the elements that provide dynamics to a signalling system are the transcription factors that bind to sequence specific transcription factor binding sites (TFBSs) along the DNA to regulate gene transcription. Transcription factors represent a potential as drug targets, as ablation of activity of a certain transcription factor may impact a distinct set of genes under its control. One option is therefore to target a transcription factor of a disease-relevant pathway.

However, the challenges associated with the development of drugs for transcription factors have to some extent limited their use, partly due to the structural requirements of inhibition. A recent example of a successful strategy involves inhibition of NOTCH1 in leukemia [1], hinting towards a more rapid development of opportunities for transcription factor inhibition. Other examples
targeting transcription factors using small molecule drugs include Stat3 [2] and NFKappaB [3].

The development of a global map of transcription factor over- and under-representation in disease could reveal information relevant for drug target prioritization, as well as serving as a novel knowledge resource.

The relation between a single transcription factor (TF) and a single disease can be probed by evaluating the frequency of binding sites for the TF in regulatory regions of genes assumed to have a role in the disease. One useful strategy in this direction has been to identify differentially expressed genes in a disease state, followed by motif discovery [4]. Binding motif profiles are available for a large number of TFs in motif libraries like Transfac [5] and JASPAR [6], facilitating investigations of multiple TFs. With the advent of technology such as ChIP-chip [7] and ChIP-seq [8], it is now becoming possible to map the binding sites for each TF in unprecedented detail, although such experimental data is still sparse. Therefore, genome-wide predictions of binding sites, albeit noisy, remain valuable sources, and predictions for a large number of TFs are available [9,10], as well as predictions of the target genes for a large number of TFs [11,12] A number of in-depth studies have addressed the functional characterization of TF binding motifs [13-15]. Also, several sources provide information regarding gene-associations for a range of diseases. Phenopedia [16] is a recently developed disease-centered view of the manually curated Human Genome Epidemiology Network (HuGENet) database of genetic associations [17], covering all multifactorial diseases. Another approach is represented by the IntOGen tool, which facilitates integration of data sources relevant for cancer development [18]. Combining such resources with TF binding predictions now permit the development of a global visualization of statistical overrepresentation of regulatory elements across all diseases.

**Results**

**Pairwise analysis of cartesian products**

In order to advance from the current typical situation of one-time integrative efforts, we have created a generic methodology for integrating complementary sources of genomic information. This is based on an abstract representation of genomic information in the form of genome annotation tracks, allowing very different information types to be treated in a similar manner. Each input source is a set of related genome annotation tracks, e.g. a set of disease tracks or a set of TF tracks. The methodology performs a pairwise, genome-wide comparison on the cartesian product of two distinct sets of annotation tracks, e.g. all combinations of one disease and one TF. The results are provided in the form of tables and interactive heatmaps with the underlying data easily available.

The pairwise comparison of annotations is based on a principled mathematical approach to genomic analysis, where the test statistic in principle can be selected from a range of relations between annotation tracks generically represented as tracks of points, segments or functions. Based on the selected test-statistic, normalized values of over-/under-representation are computed and visualized (see Figure 1 for a schematic representation of the strategy and Methods for details).

**Differential disease regulome**

In order to obtain a global view of diseases and transcription factors, we have used our integrative methodology to perform a large-scale analysis on the combination of the Phenopedia disease-gene catalogue [16] and a recent resource of predicted target genes of TFs [12] (where the datasets of 1010 diseases and 446 TF motifs are constructed as detailed in Methods). The resulting map is referred to as the differential disease transcriptional regulome. For each combination of disease and TF (i.e. the Cartesian product), we find the intersection of TF target genes and disease-associated genes across the genome. Under our main scheme, we investigate deviations from the average across the set of all selected diseases and therefore refer to the regulome as differential. We have developed two different views of the resulting data: a dynamic list of TFs for each disease, and a clustered heat map representation. While the dynamic lists provide direct access to z-score values of over-/under-representation, the heat map representation allows a broader overview of results, providing powerful visual clues of the most deviant associations, and also providing a broad impression of similarities and differences between specific TFs/diseases of interest. As both rows and columns are clustered, diseases with similar profiles of association to TFs will be adjacent (and similarly for TFs against diseases), allowing larger patterns of associations between sets of diseases and TFs to be spotted visually (as well as specific deviances within such clusters). The clustered heat map was visualized using the Google Maps engine, due to the very large number of elements (see Figure 2).

The information to be gained from this approach will obviously depend on the signal-to-noise ratio in the underlying data. To demonstrate the presence of useful signals in the disease regulome, we analyzed a prominent set of TF motifs that were overrepresented in a set of 116 immune related conditions, including immunologic deficiency syndromes, graft versus host disease, asthma, allergies, a number of autoimmune and infectious diseases. The cluster was defined by six NF-κB/Rel-related TF motifs. A smaller cluster based on a subset of 81 of the immune related conditions was also identified, based on six TF motifs, four related to the IRF family, one to Stat1 and to one Cux1 (See Supplemental Results in Additional
file 1). There is currently intense ongoing activity in mapping of the regulation of immune diseases, and both experimental data and computational methods are applied. In essence, all of the TFs identified here have previously been implicated in important immune related settings [19-26]. For each cluster, the underlying genes were ranked according to the number of TF bindings found. The highest scoring gene was TNF, regulated by

![Diagram of regulome construction]

**Figure 1** Schematic model of regulome construction. Two input sources are selected, e.g. a set of TF tracks and a set of disease tracks. For each combination, the pairwise relation model, in this case the number or genes containing TF binding locations, is evaluated and subsequently differentiated against the full matrix of counts. The main output is an interactive heat map of over-/under-representation that for each entry also includes detailed information and links to follow-up analysis. The regulome construction is performed by a web-based system, the Genomic HyperBrowser [33], that allows input data, a pairwise relation model and a measure of deviation to be selected.

![Screenshot of differential disease regulome]

**Figure 2** A screenshot of the differential disease regulome, using Google Maps API http://code.google.com/apis/maps/index.html for visualization and user interface. Detailed information about each disease-TF combination is available. The selected cell contains information about the overrepresentation of HIF-1alpha in the regulation of the genes associated to Barret Esophagus, a relation previously reported [35].
the NF-κB/Rel TFs, according to the clustering. When ranking diseases according to effect size, i.e. the over-/under-representation of TF binding as compared with the expected value for that disease, a set of autoimmune diseases, including arteritis and spondylitis scored highest.

In addition to the immune-related example, the disease regulome contains a vast number of small and large clusters not described further here, that may represent interesting hypotheses for further investigation (see Figure 3 and Additional file 2 and 3 for listings). One example is that of the ETS1 transcription factor (V$ETS1_B$), indicated to be statistically significantly over-represented in glioblastoma and astrocytoma (indexes: $(48, 141$ and $142)$). Using a decoy strategy to functionally deactivate ETS1, Sahin et al. could demonstrate reduced tumorigenesis of rat C6 glioma cells in an in vivo model, underlining the concept value [27].

An additional example investigation of a small cluster is provided in Additional file 1.

**A flexible approach to integrative genomics**

There are several reasons why the process of generating a resource like the disease regulome should be as highly automated and as flexible in scope as possible.

First, there is an obvious need to update maps such as the disease regulome, as the underlying data quality constantly improves. Second, there are also presently several alternative sources available both for disease associations and regulation, each having different characteristics, and thus having the potential to provide complementary information. We have generated various alternate versions of the disease regulome based on combinations of different sources of regulation data. For diseases, we have used both literature-mined [28] and experimentally-based cancer disease associations [18], and for TFs, we have used predicted binding sites from UCSC (see Methods). Third, the disease regulome represents only one instance of a whole class of similar maps that may be generated. We have compiled a large collection of resources similar to the disease regulome, addressing other regulatory aspects of genes, including microRNAs, histone modifications and repeat structures in DNA instead of TFs, and with gene regions associated to Gene Ontology terms or even simply chromosome arms or cytobands instead of disease gene regions (Additional file 1). A total of 17 different regulomes can be browsed interactively on our web server http://hyperbrowser.uio.no.

**Analyzing the immune component in alternative regulomes**

To further characterize the immune component that we observed in the disease regulome, we examined a map of all gene ontology terms versus TFs for potential immune related clusters, and could indeed identify two clusters containing 83 and 79 immune related GO terms, respectively. This cluster was intriguingly defined essentially by the same TF motifs as for the disease list, indicating that this is a strong signal of functional importance (Additional file 1). Based of the top 100 ranked genes from each cluster, we identified the genes with most influence on the differences and similarities across the clusters, thus most central for regulation. Of these 275 unique genes, 14 were found in all the clusters, while an additional 85 genes were present in more than one cluster (see Figure 4 and Table 1). As it is known that there is a level of transcriptional regulation by histone modifications [29], we further analyzed a heatmap of co-localization of TF binding sites and various histone modifications in T-cells, to examine whether we could identify a cluster of TFs similar to those we had identified with the disease and GO regulomes. We identified a cluster that contained 7 of the 9 TF motifs previously found that were part of the dataset used (Additional file 1), with the histone modifications H3K4me3, H3K36me3 and H2AZ being enriched. Interestingly, the H3K4me3 pattern has been identified as important for the binding of the transcription factor and autoimmune inhibitor Aire, which is not present in the TF data set used. It has previously been speculated that IRF family members may form a higher order transcriptional complex with Aire [30]. The H3K36me3 modification has been linked to a number of autoimmune diseases through SNP associations in GWAS studies [31].

**Discussion**

We have here introduced a generic methodology for large-scale integration of genomic information. Based on this methodology we have generated a collection of novel genomic resources in the form of interactive maps that show the relation between various genomic elements. This collection of genomic resources includes the disease regulome, which shows the relation between TFs and diseases in the form of ≈450000 values of over-/under-representation for specific combinations of a TF and a disease.

A common approach to automating large-scale analysis is the construction of a dedicated pipeline for the purpose. Examples of this include a pipeline of for integrating binding site scanning of TF motifs with sets of gene promoters [14], and the GREAT tool for finding enriched annotations in an input set of genomic regions [15]. Although such pipelines may allow e.g. thousands of TF vs gene set combinations to be explored efficiently, the construction of the pipeline itself is a labour-intensive task. The resulting pipeline may be quite ad hoc and limited to certain investigations. Our methodology is based on a generic representation of genomic information in the form of annotation tracks, making it possible to exchange both
input sources for a pairwise comparison. The methodology presented here can thus be used in a much wider range of settings than what can be achieved with a typical pipeline. We are able to treat the relation between for example chromosomes and histone modifications in the same manner as the TF-disease relation. This allows us to generate the large number of maps presented in the article, and also allows our methodology to be easily applied to future investigations.

In light of the many possible variations of input data and parameterizations, the question of the robustness of the methodology logically arises. Consider again the immune-related clusters discussed previously. Looking at the hit rate of a gene (the percentage of disease-TF
pairs in the clusters where the gene is relevant to disease regulation) we find that only 24 genes have a hit rate exceeding 25%. Still, the combined hit rate of these 24 genes only comprise about a third of the total hit rate, leaving the rest of the contributions scattered over many genes. These clusters therefore seem to be quite robust as regards to noise of single genes. Other clusters may, however, be chiefly caused by one or a small number of genes. Inspecting the gene lists is therefore important when assessing the robustness of a finding.

The results of the clustering algorithm seem to be quite dependent on variations in parameters and input data. When parameters are varied, large and distinct clusters will usually stay robust, but smaller, less well-defined clusters will typically move around, or be split and merged together. Thus, there is no “final” version of the disease regulome. We provide a set of disease regulomes with varying data sources and under two specific null hypotheses (Additional file 1), in addition to a range of other maps based on other combination of data sources.

Note that clustering of diseases in the disease regulome heat map should not be thought of in terms of phenotypic similarities, and neither in terms of general similarity at a molecular level. The clustering is exclusively focused on transcriptional regulation of genes connected to the diseases, where high similarity between two diseases

<p>| Table 1 Lists of unique genes contributing to the immune clusters |
|----------------------|----------------------|----------------------|----------------------|----------------------|----------------------|----------------------|</p>
<table>
<thead>
<tr>
<th>Gene</th>
<th>Hit rate</th>
<th>Gene</th>
<th>Hit rate</th>
<th>Gene</th>
<th>Hit rate</th>
<th>Gene</th>
<th>Hit rate</th>
<th>Gene</th>
<th>Hit rate</th>
<th>Gene</th>
<th>Hit rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>HLA-A</td>
<td>22.1%</td>
<td>TNF</td>
<td>99.5%</td>
<td>TLR4</td>
<td>12.1%</td>
<td>HLA-B</td>
<td>30.1%</td>
<td>CSF1</td>
<td>19.3%</td>
<td>ICAM1</td>
<td>74.7%</td>
</tr>
<tr>
<td>NOD2</td>
<td>20.1%</td>
<td>LTA</td>
<td>58.3%</td>
<td>IL6</td>
<td>9.5%</td>
<td>CARD15</td>
<td>22.3%</td>
<td>TNFSF13</td>
<td>13.3%</td>
<td>IRF6</td>
<td>44.0%</td>
</tr>
<tr>
<td>STAT1</td>
<td>12.2%</td>
<td>TGFBI</td>
<td>36.9%</td>
<td>CTLA4</td>
<td>9.4%</td>
<td>VDR</td>
<td>17.5%</td>
<td>FLT3LG</td>
<td>8.0%</td>
<td>CD69</td>
<td>28.9%</td>
</tr>
<tr>
<td>CD40</td>
<td>11.6%</td>
<td>CD4</td>
<td>24.8%</td>
<td>CCR5</td>
<td>8.5%</td>
<td>VEGF</td>
<td>4.1%</td>
<td>RELB</td>
<td>6.9%</td>
<td>REL</td>
<td>27.7%</td>
</tr>
<tr>
<td>TAP1</td>
<td>10.7%</td>
<td>AKT1</td>
<td>12.1%</td>
<td>TLR3</td>
<td>7.7%</td>
<td>NFKBIZ</td>
<td>3.3%</td>
<td>CD27</td>
<td>6.0%</td>
<td>TNFSF4</td>
<td>21.1%</td>
</tr>
<tr>
<td>NFKBA</td>
<td>10.7%</td>
<td>LT8</td>
<td>11.6%</td>
<td>HLA-DRB1</td>
<td>7.2%</td>
<td>COL6A1</td>
<td>2.4%</td>
<td>PER1</td>
<td>5.9%</td>
<td>RELA</td>
<td>18.7%</td>
</tr>
<tr>
<td>IRF1</td>
<td>10.1%</td>
<td>IL1RN</td>
<td>10.2%</td>
<td>IL-2</td>
<td>5.9%</td>
<td>PSORS1</td>
<td>2.1%</td>
<td>PTPN6</td>
<td>5.6%</td>
<td>EDC4</td>
<td>16.9%</td>
</tr>
<tr>
<td>CXCL10</td>
<td>10.0%</td>
<td>IL2RA</td>
<td>10.2%</td>
<td>IFNBI</td>
<td>4.2%</td>
<td>CYBA</td>
<td>1.7%</td>
<td>LCK</td>
<td>5.3%</td>
<td>CD58</td>
<td>14.5%</td>
</tr>
<tr>
<td>IRF5</td>
<td>7.3%</td>
<td>TNFRSF18</td>
<td>5.3%</td>
<td>CCL2</td>
<td>4.1%</td>
<td>DDAH2</td>
<td>1.5%</td>
<td>MYD88</td>
<td>5.0%</td>
<td>TNFRSF18</td>
<td>14.1%</td>
</tr>
<tr>
<td>PSMB8</td>
<td>6.7%</td>
<td>CD86</td>
<td>5.3%</td>
<td>TNFRSF18</td>
<td>3.5%</td>
<td>CYP27B1</td>
<td>1.4%</td>
<td>PTMA</td>
<td>4.8%</td>
<td>CD83</td>
<td>14.1%</td>
</tr>
<tr>
<td>PSMB9</td>
<td>6.2%</td>
<td>ITGAM</td>
<td>5.1%</td>
<td>MX1</td>
<td>3.3%</td>
<td>PLAU</td>
<td>1.4%</td>
<td>IL27</td>
<td>4.7%</td>
<td>TNFRSF9</td>
<td>12.9%</td>
</tr>
<tr>
<td>FAS</td>
<td>5.5%</td>
<td>TRADD</td>
<td>5.0%</td>
<td>STAT5A</td>
<td>3.0%</td>
<td>RUNX1</td>
<td>1.3%</td>
<td>B2M</td>
<td>4.3%</td>
<td>IL17C</td>
<td>12.0%</td>
</tr>
<tr>
<td>IL7R</td>
<td>2.6%</td>
<td>MIF</td>
<td>3.9%</td>
<td>NOD1</td>
<td>3.0%</td>
<td>RUNX3</td>
<td>1.2%</td>
<td>IRF2</td>
<td>3.1%</td>
<td>CD5</td>
<td>10.8%</td>
</tr>
<tr>
<td>IFH1</td>
<td>1.9%</td>
<td>NFKBIB</td>
<td>3.4%</td>
<td>TLR1</td>
<td>2.9%</td>
<td>PAX2</td>
<td>1.1%</td>
<td>STAT3</td>
<td>3.0%</td>
<td>CD70</td>
<td>9.6%</td>
</tr>
<tr>
<td>CXCL5</td>
<td>3.2%</td>
<td>HLA-DMA</td>
<td>2.3%</td>
<td>CCND1</td>
<td>1.0%</td>
<td>TAPBP</td>
<td>2.9%</td>
<td>DPP4</td>
<td>9.6%</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PTGS1</td>
<td>3.0%</td>
<td>CASP1</td>
<td>2.1%</td>
<td>RXR8</td>
<td>1.0%</td>
<td>BIRC3</td>
<td>2.5%</td>
<td>NAFATC2</td>
<td>9.0%</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SOCS1</td>
<td>1.7%</td>
<td>HIF1A</td>
<td>0.8%</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TRIM21</td>
<td>1.3%</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CCL21</td>
<td>1.1%</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Lists of genes that are unique for selected combinations of the four immune-related clusters analyzed (see Figure 4). For each gene, the hit rate (proportion of cluster where the gene is relevant to disease regulation) is presented. The NF-κB/Rel cluster in the Gene Ontology regulome is included because of the large hit rates (only the top of the list is shown). Note that TNF is relevant for regulation for nearly all disease-TF pairs in both NF-κB/Rel-related clusters. The full gene listing is included in Additional file 4.
could mean that these diseases share several associated genes, or they could be associated to different genes that are still targeted by many of the same TFs.

As genome-wide dataset typically contain a substantial amount of noise, a main consideration when generating regulomes is to provide sufficient signal-to-noise ratio for the global maps to be meaningful. Predicted TF binding sites are currently of very limited accuracy. The predicted TF-gene bindings used in the main disease regulome are more accurate [12], but could still be largely improved by either more precise prediction or by substituting predictions with forthcoming ChIP-seq based results for many of the TFs. For instance, a very interesting regulome would be that of combining TF binding sites actually used in T cells and T cell histone modifications, as opposed to the presently applied less precise prediction scheme. The Phenopedia disease-association database [16] is constantly growing, and could also be complemented by experimentally-based evaluations of disease associations. The TF-relevance for a given disease could use more sophisticated strategies than the generic model of track intersection used here, as could other model assumptions for the expected values be improved from the two used here.

A large-scale, automated effort like the disease regulome clearly represents an inferior handling of a specific TF-disease relation compared to what can be achieved by a separate, manual investigation of the same relation. We don’t claim that each value in the disease regulome represents a satisfactory conclusion regarding the relation between a specific TF and disease. What we claim is merely that our approach is able to capture a part of the underlying reality, and that multiplied with the large number of combinations studied, a map like the disease regulome constitutes a substantial resource of genomic information. We consider the disease regulome as mainly a hypothesis-generating tool to be used as the starting point for a number of future investigations.

Conclusion
We believe the disease regulome may prove immensely useful in early phases of research projects, as a resource for obtaining an initial overview of the regulation of disease and for supporting the formation of hypotheses to be studied further by computational or experimental methods. Moreover, we believe that the disease regulome will provide important pathway information for diseases, thereby serving as an aid to target identification and drug development.

Methods
Data set of transcription factor binding sites
For the main disease regulome, we used binding predictions for a set of 446 transcription factor motifs (PWMs), each with a list of the top 1000 predicted target genes [12]. These predictions are based on machine learning from 29 relevant features, including conservation, CpG island content, DNase I hypersensitivity and histone modifications, in addition to the PWM score. Other regulomes were calculated on the basis of a track of transcription factor binding site (TFBS) predictions for 258 PWMs from the UCSC genome browser called "TFBS conserved". The track was generated by Matt Weirauch and Brian Raney at UCSC and last updated July 17, 2007 [http://genome.ucsc.edu/cgi-bin/hgTrackUi?hgsid=153908909&c=chrX&g=tfbs-ConsSites]. Each PWM represent the binding specificity of one or a small set of closely related transcription factors. For simplicity, we mostly refer to the regulatory categories as TFs, instead of TF motifs, or PWM.

Data set of disease gene lists
Disease-associated genes were mapped to genomic coordinates, resulting in a set of genome regions for each disease. For the main regulome we used the complete list of all disease-gene association from Phenopedia [16], which is based on years of manually curation of reported associations in literature, collected the Human Genome Epidemiology Network (HuGENet) database. Only diseases with more than 20 gene associations were included.

An alternative data source for disease-associated genes sets was also used, based on citations and co-citations of disease and gene terms in the literature, as collected by PubGene [28]. Let N be the number of documents in this collection and let m be the number of documents that mention disease term d and n the number of documents that mention gene term g. Under the null model that there is no association between the disease term d and gene term g, the number of documents that mention both d and g follows a hypergeometric distribution with parameters N, m and n. We then define the gene list for a given disease to be all genes for which we obtain a Bonferroni-corrected p-value less than 0.02. Only diseases with more than 300 citations in literature were included. Both sources of gene lists use the set of diseases as defined by MeSH (Medical Subject Headings).

Intersecting transcription factors and diseases
For each combination of TF and disease, we counted, over the whole genome, the number of segments (disease genes) with at least one point (TF binding prediction) falling inside them. For the main TF data set, the points refer to the TSS of the genes with predicted associations. As this dataset used gene coordinates according to RefSeq, while the disease datasets were encoded using Ensembl, we extended the gene regions by 150 bp upstream to provide support for TSS inconsistencies between the two standards. To reduce noise, only diseases with a gene list
of at least 20 genes were included. For the UCSC data set, each point refers to a predicted TFBS. As TFBSs acting on a gene are often close to but outside a gene, we extended the gene regions to include flanks, set to 5 kb in each direction. In this case we only included diseases with a gene list of at least 10 genes. The choice of 5 kb in each direction is somewhat arbitrary, simply assuming that a substantial amount of relevant binding sites would be included. Other schemes are easily incorporated in the approach.

To conclude whether a TF is associated to a disease, we used hypothesis testing to investigate whether the number of disease-associated gene regions containing locations of binding (TSS or TFBS) of the given TF, were greater or less than expected by chance. Two different tests, providing complementary information, have been implemented. In both tests we assume under the null hypothesis that gene regions are fixed, that the number of binding locations for each TF is fixed, and that their positions are randomly selected among the positions containing binding locations for any TF.

We have calculated z-scores based on the deviation from expected values under two specific null hypotheses. Several different model assumptions may be reasonable. In our main scheme, the null-hypothesis is that the proportion of binding sites associated to a given TF is the same within the regions of a given disease as it is across all diseases.

The first test further assumes in the null hypothesis that binding locations of a given TF falls uniformly among the set of positions containing binding locations for any TF.

Conversely, the second test modifies this by assuming in the null hypothesis that binding locations of the given TF falls inside gene regions of the given disease proportionally to how often the binding locations of this TF on average falls inside gene region sets across all diseases. The first test is based on the hypergeometric distribution, while the second test is based on the binomial distribution. Details and formulas for both tests are provided in Additional file 1.

### Clustering

Groups of similar disease/TF tracks are found by separately clustering rows and columns of a matrix of z-values (see above). Hierarchical clustering has been used, as this provides information on several levels, both closely related diseases and large groups of diseases with a certain amount of similarity. As similarity measure between individual objects (diseases/TFs) we used the Euclidean distance, and for distance between clusters we used the average (Euclidean) distance between all pairs of objects. Further details on the clustering are given in Additional file 1.

### Data sets of complementary regulomes

A range of regulomes have been generated based on different combinations of input data. These regulomes make use of gene lists associated with GO terms, as well as histone modification data. The gene lists for GO terms are generated based on literature co-occurrence in the same way as the gene lists for diseases. The histone modification dataset is based on raw tag hit data from ChIP-seq experiments on human T-Cells [29]. These were preprocessed using the NPS (Nucleosome Positioning from Sequencing) software [32], using peak detection, leading to nucleosome positioning information as short segments, treated as points.

When looking at the regulatory effects of histone modifications, we counted the number of points (defined as the middle of the DNA strand eclipsing modified nucleosomes) in the 2 kb up- and downstream region surrounding the transcription start site of each gene. A complete overview of complementary regulomes is given in Additional file 1.

### Software

The methodology is implemented within a software system that supports interactive, real-time, large-scale genomic analyses [33] (further details given in Additional file 1). The software system allows large and fully customizable analyses to be performed interactively. The system is open source, runs integrated with the Galaxy web server [34], and is available on the web at http://hyperbrowser.uio.no.

### Additional material

#### Additional file 1: Supplemental material for “The differential disease regulome”

Miscellaneous supplemental material: details on the Genomic HyperBrowser; overview of generated regulomes; details on immunology example; additional example from a disease regulome variant; statistics overview and supplemental figures.

#### Additional file 2: TF-disease clusters

A listing of 57 manually identified TF-disease clusters in the differential disease regulome.

#### Additional file 3: TF-GO clusters

A listing of 105 manually identified TF-GO clusters in the Gene Ontology regulome.

#### Additional file 4: Gene listings for immunology example

Listings of all the genes of the immunology regulome in the differential disease regulome and the Gene Ontology regulome, sorted on their hit rate. The genes indicated in the different sections of the Venn diagram in Figure 4 are also detailed here.

### List of abbreviations used

- bp: base pair
- GO: Gene Ontology
- kb: kilobases
- PWM: position weight matrix
- TF: transcription factor
- TFBS: transcription factor binding sites
- TSS: Transcription start site

### Acknowledgements and Funding

Peter Wiedswang is acknowledged for administrative assistance. We thank the Scientific Computing Group at USIT for providing friendly and helpful assistance on system administration. We also thank PubGene Inc. for kind assistance in the development of literature tracks. Additional funding was...
kindly provided by FUGE, EMBIO, UOIO and Helse Sør-Øst and the European Commission under the Seventh Framework Programme, grant agreement number 223367, MultiMod. This work was performed in association with Statistics for Innovation.

Author details
1. Department of Informatics, University of Oslo, Blindern, 0316 Oslo, Norway.
2. Department of Tumor Biology, The Norwegian Radium Hospital, Oslo University Hospital, Montebello, 0310 Oslo, Norway.
4. Centre for Cancer Biomedicine, The Norwegian Radium Hospital, Oslo University Hospital, Montebello, 0310 Oslo, Norway.
5. Department of Mathematics, University of Oslo, Blindern, 0316 Oslo, Norway.
6. Department of Cancer Research and Molecular Medicine, Norwegian University of Science and Technology (NTNU), 7491 Trondheim, Norway.
7. Institute for Medical Informatics, The Norwegian Radium Hospital, Oslo University Hospital, Montebello, 0310 Oslo, Norway.
8. Department of Biostatistics, Institute of Basic Medical Sciences, University of Oslo, Blindern, 0317 Oslo, Norway.

Authors’ contributions
GKS and SG contributed equally to this work. GKS, SG and EH conceived the regulome approach. GKS, AF and EH conceived the general HyperBrowser (HB) approach. HR inspired the clustered heatmap presentation. SG and MJ developed the regulome-specific functionality. GKS, SG and MJ implemented the regulome approach. FC, VN and ET participated in the general HB development. FD and EH contributed to disease regulome development and analysis. GKS and EH wrote the main parts of the manuscript. EH supervised the project. All authors read and approved the final manuscript.

Received: 28 January 2011 Accepted: 7 July 2011 Published: 7 July 2011

References


doi:10.1186/1471-2164-12-353
Cite this article as: Sandve et al.: The differential disease regulome. BMC Genomics 2011 12:353.
Appendix A

Analyses and tools in the Genomic HyperBrowser
Table A.1: Selected descriptive statistics and hypothesis tests available through the “Analyze genomic tracks” tool of the Genomic HyperBrowser. Each analysis is defined for either one or two tracks, with the corresponding track type denoted in the columns “Track 1 type” and “Track 2 type”. The track type abbreviations, as defined in Paper II, are as follows: Points (P), Segments (S), Valued Points (VP), Valued Segments (VS), Function (F), Linked Genome Partition (LGP), and any Linked (L) track. In addition, attached values are: number (default), case/control (c/c), category (cat), and weighted edges (w). Most hypothesis tests are available in one- and two-sided versions. Looking at e.g. overlap, the possible alternative hypotheses would then be whether the segments of track1 are overlapping the segments of track2, more, less, or differently than expected by chance. Results of the analyses are given both at the global level and for local regions along the genome. The table was fetched from manuscript to a recently published article not included with this thesis [211], edited to fit a two-page spread. The table was created by the author of this thesis, with contributions from the first author of the paper.

<table>
<thead>
<tr>
<th>Track 1 type</th>
<th>Track 2 type</th>
<th>Statistical investigation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>P</td>
<td>P</td>
<td>Different frequencies?</td>
<td>Where is the relative frequency of points of track 1 different from the relative frequency of points of track 2, more than expected by chance?</td>
</tr>
<tr>
<td>P</td>
<td>P</td>
<td>Located nearby?</td>
<td>Are the points of track 1 closer to the points of track 2 than expected by chance?</td>
</tr>
<tr>
<td>P</td>
<td>S</td>
<td>Located inside?</td>
<td>Are the points of track 1 falling inside the segments of track 2, more than expected by chance?</td>
</tr>
<tr>
<td>P</td>
<td>S</td>
<td>Located nonuniformly inside?</td>
<td>Do the points of track 1 tend to accumulate more towards the borders of the segments of track 2?</td>
</tr>
<tr>
<td>P</td>
<td>S</td>
<td>Located nearby?</td>
<td>Are the points of track 1 closer to the segments of track 2 than expected by chance?</td>
</tr>
<tr>
<td>S</td>
<td>S</td>
<td>Similar segments?</td>
<td>Are track1-segments similar (in position and length) to track2-segments, more than expected by chance?</td>
</tr>
<tr>
<td>S</td>
<td>S</td>
<td>Overlap?</td>
<td>Are the segments of track 1 overlapping the segments of track 2, more than expected by chance?</td>
</tr>
<tr>
<td>S</td>
<td>S</td>
<td>Located nearby?</td>
<td>Are the segments of track 1 closer to the segments of track 2 than expected by chance?</td>
</tr>
<tr>
<td>F</td>
<td>F</td>
<td>Correlated?</td>
<td>Are the values of track1 and track2 more positively correlated than expected by chance?</td>
</tr>
<tr>
<td>P</td>
<td>F</td>
<td>Higher values at locations?</td>
<td>Are the values of track 2 higher at the points of track 1, than what is expected by chance?</td>
</tr>
<tr>
<td>S</td>
<td>F</td>
<td>Higher values inside?</td>
<td>Are the values of track 2 higher inside the segments of track 1, than what is expected by chance?</td>
</tr>
<tr>
<td>P</td>
<td>VS</td>
<td>Located in segments with high values?</td>
<td>Does the number of track1-points that fall in track2-segments depend on the value of track2-segments?</td>
</tr>
<tr>
<td>S</td>
<td>VP</td>
<td>Higher values inside segments?</td>
<td>Do the points of track 2 that occur inside segments of track 1 have higher values than points occurring outside the segments of track 1?</td>
</tr>
<tr>
<td>VP</td>
<td>VP</td>
<td>Nearby values similar?</td>
<td>When track1-points and track2-point are nearby each other, are the values more similar than expected by chance?</td>
</tr>
<tr>
<td>P</td>
<td>VS (c/c)</td>
<td>Located in case segments</td>
<td>Does the number of track1-points that fall in track2-segments depend on whether the track2-segments are marked as case or control?</td>
</tr>
<tr>
<td>VS (c/c)</td>
<td>S</td>
<td>Preferential overlap?</td>
<td>Are the segments of track 1 marked as case overlapping unexpectedly more with the segments of track 2 than the segments of track 1 marked as control?</td>
</tr>
<tr>
<td>VP (cat)</td>
<td>VS (cat)</td>
<td>Category pairs differentially co-located?</td>
<td>Which categories of track1-points fall more inside which categories of track2-segments?</td>
</tr>
<tr>
<td>LGP</td>
<td>P</td>
<td>Colocalized in 3D?</td>
<td>Are the points of track 2 closer in 3D (as defined by track 1) than expected by chance?</td>
</tr>
</tbody>
</table>

Continued on next page
### Table A.1 – Continued from previous page

<table>
<thead>
<tr>
<th>Track 1 type</th>
<th>Track 2 type</th>
<th>Statistical investigation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>P</strong></td>
<td><strong>P</strong></td>
<td>Counts</td>
<td>The number of track1-points</td>
</tr>
<tr>
<td><strong>P</strong></td>
<td><strong>P</strong></td>
<td>Frequency</td>
<td>The frequency of track1-points</td>
</tr>
<tr>
<td><strong>P</strong></td>
<td><strong>P</strong></td>
<td>Mean and variance of gaps</td>
<td>Mean and variance of gaps between track1-points</td>
</tr>
<tr>
<td><strong>P</strong></td>
<td><strong>P</strong></td>
<td>Frequency proportion</td>
<td>The proportion of all points (track1 and track2) arising from track1</td>
</tr>
<tr>
<td><strong>P</strong></td>
<td><strong>P</strong></td>
<td>Point distances</td>
<td>The distribution of distances from each track1-point to the nearest track2-point</td>
</tr>
<tr>
<td><strong>P</strong></td>
<td><strong>S</strong></td>
<td>Count inside/outside</td>
<td>The number and proportion of track1-points inside and outside track2-segments</td>
</tr>
<tr>
<td><strong>P</strong></td>
<td><strong>S</strong></td>
<td>Matrix of count inside</td>
<td>The number of track1-points inside track2-segments, for all combinations of categories from both tracks</td>
</tr>
<tr>
<td><strong>P</strong></td>
<td><strong>S</strong></td>
<td>Relative position within segments</td>
<td>The average relative position of track1-points within track2-segments</td>
</tr>
<tr>
<td><strong>P</strong></td>
<td><strong>S</strong></td>
<td>Point to segment distances</td>
<td>The distribution of distances from each track1-point to the nearest track2-segment</td>
</tr>
<tr>
<td><strong>S</strong></td>
<td><strong>S</strong></td>
<td>Bp coverage</td>
<td>The number of base pairs covered by track1</td>
</tr>
<tr>
<td><strong>S</strong></td>
<td><strong>S</strong></td>
<td>Proportional coverage</td>
<td>The proportion of total base pairs covered by track1</td>
</tr>
<tr>
<td><strong>S</strong></td>
<td><strong>S</strong></td>
<td>Avg segment length</td>
<td>The average length of segments of track1</td>
</tr>
<tr>
<td><strong>S</strong></td>
<td><strong>S</strong></td>
<td>Segment lengths</td>
<td>The distribution of lengths of each track1-segment</td>
</tr>
<tr>
<td><strong>S</strong></td>
<td><strong>S</strong></td>
<td>Coverage</td>
<td>Base pair and proportional coverage by track1, track2 and by both</td>
</tr>
<tr>
<td><strong>S</strong></td>
<td><strong>S</strong></td>
<td>Enrichment</td>
<td>The enrichment of track1 inside track2 and vice versa, at the bp level</td>
</tr>
<tr>
<td><strong>S</strong></td>
<td><strong>S</strong></td>
<td>Segment distances</td>
<td>The distribution of distances from each track1-segment to the nearest track2-segment</td>
</tr>
<tr>
<td><strong>F</strong></td>
<td><strong>F</strong></td>
<td>Mean</td>
<td>The mean value of track1</td>
</tr>
<tr>
<td><strong>F</strong></td>
<td><strong>F</strong></td>
<td>Sum</td>
<td>The sum of values of track1</td>
</tr>
<tr>
<td><strong>F</strong></td>
<td><strong>F</strong></td>
<td>Variance</td>
<td>The variance of values of track1</td>
</tr>
<tr>
<td><strong>F</strong></td>
<td><strong>P</strong></td>
<td>Min and max</td>
<td>The extreme values (min/max) of track1</td>
</tr>
<tr>
<td><strong>F</strong></td>
<td><strong>S</strong></td>
<td>Mean at points</td>
<td>The mean value of track1 at positions of track2</td>
</tr>
<tr>
<td><strong>F</strong></td>
<td><strong>S</strong></td>
<td>Mean inside and outside</td>
<td>The mean value of track1 inside track2 and outside track2</td>
</tr>
<tr>
<td><strong>F</strong></td>
<td><strong>F</strong></td>
<td>CC</td>
<td>Pearson's correlation coefficient of track1 and track2</td>
</tr>
<tr>
<td><strong>VP</strong></td>
<td><strong>S</strong></td>
<td>Values</td>
<td>The distribution of values of track1-elements</td>
</tr>
<tr>
<td><strong>VP</strong></td>
<td><strong>S</strong></td>
<td>Values inside</td>
<td>The distribution of values of track1-elements inside track2-elements</td>
</tr>
<tr>
<td><strong>VS (c/c)</strong></td>
<td><strong>P</strong></td>
<td>Inside case vs control</td>
<td>The number of track2-points that falls inside track1-segments marked as case or control</td>
</tr>
<tr>
<td><strong>VS (c/c)</strong></td>
<td><strong>VS (c/c)</strong></td>
<td>Two-by-two table of inside</td>
<td>Two-by-two table of case/control track1-points that falls inside case/control track2-segments</td>
</tr>
<tr>
<td><strong>VS (cat)</strong></td>
<td><strong>VS (cat)</strong></td>
<td>Category bp coverage</td>
<td>The number of base pairs covered by each category of track1</td>
</tr>
<tr>
<td><strong>VS (cat)</strong></td>
<td><strong>VS (cat)</strong></td>
<td>Category point count</td>
<td>The number of elements of each category of track1</td>
</tr>
<tr>
<td><strong>VP (cat)</strong></td>
<td><strong>VS (cat)</strong></td>
<td>Contingency table of inside</td>
<td>Contingency table of categorical track1-points that falls inside categorical track2-segments</td>
</tr>
<tr>
<td><strong>L</strong></td>
<td><strong>L</strong></td>
<td>Number of nodes and edges</td>
<td>The number of nodes and edges in track1</td>
</tr>
<tr>
<td><strong>L</strong></td>
<td><strong>L</strong></td>
<td>Number of neighbors</td>
<td>The distribution of the number of neighbors for each node in the graph (track1)</td>
</tr>
<tr>
<td><strong>L (w)</strong></td>
<td><strong>L (w)</strong></td>
<td>Edge weights</td>
<td>The distribution of weights for each edge of the graph (track1)</td>
</tr>
<tr>
<td><strong>L (w)</strong></td>
<td><strong>L (w)</strong></td>
<td>Clustered heatmap of graph</td>
<td>Clustered heatmap of weights of the graph (track1)</td>
</tr>
</tbody>
</table>
Table A.2: Tools for statistical, visual and specialized analyses of genomic tracks in the Genomic HyperBrowser. Further descriptions are given at the web pages of the tools themselves, along with demo buttons and links to reproducible examples of how each tool can be used. The “Analyze genomic tracks” tool is also described in **Paper II**. The table was fetched from manuscript to a recently published article not included with this thesis [211], edited to fit a two-page spread. The references in the table can be mapped as follows: (4) -> **Paper I**; (14) -> [232]; (8) -> [212]. The table was created by the author of this thesis, with contributions from the first author of the paper.

<table>
<thead>
<tr>
<th>Tool name</th>
<th>Description</th>
<th>Genomic example</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Statistical analysis</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Analyze genomic tracks</td>
<td>The main analysis interface of the Genomic HyperBrowser (4). Executes analyses on a single genomic track or on the relation between two tracks. Allows specification of additional input parameters for the analyses, specifically including the specification of alternative hypotheses and null models for the hypothesis tests. Contains 56 descriptive statistics and 20 hypothesis tests.</td>
<td>Analyze cell-specificity of active chromatin in disease regions, as described in section “Full analysis scenario.”</td>
</tr>
<tr>
<td><strong>Visual analysis of tracks</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Visualize track elements relative to anchor regions</td>
<td>Allows visualization of the distribution of track elements along chromosomes, or along custom specified bins. The specified regions are displayed vertically, in order to simplify visual comparison.</td>
<td>Visualize the detailed positioning of histone modifications relative to the TSS of a selected set of gene regions.</td>
</tr>
<tr>
<td>Create high-resolution map of track distribution along genome</td>
<td>Visualizing track elements along a line, such as in the UCSC genome browser or the relative positioning visualization tool, can necessarily only offer a global overview at a very limited resolution. This tool instead uses a fractal layout of the genome line (similar to Hilbert curve (14)) to map genome locations to individual pixels in a matrix instead of along a line, effectively increasing the resolution quadratically. Although the interpretation requires a certain effort, this form of visualization can potentially be very informative.</td>
<td>Visualize the genome-wide distribution of a densely populated track, such as repeating elements or a DNase accessibility experiment.</td>
</tr>
<tr>
<td>Create high-resolution map of multiple track distributions along genome</td>
<td>Similar to the one-track version above, but uses up to three separate color channels (red,green,blue) to visualize the presence of up to three different tracks in corresponding parts of the genome by combining their color channel values at individual pixels.</td>
<td>Visualize the comparative distribution of DNase accessibility in three different cell types to see patterns of similar and distinct accessibility.</td>
</tr>
<tr>
<td>Visualize relation between two tracks across genomic regions</td>
<td>Used to reveal complex relations between tracks along the genome. For each defined analysis region (bin), a score is calculated for both tracks, using the specified summarizing function. The resulting (x,y) scores are then visualized as a single point in a scatter plot.</td>
<td>Plot exon density vs average melting temperature in 10mbp bins along the genome.</td>
</tr>
<tr>
<td>Aggregation plot of track elements relative to anchor regions</td>
<td>Used to reveal trends of how track elements are distributed relative to a set of anchor regions (bins). All anchor regions are divided into the same number of sub-bins, and a summary statistic is calculated for each sub-bin and averaged across all anchor regions. The tool returns a plot of the average values with 95% confidence intervals.</td>
<td>Positions of histone modifications around TSS</td>
</tr>
</tbody>
</table>

*Continued on next page*
<table>
<thead>
<tr>
<th>Tool name</th>
<th>Description</th>
<th>Genomic example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Analyze co-localization of input genomic regions</td>
<td>Analyze a selected track of genome locations for spatial co-localization with respect to the three-dimensional structure of the genome, as defined using results from recent Hi-C experiments. The Hi-C data has been corrected for bias using a method presented in a recent paper (8), and further normalized by subtracting the expected signal given the sequential distance between elements.</td>
<td>Analyze whether somatic mutations in cancer are co-localized in 3D in a relevant cell type.</td>
</tr>
<tr>
<td>Perform clustering of genomic tracks</td>
<td>Used to investigate relations between multiple tracks in an unsupervised manner (manuscript submitted). This tool allows an essentially unlimited number of tracks to be selected, and further allows the distance measure to be used for the clustering to be precisely specified through selection among a varied set of notions of track similarity.</td>
<td>Analyze similarities between histone modifications in different cell types.</td>
</tr>
<tr>
<td>Analyze k-mer occurrences</td>
<td>Used to analyze global track of occurrence locations for a specified k-mer from a particular reference genome. All relevant analyses in the “Analyze genomic tracks” tool can be used.</td>
<td>Analyze correlation of a specific k-mer with other tracks, e.g. genes, to in order find functional significance.</td>
</tr>
<tr>
<td>Inspect k-mer frequency variation</td>
<td>Used to calculate and visualize the frequency distribution of a particular k-mer along a genome reference. Splits the selected analysis regions (e.g. chromosomes) into a suitable number of subregions (bins). For each bin, the number of occurrences of the selected k-mer is counted and plotted.</td>
<td>Inspect the frequency variation of a particular k-mer along the genome.</td>
</tr>
</tbody>
</table>
Table A.3: Tools for extracting genomic tracks from the Genomic HyperBrowser repository, customizing tracks into forms suitable for a subsequent analysis of interest, generating new tracks, and formatting and converting existing tracks. Further descriptions are given at the web pages of the tools themselves, along with demo buttons and links to reproducible examples of how each tool can be used. The GTrack-related tools are also described in Paper II. The table was fetched from manuscript to a recently published article not included with this thesis [211], edited to fit a two-page spread. The references in the table can be mapped as follows: (6) -> Paper II. The table was created by the author of this thesis, with contributions from the first author of the paper.

<table>
<thead>
<tr>
<th>Tool name</th>
<th>Description</th>
<th>Genomic example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Extract track from HyperBrowser repository</td>
<td>Used to extract datasets from the track repository stored on the HyperBrowser server. Datasets can be extracted in a range of different formats, and from limited regions of the genome, if needed. Also, overlapping segments can be merged.</td>
<td>Extract the RefSeq gene track, in order to expand the gene segments with the &quot;Expand BED segments&quot; tool.</td>
</tr>
</tbody>
</table>

Customize tracks

<table>
<thead>
<tr>
<th>Tool name</th>
<th>Description</th>
<th>Genomic example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Expand BED segments</td>
<td>Allows extracting start-, mid-, or endpoints of genomic intervals, as well as expanding either the original intervals or the extracted start-/end-/mid-points. This is useful in a variety of situations where an analysis of interest involves either proximity to or positioning relative to the original track elements, or where a size unification of track elements is desired (based on e.g. taking midpoints and then expanding a certain distance). Also, if the expanded region crosses any chromosome borders, this is handled correctly.</td>
<td>An example of an analysis involving both proximity and relative positioning is the analysis of histone modification frequencies in bins of particular distances relative to the upstream end points of genes (transcription start sites).</td>
</tr>
<tr>
<td>Combine two BED files into single case-control track</td>
<td>Allows combining elements from two separate data sets into a single track where the elements are denoted as case (target) or control, depending on their source. This allows analyses of how other tracks preferentially interact with case elements as opposed to control elements.</td>
<td>An example is to combine chromatin states from two different cell types as case and control elements, in order to ask whether regions associated to Multiple Sclerosis susceptibility overlap more with case than control segments. See section &quot;Full analysis scenario&quot;.</td>
</tr>
<tr>
<td>Merge multiple BED files into single categorical track</td>
<td>Allows combining elements from multiple data sets into a single track, denoted with a category that reflects their source.</td>
<td>Merge segment tracks denoting e.g. exons, introns, and intergenic regions in order to create a category track spanning the whole genome.</td>
</tr>
</tbody>
</table>

Continued on next page
<table>
<thead>
<tr>
<th>Tool Name</th>
<th>Description</th>
<th>Genomic Example</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Generate tracks</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Generate bp-level track from DNA sequence</td>
<td>Supports a rich set of possibilities for constructing tracks based on the DNA sequence itself along a reference genome.</td>
<td>Construct a bp-level track of GC content in a sliding window of selectable size along the genome.</td>
</tr>
<tr>
<td>Generate bp-level track of distance to nearest segment</td>
<td>Allows the generation of tracks giving for each bp the distance (in bps) to the nearest element in any track.</td>
<td>Generate a bp-level track of distance to nearest gene.</td>
</tr>
<tr>
<td>Generate intensity track for confounder handling</td>
<td>Generates so-called “intensity tracks” which are used in controlling for confounder tracks in particular analyses. The user selects a target track as well as a set of control tracks, i.e. a set of tracks whose influence on the target track one aims to control for. The generated intensity track defines, for each base pair, the probability that an element of the target track lands at that position during randomization. The intensity track can afterwards be selected as part of the null model specification when doing hypothesis testing through the “Analyze genomic tracks” tool.</td>
<td>Can e.g. be used to control for the influence of gene proximity when analyzing the relation between TF binding locations and active regions in a given cell type.</td>
</tr>
<tr>
<td>Generate k-mer occurrence track</td>
<td>Generates a global track of occurrence locations for a specified k-mer on a particular reference genome.</td>
<td>Generate a track of all occurrences of the 8-mer “ACGTTGCA” in the human hg19 genome assembly.</td>
</tr>
<tr>
<td>Generate track of genes associated with literature terms (using Coremine)</td>
<td>Generates a track of gene segments along the human genome, where the genes are associated with one or more specified literature terms. The associations are provided by the CoreMine medical database, which is regularly updated with term-gene associations mined from published literature.</td>
<td>Find a set of genes associated with melanoma. Each gene will have an attached p-value, denoting the strength of the association.</td>
</tr>
</tbody>
</table>

**Format and convert tracks**

<table>
<thead>
<tr>
<th>Tool Name</th>
<th>Description</th>
<th>Genomic Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Convert between GTrack/Bed威海/BedGraph/GFF/FASTA files</td>
<td>The most commonly used formats for genomic location data are (arguably) the formats BED, BedGraph and WIG defined by the UCSC Genome Browser, as well as the format GFF in various versions. The tool allows converting between these formats, to the degree they are able to represent the same information. The tool also allows converting data to and from the recent GTrack format, which is a recent, unified format that is capable of representing data of any track type, and thus data stemming from any of the other file formats (6).</td>
<td>Convert a GTrack file to the BED format in order to use BED-specific Galaxy tools.</td>
</tr>
<tr>
<td>Create GTrack file from unstructured tabular data</td>
<td>The tool allows structuring unformatted tabular data into a GTrack file by specifying the necessary meta-data through simple selection boxes, inferring further properties of the data where possible.</td>
<td>Import virus integration sites of the Human Papilloma Virus (HPV) from an Excel spreadsheet into a GTrack file for further analysis by the “Analyze genomic tracks” tool.</td>
</tr>
</tbody>
</table>
Figure A.1: Schematic overview of tool categories available at the Genomic HyperBrowser server. The figure indicates at which points of a typical analysis scenario the various tools may be of use, from the initial collection and preparation of data, through customization of data to match the analysis, to the statistical evaluation of a biological hypothesis. For boxes representing several tools, the precise list of tools can be found under the corresponding header in the table that is referred to (for instance, the two tools represented by the “Format and convert” box can be found under the heading “Format and convert tracks” of table A.3). The figure was fetched from manuscript to a recently published article not included with this thesis [211]. The figure was created by the author of this thesis.
Appendix B

GTrack 1.0 specification
----------
| Specification of the GTrack file format |
----------

GTrack version: 1.0
Document version: 1.0.5
Date: 07 Apr 2012
Authors: Sveinung Gunderson, Matus Kalas, Osman Abul, Arnoldo Frigessi, Eivind Hovig, Geir Kjetil Sandve

Contents

* Reading the specification
* What is GTrack?
* Example GTrack files
* Basic specification
  i. Comments
  1. Header lines
  2. Column specification line
  3a. Bounding region specification line
  3b. Data lines
    - BED compatibility
    - Compression
    - Detailed specification of character usage
* Extended specification
  - Redefining column names
  - WIG compatibility
  - FASTA compatibility
  - Defining GTrack subtypes
* References
* Change log

Reading the specification

This document contains the complete specification of the GTrack format. As the document contains many details, we here present some reading recommendations:

- Skip the "Developer notes" sections if you are not planning to develop parsers of the GTrack format.

- The "Restrictions" section after each main type of GTrack lines contains detailed descriptions that can be skipped by most readers.

- The section "Detailed specification of character usage" contains very detailed information and can be skipped by most readers.
- The sections under "Extended specification" describes extensions that do not add any new types of information to a GTrack file, only alternative ways of expressing the same information, in addition to functionality for defining GTrack subtypes. These sections are thus not required for basic use.

A HTML version of this specification is available at [3]. In the HTML version, the sections described above are hidden by default.

---------------
What is GTrack?
---------------

GTrack is short for both "Genomic Track" and "Generic Track". GTrack is a general purpose, tabular file format for representing data in the form of genomic tracks, that is, as elements associated to positions along a reference (genome) sequence, or a set of sequences.

GTrack emphasizes preciseness, flexibility, and simple parsing. This is achieved by allowing flexible column specification and declaring syntactic properties at the beginning of the file (allowing parsers to cleanly restrict support to a subset of the GTrack specification).

A main contribution by the format is the unified and optimized formalization of sequence level genomic data into one of fifteen track types, as developed in [1]:

Points (P)
Valued Points (VP)
Segments (S)
Valued Segments (VS)
Genome Partition (GP)
Step Function (SF)
Function (F)
Linked Points (LP)
Linked Valued Points (LVP)
Linked Segments (LS)
Linked Valued Segments (LVS)
Linked Genome Partition (LGP)
Linked Step Function (LSF)
Linked Function (LF)
Linked Base Pairs (LBP)

These fifteen track types encompass most of the existing file formats, while providing support for, among other things, genomic data of a three-dimensional nature. The primary goals of the GTrack format are to support all track types systematically, simplify parsing and manipulation, allow custom extensions, and provide efficient storage.
Example GTrack files

Before delving into the details, it is recommended that you examine these examples of simple GTrack files. You may return to them while reading the rest of the specification, if needed. The first example is the simplest version of GTrack, without any specification lines. It shows a data set of a couple of genomic segments, and the track type is simply Segments (S).

# # GTrack example file 1 # # A GTrack file without headers is handled as three-column BED [2] # chr1 121 201 chr2 486 1240

The second example contains all GTrack specification lines (header line, column specification line and bounding region specification line) and shows a dataset of genomic segments with additional associated information in extra columns. One of these is selected as the main “value” of the segments, which are then of type Valued Segments (VS). The example also shows how to add custom columns.

# # GTrack example file 2 # # Note: tech is a custom column and not part of the GTrack specification # ###Track type: valued segments ###seqid tech start end value strand ###genome=hg19 chr1 ChIP-seq 1047 1165 0.625 - chr2 ChIP-chip 2002 2450 . + chr2 ChIP-chip 3033 3246 0.355 +

The third example is more advanced, showing a Step Function dataset, that is, a dataset where every base pair in the domain has an associated value, but where this value is constant, or approximated, over larger regions (250-500 bps). The domain is, in this case, composed of two bounding regions. In addition, some of the regions are linked by edges to other regions in the genome. This example file is thus of type Linked Step Function (LSF).
# GTrack example file 3
#
##Track type: linked step function
##Edge weights: true
##Undirected edges: true
###id end value edges

###seqid=chr1; start=1000; end=2250
1 1250 10 4=0.4
2 1500 7 .
3 2000 2 .
4 2250 6 1=0.4;6=0.3

###seqid=chr1; start=3000; end=4000
5 3250 7 .
6 3500 4 4=0.3
7 4000 6 .

(Note that, for readability issues, spaces are used instead of tab characters in these example files. They will therefore not work "out of the box". All example files are available as working GTrack files from [3].)
GTrack is a tabular text file format. All GTrack filenames should end with ".gtrack". The GTrack format consists of 5 different line types, distinguished by the leading characters and numbered here by order of appearance in the file:

i. Comments
   1. Header lines
   2. Column specification line
   3a. Bounding region specification line
   3b. Data lines

Note: The number preceding each line type defines the order in which the lines must be present, i.e. column specification must follow the header lines, but comments may be present anywhere. Note that a bounding region specification line must be followed by a data line, but that a file may have multiple bounding region specifications with data lines in between.

A GTrack validator is available at [3].

-------------
i. Comments
-------------

- Leading characters: #
- Example

    #This is a comment!

- Usage: Optional

   Comments are ignored by parsers and may be present anywhere in the file.

-------------

1. Header lines
-------------

- Leading characters: ##

- Format

    ##VARIABLE:[ ]*VALUE

   where
   VARIABLE = Header variable name
   [ ]* = Optional space characters
   VALUE = Header variable value
- Example

```gtrack
##gtrack version: 1.0
##track type: valued points
##value type: category
###1-indexed: False
##end inclusive: True
```

- Usage

Optional, but any header variables not declared regain their default values.

- Restrictions

* GTrack files may add custom header variables, e.g. as part of the definition of a GTrack subtype (see section "Defining GTrack subtypes"). For reserved header variables, however, the values are restricted to the ones allowed by the header variable (see below).

* All variable names and reserved variable values are treated as case insensitive and do not support character escaping. Custom values, i.e. header values of non-reserved header variables, do, however, support escaping. For more details, see the section "Detailed specification of character usage".

Header lines provide structural information readable by both humans and automatic parsers. The GTrack format defines a reserved set of header variables, each with a default value. If a header variable is not declared in the header lines, the default value is used. We encourage the use of header lines even when they contain default values as this adds to the clarity of the file and helps reduce parsing errors. The order of the header lines is unimportant.

Developer notes
----------------
As not all parsers/tools will have the need to support the full GTrack specification, developers are welcome to support only subsets. We do, however encourage all GTrack parsers to always check the GTrack header lines and give feedback to the user if a particular feature is unsupported by the parser/tool. Note that non-reserved header lines should be ignored by parsers, unless they specifically support the particular extensions. We encourage parsers to print warning outputs for any unsupported, non-reserved header lines, as they may be a result of typing errors.

Note also that, for consistency, the default values will not change in future versions of the GTrack specification.
----------------
Reserved header variables
------------------------

- GTrack version

  The version of the GTrack specification used for the file.

  Default value: 1.0

- Track type*
  one of:
  points
  valued points
  segments
  valued segments
  genome partition
  step function
  function
  linked points
  linked valued points
  linked segments
  linked valued segments
  linked genome partition
  linked step function
  linked function
  linked base pairs

  Defines the track type of a GTrack file. Each track type defines a set of core columns to be used. See the section "Column specification line" for more details.

  Default value: segments

- Value type

  one of:
  number
  binary
  character
  category

  Only used if the "value" column is defined. Defines the kind of content accepted in the value column. See the section "Column specification line" for more details.

  Default value: number
- Value dimension

  one of:
  scalar
  pair
  vector
  list

  Only used if the "value" column is defined. Defines the dimension of the content accepted in the value column. See the section "Column specification line" for more details.

  Default value: scalar

- Undirected edges*

  Only used if the "edges" column is defined. True if all edges specified in the GTrack file are undirected, else false. Note that undirected edges between two track elements must still be specified in both data lines, using the same weights.

  Default: false

- Edge weights*

  Only used if the "edges" column is defined. True if weights are specified for edges, else false. If true, all edges must have a weight specification, if false, no edges must specify weight.

  Default value: false

- Edge weight type

  one of:
  number
  binary
  character
  category

  Only used if the "edges" column is defined and the "Edge weights" header variable is set to true. Defines the kind of content accepted as edge weights. See the section "Column specification line" for more details.

  Default value: number
- Edge weight dimension

  one of:
  scalar
pair
vector
list

Only used if the "edges" column is defined and the "Edge weights" header variable is set to true. Defines the dimension of the content accepted as edge weights. See the section “Column specification line” for more details.

Default value: scalar

- Uninterrupted data lines*

  True if it is guaranteed that the data lines are not interrupted by bounding region specification lines (i.e. that more than one bounding region is specified), comments or blank lines, else false. This is used to help simple parsers.

Default value: false

- Sorted elements*

  True if it is guaranteed that all bounding regions and track elements come in sorted order. Bounding regions must be sorted first, and the track elements in each bounding region block second. Regions are sorted by the following fields, in ascending order (using only the ones that are defined): genome, seqid, start, end.

Default: false

- No overlapping elements*

  Only used for tracks of type Points and Segments, and the variations of these, i.e. Linked and/or Valued Points (VP/LP/LVP) and Linked and/or Valued Segments (VS/LS/LVS). True if it is guaranteed that no two track elements overlap, else false.

Default: false

- Circular elements*

  True if any track element or bounding region cross the coordinate borders of a circular sequence, i.e. that the "end" value is smaller than the "start" value.

Default: false
- 1-indexed

  True if the coordinates start at 1, false if the coordinates start at 0.

  Default value: false

- End inclusive

  True if the end coordinates should be included in intervals, else false. For example, if "End inclusive" is true, the position 10 is included in the interval [0,10], if false, the interval ends with 9.

  Default value: false

Developer notes

We recommend that all parsers always check the values of the header variables "1-indexed" and "End inclusive", even if only one or some settings are supported by the parser. If the values defined in a GTrack file are unsupported, the parser should fail. This greatly reduces the risk of erroneous positional information.

(Note that the section "Extended specification" includes more reserved header variables.)

* Some header lines include redundant information compared to the rest of the file. These are marked with * in the listing above. The redundant header lines are still explicitly defined for several reasons. First, in order for a human reader to easily find out which features are used in a file. Second, as a way for simple parsers that only use a subset of the specification to check whether they can parse a particular file. Third, it enables automatic validation of whether a file contains the information in the way the author intended. These header lines can be automatically extracted from the rest of a GTrack file by the "Expand GTrack headers" tool, available at [3].
2. Column specification line

- Leading characters: ###

- Format

  ###COL1  COL2  COL3...

  where
  COL1, COL2, COL3 = Column names
  " " = tab character

- Example

  ###genome  seqid  start  end  strand  geneId  score  id  edges
  (with tabs instead of spaces)

- Default value

  ###seqid  start  end
  (with tabs instead of spaces)

- Usage

  Optional, but if not defined, retains the default value.

- Restrictions

  * Column names are treated as case insensitive and do not support
    character escaping. For more details, see the section “Detailed
    specification of character usage”.

  * All column names must be unique.

The column specification line is a tab-separated list of column names.

The GTrack specification defines a set of eight reserved column names. Four of
these are associated with the four core informational properties: gaps,
lengths, values and interconnections. The specific set of core columns present
defines the track type (see [1] for more details). The GTrack format also
defines 4 reserved columns that, although they do not define track type, have
reserved meanings. The associations between the reserved columns and track
types are shown in the following table:
<table>
<thead>
<tr>
<th>Column name</th>
<th>genome seqid</th>
<th>start</th>
<th>end</th>
<th>value</th>
<th>strand</th>
<th>id</th>
<th>edges</th>
</tr>
</thead>
<tbody>
<tr>
<td>Track type:</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Points</td>
<td>(P)</td>
<td>?</td>
<td>!</td>
<td>.</td>
<td>.</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>Segments</td>
<td>(S)</td>
<td>?</td>
<td>!</td>
<td>X</td>
<td>X</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>Genome Partition</td>
<td>(GP)</td>
<td>?</td>
<td>!</td>
<td>.</td>
<td>X</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>Valued Points</td>
<td>(VP)</td>
<td>?</td>
<td>!</td>
<td>X</td>
<td>.</td>
<td>X</td>
<td>?</td>
</tr>
<tr>
<td>Valued Segments</td>
<td>(VS)</td>
<td>?</td>
<td>!</td>
<td>.</td>
<td>X</td>
<td>X</td>
<td>?</td>
</tr>
<tr>
<td>Step Function</td>
<td>(SF)</td>
<td>?</td>
<td>!</td>
<td>.</td>
<td>X</td>
<td>X</td>
<td>?</td>
</tr>
<tr>
<td>Function</td>
<td>(F)</td>
<td>?</td>
<td>!</td>
<td>.</td>
<td>.</td>
<td>X</td>
<td>?</td>
</tr>
<tr>
<td>Linked Points</td>
<td>(LP)</td>
<td>?</td>
<td>!</td>
<td>X</td>
<td>.</td>
<td>.</td>
<td>X</td>
</tr>
<tr>
<td>Linked Segments</td>
<td>(LS)</td>
<td>?</td>
<td>!</td>
<td>X</td>
<td>X</td>
<td>?</td>
<td>X</td>
</tr>
<tr>
<td>Linked Genome Partition</td>
<td>(LGP)</td>
<td>?</td>
<td>!</td>
<td>.</td>
<td>X</td>
<td>?</td>
<td>X</td>
</tr>
<tr>
<td>Linked Valued Points</td>
<td>(LVP)</td>
<td>?</td>
<td>!</td>
<td>X</td>
<td>.</td>
<td>X</td>
<td>?</td>
</tr>
<tr>
<td>Linked Valued Segments</td>
<td>(LVS)</td>
<td>?</td>
<td>!</td>
<td>.</td>
<td>X</td>
<td>X</td>
<td>?</td>
</tr>
<tr>
<td>Linked Step Function</td>
<td>(LSF)</td>
<td>?</td>
<td>!</td>
<td>.</td>
<td>X</td>
<td>X</td>
<td>?</td>
</tr>
<tr>
<td>Linked Function</td>
<td>(LF)</td>
<td>?</td>
<td>!</td>
<td>.</td>
<td>.</td>
<td>X</td>
<td>?</td>
</tr>
<tr>
<td>Linked Base Pairs</td>
<td>(LBP)</td>
<td>?</td>
<td>!</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>X</td>
</tr>
</tbody>
</table>

C - Core reserved column (defines track type)
N - Non-core reserved column (reserved, but does not define track type)
X - Column is mandatory
? - Column is optional
- - Column is not allowed
! - Property must be present, either as a column or in a bounding region specification (see below)

Table 1: Overview of the eight reserved columns in the GTrack format and their associations to track type.

Reserved columns

-------------

- genome

The genome assembly of the track element (e.g. hg19, mm9). The GTrack format has no explicit requirements on the syntax or semantics of the genome specification; the interpretation is up to the particular parsers/tools. Elements from different genomes are allowed in the same GTrack file.

Specifying the genome of a track element is optional. The genome may be specified either as a separate column in the data lines, or in a preceding bounding region specification line (see below), or both. If genome is specified both in a bounding region specification and as a column, the values must be equal.
- seqid

A sequence identifier, i.e. an identifier of the underlying sequence of the particular track element. Usually defined as chromosome (e.g. chr3, chrY, chr2_random) or scaffold (e.g. scaffold10671), as defined in the genome assembly. As with the "genome" column, the GTrack format has no explicit requirements on the syntax or semantics of the "seqid" column; the interpretation is up to the particular parsers/tools. Some parsers may for instance allow chromosome arms (e.g. chr1p) as seqid.

All track elements in a GTrack file must have a seqid, either as a separate column in the data lines, or in a preceding bounding region specification line (see below), or both. If seqid is specified both in a bounding region specification and as a column, the values must be equal.

- start

The start position of the track element, using the indexing system defined in the header (0- or 1-based).

Developer notes
-----------------

The start column is not defined for some track types (as described in Table 1). In order to still work on the start position of an element, it has to be inferred from other information in the following manner, according to track type:

Genome Partition (GP), Step Function (SF), Linked Genome Partition (LGP) and Linked Step Function (LSF):

The start position of each track element can be seen as the position immediately following the end of the track element of the previous line. The exact value of the start position depends on the "End inclusive" header variable, i.e. if the coordinates are end-exclusive, the start position of one track element should be exactly the same as the end position of the previous line, if not, the start position should be set to the previous end position + 1. For the first line in a set of data lines, the start position should be set to the start position of the preceding bounding region (see section "Bounding region specification line").

Function (F), Linked Function (LF) and Linked Base Pairs (LBP):

Each line defines a successive location along the genome. The start of the first line in a set of data lines is then the start position of the preceding bounding region. The start value is then increased by 1 for each line.
-----------------
- end

The end position of the track element, using the indexing system (0- or 1-based) and "End inclusive" property as defined in the header.

Developer notes
-------------
The end column is not defined for some track types (Points (P), Valued Points (VP), Function (F), Linked Points (LP), Linked Valued Points (LVP), Linked Function (LF) and Linked Base Pairs (LBP), as described in Table 1). In order to still work on the end position of an element, it has to be inferred from the start position. In these cases, the end position depends on the "End inclusive" header variable. If true, the end position is the same as the start position, if false, the end position is the start position + 1.

-------------

- strand

The strand of the track element. "+" for positive, "-" for negative strand, and "." when strand information is missing or irrelevant.

- value

The value or score of the track element. The character "." denotes that the track element has a missing value. The basic type of the contents follow the "Value type" header variable as follows:

number

One floating point number, e.g. -1.23, 12 or 3.1e-4. English decimal notation is used, including scientific e notation, with the period character representing the decimal separator, but with no spacing. Note that integer numbers are a subset of floating point numbers, and should use "number" as the value type.

binary

One binary value. If this value is used to denote case and control, the following notation must be used: 1 for case, 0 for control.

color

One ASCII character, e.g. A, T, C. See the section "Detailed specification of character usage" for restrictions.
category

A string defining a category. The set of all category values over all track elements form a category set, e.g: {gene, exon, promoter}. See the section “Detailed specification of character usage” for restrictions.

In addition, the “Value dimension” header variable may define that the value contains more than one instance of the basic value type, as follows:

list

A list of values, following the basic type defined in the “Value type” header variable. Lists of numbers and categories are delimited by comma, e.g. 1.23,2.34,3.45,4,5 or exon,gene,CDS,gene. Lists of binary values and characters use no delimiter, e.g. 1011011010 or ATGCTCGACG. Lists that combine different basic types are not allowed. The length of lists may vary between track elements.

The missing element character, ".", is allowed as list values, and a single missing element character denotes a zero-length list.

evector

A vector of values, similarly defined as a list, with the only difference that vectors must have the same length throughout the GTrack file.

The missing element character, ".", is allowed as vector values, but a single missing element character for the entire vector is not allowed. A vector with 3 missing numbers should thus be denoted ".,.,”.

pair

A pair of values, similarly defined as a vector, with the only limitation that the length is exactly 2.

scalar

A single value, following the basic type defined in the “Value type” header variable, e.g. 1.23, 0, g or exon, respectively.

Developer notes
-----------------
Note that the different dimensions are defined in a hierarchical manner: lists > vectors > pairs & scalars. All scalars or pairs are also vectors of length 1 or 2, respectively, and all vectors are lists. Support for lists in a parser should then also lead to the support of its “sub-dimensions”, given, of course, that the analysis allows that they are treated in an equal fashion.
-----------------
- id

An unique string identifying each track element (data line). Can be in any format, e.g. 1, aab or uc002ico.1. See the section “Detailed specification of character usage” for restrictions.

- edges

A semicolon-separated list of id’s, representing edges from the track element in the current line to the track elements which the id’s identify. A “.” character denotes that the track element has no edges. An edge is by default directed.

If the header variable "Edge weights" is set to true, each edge must have a weight value directly following, after an equals sign. The format of the weight value follows the "Edge weight type" and the "Edge weight dimension" header variables in the same way as the "value" format follows the "Value type" and "Value dimension" header variables (see above). Note that no space characters are allowed after the semicolon.

Example:

```plaintext
###seqid  start  end  id  edges
chr1    0      100  aab  aab=1.2;aac=.
chr1    200    350  aab  aaa=1.1
chr1    450    500  aac  
```

Here, the aab node is connected to the aab node with two directed edges, with the edge from aaa to aab having higher weight than the one in the other direction. Note that undirected edges must still be specified in both directions, using the same weights. This adds redundancy, but simplifies parsing. If all edges in a GTrack file are undirected, the header variable “Undirected edges” should be set to true.
3a. Bounding region specification line

- Leading characters: ####

- Format

  Type A) ####genome=VAL1

  or

  Type B) ####[genome=VAL1;[ ]*seqid=VAL2|[ ]*start=VAL3|[ ]*end=VAL4]

  where
  [x] = ”x” is optional
  [ ]* means optional space characters
  genome, seqid, start, end = reserved attribute names
  VAL1, VAL2, VAL3, VAL4 = attribute values

- Example

  ####genome=hg18; seqid=chr1; start=100; end=10000

- Usage

  Type B is mandatory for GTrack files of one of the following track types:
  Genome Partition (GP)
  Step Function (SF)
  Function (F)
  Linked Genome Partition (LGP)
  Linked Step Function (LSF)
  Linked Function (LF)
  Linked Base Pairs (LBP)

  For all other track types, bounding region specification lines are optional.

- Restrictions

  * Attribute names are treated as case insensitive and do not support character escaping. Genome and seqid values do, however, support escaping. For more details, see the section ”Detailed specification of character usage”.

  * A bounding region specification remains in effect for a set of data lines until the next bounding region specification.

  * If a GTrack file contains any bounding regions, then all elements must be enclosed by one.
* Bounding regions are not allowed to overlap.

* Bounding regions of type A and B are not allowed in the same GTrack file.

* No data lines following a bounding region of type B may have start or end positions defined outside the bounding region.

* For track types Genome Partition (GP), Step Function (SF), Linked Genome Partition (LGP) and Linked Step Function (LSF), the "end" attribute must be equal to the end position of the last track element of the block of data lines immediately following the bounding region specification line.

**Example:**

```plaintext
##track type: genome partition
###end
####seqid=chr1; start=100; end=200
125
133
200
```

* For track types Function (F), Linked Function (LF) and Linked Base Pairs (LBP), the "end" attribute must be exactly equal to the "start" attribute plus the number of data lines immediately following the bounding region specification line. If the header line "End inclusive" is true, the end position should be 1 less.

**Example:**

```plaintext
##track type: function
###value
####seqid=chr1; start=100; end=103
1.2
-0.1
0.8
```

A bounding region specifies a genomic interval encompassing the data lines that follow. A bounding region should be thought of as constituting the domain of the following track elements, i.e. the region where we have information about the properties modeled by the track elements. The set of all bounding regions of a track then constitutes the domain of the track.

Note that, in the case of Points and Segments (and the variations of these, i.e. Linked and/or Valued Points (VP/LP/LVP) and Linked and/or Valued Segments (VS/LS/LVS), see Table 1), lack of elements is also considered information. A bounding region is then, in this case, a region where we know that the lack of data means something. Areas of the genome that has not been investigated (such as centromeres) should be left outside the bounding regions. For track types other than Points and Segments (and their variations), the track elements do
by definition fill the entire domain. For example, a Function has, by
definition, a value for all base pairs in the domain. A bounding region is
then just the smallest region encompassing the track elements that follow. For
more details, see [1].

The bounding region specification comes in two flavors:

A)
The bounding region specifies the genome assembly for the following track
elements, using the same format as for the "genome" column (see the
"Column specification line" section). The domain of the track is then the
set of sequences constituting the genome, e.g. all chromosomes of the
genome. If a track contains several genomes, the domain of the track is
the collected set of sequences constituting all the specified genomes.

B)
The bounding region specifies a single sequence, or part of this sequence,
as the domain of the following track elements. The format is a set of
attribute pairs separated by semicolon and optional space characters. For
each attribute pair, the attribute name and the value are separated by the
equals sign. The attributes may appear in any order. The allowed
attributes are the following:

- genome

The genome assembly of the bounding region(e.g. hg19, mm9). The format
of the genome attribute is the same as for the "genome" column (see
the section "Column specification line"). The "genome" attribute is
optional.

- seqid

A sequence id, e.g. the id of the underlying sequence of the bounding
region. The format of the seqid attribute is the same as for the
"seqid" column (see the section "Column specification line"). The
"seqid" attribute is mandatory for a bounding region specification
line of type B.

Note that if type B bounding region specifications are not defined,
the "seqid" column must be included in the column specification line.

- start

The start position of the bounding region, using the indexing system
defined in the header (0- or 1-based). The "start" attribute is
optional.
Developer notes
-------------
If the "start" attribute is not specified, the start position of the bounding region is 0 (or 1, if the header variable "1-indexed" is true).
-------------
- end

The end position of the bounding region, using the indexing system (0- or 1-based) and "End inclusive" property as defined in the header. The "end" attribute is optional.

Developer notes
-------------
If the "end" attribute is not specified, the end position of the bounding region is the same as the end position of the sequence referenced by the "seqid" attribute, e.g. the length of the current chromosome. If the parser does not have information about the length of the sequence in question, the user should be informed, or, in the case that the bounding region is unimportant for the parser, the bounding region specification should be ignored.

Note that the restrictions regarding the "end" attribute for certain track types (see section "Restrictions" above) must still hold, even if the "end" attribute is not explicitly specified.
-------------
3b. Data lines

- Leading characters:

- Format

   VAL1  VAL2  VAL3...

   where
   VAL1, VAL2, VAL3 = column values
   " " = tab character

- Example

   chr21  304  997 - FOOGENE  423  1 .
   (with tabs instead of spaces)

- Usage

   Data lines are optional.

- Restrictions

   * Column values support character escaping, as specified in the section
     "Detailed specification of character usage".

   * The number of columns of each data line must be equal to the number of
     columns in the column definition line.

   * For track types Genome Partition (GP), Step Function (SF), Linked Genome
     Partition (LGP), and Linked Step Function (LSF), the data lines in each
     bounding region block must be sorted on the "end" value, in ascending
     order.

Each data line is a tab-separated list of values, as defined by the column
definition line. If there is a missing value in either of the "value" and
"edges" columns, the period character, ".", may be used. See the section
"Column specification line" for more details.
BED compatibility

Note that a simple BED file only using the three columns chr, start and end is directly compatible with the GTrack format. This is because the default track type of a GTrack file is Segments (S), which defines the same three core columns as a simple BED file (see Table 1). One may thus only rename the file ending of such a file from "bed" to "gtrack" and run it through a GTrack parser. If a UCSC custom track definition line or other headers are present, they must be commented out. More complex BED files must be converted. Converters to common file formats are available at [3].

Compression

As genomic tracks may contain large amounts of data, we require that fully compliant GTrack parsers support the expansion of tabular files compressed with the gzip compression algorithm [4]. Such GTrack files should have the suffix "gtrack.gz".

Detailed specification of character usage

- The GTrack format supports escaping of special characters using URL escaping conventions (%XX hex codes). All ASCII characters are supported, except the following, which must be escaped everywhere:

  Most control characters (except TAB, LF, CR): %00-%08, %0B-%0C, %0E-%1F, %7F

  Extended ASCII characters: %80 through %FF

Also, the following characters have reserved meaning, and must be escaped when used with other meanings in places where they may interfere with the parsing:

  tab (TAB): %09
  newline (LF): %0A
  carriage return (CR): %0D
  space: %20
  # (hash): %23
  % (percent): %25
  , (comma): %2C
  ; (semicolon): %3B
  = (equals): %3D
  . (period): %2E
Note that spaces needs not be escaped in data lines, as the data values are separated by tabs.

- Reserved phrases in a GTrack file receive special treatment. Reserved phrases include all header variable names, reserved header variable values (excluding custom header variable values), column names (including custom columns) and bounding region attribute names. Reserved phrases should be treated as case insensitive and do not support URL escaping.

- One must in all cases avoid starting or ending a value with unescaped whitespace.

- A line must end with the newline character (LF), optionally preceded by a carriage return (CR).

- Blank lines should be ignored by parsers.

- Comments, header lines, column specification lines and bounding region specification lines are characterized by the leading number of #-characters. Note that, except for comments, once the file reaches a certain "level" of #-characters, this count never goes down. Thus, header lines, column specification and bounding region specifications are always found in that order.

- Note that delimiter characters differ for the various lines/columns. See the specification above for details. Also note that examples in this file use spaces instead of tabs for readability. These examples should not be directly copied into GTrack files.
---

**Extended specification**

---

The extended part of the GTrack specification consists of the following header variables:

- Value column
- Edges column
- Fixed length
- Fixed gap size
- Fixed-size data lines
- Data line size
- GTrack subtype
- Subtype version
- Subtype URL
- Subtype adherence

These header variables are redundant compared to the basic GTrack specification, that is, they do not allow any extra types of information to be represented. They do, however, allow existing information to be represented in more practical ways, in addition to supporting standardized ways of extending the GTrack format by defining GTrack subtypes.

---

**Redefining column names**

---

A GTrack file may contain several columns that could be used as the "value" column, and similarly for the "edges" column. To change which columns are used, one must, as described in the basic GTrack specification, modify the column specification line. The following header variables may, however, simplify the process.

- Value column

  The name of the column to be used as the "value" column.

  Default: value

- Edges column

  The name of the column to be used as the "edges" column.

  Default: edges

Note that if either of these header variables has a non-default value, the corresponding default value ("value" or "edges") must not be included in the column specification line. The following example is thus an incorrect GTrack file:
## APPENDIX B

```
##track type: valued segments
##value column: score
###seqid start end value score
chr1 0 50 1.0 0.9
chr1 100 125 1.1 0.8
```

The following file does, however, follow the GTrack specification:

```
#
# GTrack example file 4
#
##track type: valued segments
##value column: score2
###seqid start end score1 score2
chr1 0 50 1.0 0.9
chr1 100 125 1.1 0.8
```

### Developer notes

The "Value column" and the "Edges column" header variables should be interpreted prior to parsing the column specification line. The column name referred to by the variable(s) should be renamed to "value" or "edges", respectively. If two columns in this way ends up with the same name, the parser should return an error. In this way, a parser that does not support the "Value column" and "Edges column" header variables will issue an error when a properly specified GTrack file with such headers are parsed, as, in that case, the track type will not match the column specification line according to table 1. Parsing errors are recommended over incorrect analysis results caused by erroneous interpretation of columns.

```
# WIG compatibility
--------------

The WIG format [6] includes the parameters "step" and "span", specifying a fixed step size, i.e. the distance between start positions, and a fixed span size, i.e. the length of track elements, respectively. Consider for instance the following WIG file:

```
fixedStep chrom=chr1 start=201 step=100 span=50
  25.0
  26.0
fixedStep chrom=chr2 start=151 step=100 span=50
  10.0
  11.0
```

A GTrack version of this file, using the basic specification, would look something like this, using three columns instead of one:

```
# GTrack example file 5A
#
##Track type: valued segments
##1-indexed: true
##End inclusive: true
###start  end  value
####seqid=chr1
  201  250  25.0
  301  350  26.0
####seqid=chr1
  151  200  10.0
  251  300  11.0

In order to support WIG-like functionality in GTrack, the following header variables may be used:

- **Fixed length**

  Only used when the end column is not specified. Defines a fixed length for all elements in the GTrack file.

  Restrictions:

  * fixed length >= 1

  Track type dependency:

  When fixed length > 1, the track type should be determined as though the end column is present (see Table 1).

  Default: 1

  Developer notes
  ---------------------

  Contrary to the restrictions of bounding regions of type B (see above), the end position of the segments in a bounding region is allowed to cross the region border, if implicitly defined by the "fixed length" header variable. Depending on the application, the parser must decide whether to crop the length of the elements, i.e. set the end position of any elements crossing the region border (typically the last element) equal to the end position of the surrounding bounding region.

  ---------------------

- **Fixed gap size**

  Only used when neither the start nor the end column is specified. Defines fixed-size gaps between all neighboring elements in the same bounding region. Gap size is defined as the number of uncovered base pairs between the elements. The following equation defines the relation between length, gap size and start positions:
start_n+1 = start_n + fixed length + fixed gap size

where
“start_n+1” is the start position of a track element immediately following an element with start position “start_n” in the same bounding region.

Restrictions:
* fixed length + fixed gap size > 0

* Only allowed in GTrack files using bounding regions of type B (see section “Bounding region specification line”). The start position of the first element in a bounding region is then equal to the start of the bounding region.

Track type dependency:

When fixed gap size != 0, the track type should be determined as though the start column is present (see Table 1).

Default: 0

To convert from a WIG file to a GTrack file, one may use the following formulas:

fixed length = span

fixed gap size = step - span

The WIG file shown above may then be represented in the following way as a GTrack file:

```plaintext
# GTrack example file 5B
#
##Track type: valued segments
##1-indexed: true
##End inclusive: true
##Fixed length: 50
##Fixed gap size: 50
###value
####seqid=chr1; start=201
  25.0
  26.0
####seqid=chr2; start=151
  10.0
  11.0
```
Note that the definitions above allow negative values for the variable "Fixed gap size". Such values may be used to represent sliding windows, i.e. segments that overlap with a fixed number of base pairs.

-------------------
FASTA compatibility
-------------------

The following header variables may be used to represent FASTA-like sequences [5], and other simple function tracks, such as GC content, in a condensed manner. Consider a GTrack file of type "Function", with only the value column specified:

```
#
# GTrack example file 6A
#
##Track type: function
##Value type: character
###value
####seqid=seq001
A
G
C
####seqid=seq002
G
```

This is a valid GTrack file according to the basic specification. However, reading a sequence using only one nucleotide per line is quite impractical. The following header variables changes the interpretation of the data lines:

- **Fixed-size data lines**

  True if each data line has an exact size in terms of number of characters. This is only allowed for track type Function (F), and only if the only column specified is "value". Newline and carriage return characters are ignored when parsing, and the data lines are separated using the number of characters specified in the header variable "Data line size" (below).

  **Developer notes**
  ----------------
  Note that parsers still need to be able to recognize bounding region specification lines.
  ----------------

  **Default:** false
- Data line size

   The size of each data line in terms of number of characters. Is only used if the header variable "Fixed-size data lines" (above) is true.

   Default: 1

Using these header variables, the example GTrack file shown above can be expressed in the following way:

```
#
# GTrack example file 6B
#
##Track type: function
##Value type: character
##Fixed-size data lines: true
##Data line size: 1
###value
####seqid=seq001
AGC
####seqid=seq002
GG
```

-------------------------
Defining GTrack subtypes
-------------------------

The GTrack format includes support for defining GTrack subtypes, that is, file formats that adhere to only a subset of the GTrack specification. This allows implementation of more specialized parsers, while at the same time ensuring that subtype GTrack files still work with fully compliant GTrack parsers. GTrack subtypes may also be used to standardize special GTrack configurations, removing the need for the individual GTrack files to include all the required meta information. We encourage independent specification of subtypes catering to specialized needs.

A GTrack subtype defines default values for header variables and/or the column specification line. A subtype may also add new header variables or define how parsers should interpret the values of any non-reserved columns. GTrack subtypes must still conform to the GTrack specification. Interpretation of new columns or header lines do of course require specialized parsers.

Example #1: FASTA
-------------------

As an example of the use of subtypes, we show how GTrack can be used in a similar manner as conventional FASTA files [5] (see the section "FASTA compatibility" above). Example file 7A is the subtype specification file:

```
# GTrack example file 7A
#
# Specification of FASTA subtype for GTrack.
# Available at http://gtrack.no.fasta.gtrack
#
##GTrack version: 1.0
##GTrack subtype: FASTA
##Subtype version: 1.0
##Subtype adherence: strict
##Track type: function
##Value type: character
##Fixed-size data lines: true
##Data line size: 1
##value

When using the subtype, an "online" parser will download the subtype specification file (over) and use the specified header values and/or column specification line instead of the GTrack default values. The header of a GTrack file adhering to the subtype may then be as simple as including the URL of the subtype specification, as in example file 7B:

#
# GTrack example file 7B
#
# This file makes use of the FASTA subtype specification.
#
##Subtype URL: http://gtrack.no.fasta.gtrack
###seqid=seq0001
TAGACATTACGCTAGGATGATGCGATCGATCGATCCCTCTGGATTAGGAGATCTAGATCGATGATATCCTCNN NNNNNNNATTGCTCTAGCTCTAGCTCTAGCT
###seqid=seq0002
GATTACATATCGATCGACTCGCCACTATAACTTCGAGTCTGACGATGGGGGG

GTrack subtype header lines
-------------------------------

Subtype functionality is applied with the following header variables:

- **GTrack subtype**

  The name of the subtype of the GTrack format used for the file, if any. May be specified if a GTrack file conforms to a subtype, even if the header variable "Subtype URL" is not specified.
Developer notes
-------------
Custom parsers that only support certain subtypes should check this header and give feedback to users if the subtype is not correct.
-------------

Default value: ""

- Subtype version

The version of the GTrack subtype. May be specified if a GTrack file conforms to a subtype, even if the header variable "Subtype URL" is not specified.

Default value: 1.0

- Subtype URL

URL to a GTrack file used as a specification/model for the GTrack subtype, if any. The subtype GTrack specification file is a normal GTrack file, but without bounding region specification lines or data lines. The header lines and/or the column specification line of a GTrack subtype model file is used instead of the default values for other GTrack files that adhere to the subtype. Any other specifications/restrictions should be included as comments.

The "Subtype URL" header variable is not allowed in GTrack subtype specification files.

Developer notes
-------------
If a GTrack file contains a Subtype URL header line, the subtype specification file should be downloaded by the parser. Incomplete URLs without a specified scheme (e.g. "gtrack.no") should be treated as HTTP-addresses (e.g. "http://gtrack.no"). Any inconsistencies between header lines of the GTrack files and the subtype headers should be treated according to the "Subtype adherence" header variable (see below). If the header variables "GTrack subtype" or "Subtype version" (see below) in a GTrack file do not correspond to the same header variables in the subtype specification file, the user should be informed. It is then up to the parser to decide whether or not to continue parsing.

If subtype specification downloading is not supported by the parser and a subtype URL is provided in the GTrack file, the user should be informed that he/she may use the "Expand GTrack headers" tool available at [3] in order to merge the subtype headers with the GTrack file for use in "offline" parsers.
-------------

Default value: ""
- Subtype adherence

Subtype adherence may be specified in the subtype GTrack specification file and will then regulate the way a GTrack file may override the subtype specification. The subtype adherence may also be specified in a GTrack file, and will in this case function as a signal to parsers. In this way, different parsers may allow different levels of adherence for GTrack files of the same subtype.

The following values are allowed:

strict

Values of header variables and the column specification line, as defined by the subtype, may not be overridden by the contents of a file. GTrack defaults may be overridden.

This option may be used to force users of a subtype to follow the specification exactly.

extensible

As strict, but allows redefinition of the column specification line in one aspect:

* any number of extra columns, including non-core reserved columns, may be added to the end of the column specification line. Adding core reserved columns is not allowed.

This option may be used to allow users of a subtype to add their own content, while maintaining the exact interpretation of the first columns as defined by the subtype.

redefinable

As extensible, but allows redefinition of the column specification line in another aspect:

* the "value" and "edges" columns may be redefined, i.e. any non-core column names may be renamed to "value" or "edges", and vice-versa, or the "value" and/or "edges" column may be added to the end of the column specification line.

* correspondingly, the header lines "Track type", "Value type", "Value dimension", "Undirected edges", "Edge weights", "Edge weight type", "Edge weight dimension", "Value column" and "Edges column" may also be redefined by the GTrack file.
This option may be used to allow users of a subtype to add their own content, including redefining the "value" and "edges" columns, while maintaining exactly the same content in the first columns as defined by the subtype.

reorderable

As strict, but allows redefinition of the column specification line in the following manner:

* all columns specified in the subtype specification must be included, but can be put in any order, and any extra columns may be added.

* correspondingly, the header line “Track type” may also be redefined by the GTrack file.

Note that in this case, redefinition of the “value” or "edges" columns is not allowed, as in "redefineable", but a "value" or an "edges" column may be added, if not present. This restriction guarantees consistent indentification of columns by column name.

This option may be used to allow users of a subtype to adopt their own column ordering, while at the same time maintaining that a minimum of columns must be present, identifiable by column name.

free

Everything is allowed, as long as the GTrack specification is followed.

This option leads to the subtype specification being used for no more than an alternative definition of default values of the GTrack header lines and column specification line.

Developer notes

Note that if subtype adherence is specified in the subtype specification as anything other than "free", a GTrack file using the subtype specification may not redefine this value.

Default value: free

Example #2: Short reads

As an extra example of the subtype functionality, we here propose a format for storing short reads (e.g. from ChIP-seq experiments). Again, example file 8A is the GTrack subtype specification file, and example file 8B is a GTrack file making use of the subtrack:
The "Short reads example" subtype defines two extra columns, named "read" and "quality". A read is then either the exact read (using nucleotide symbols with the exact same length as the track element) or a semicolon-separated list of colon-separated mismatches, where a mismatch is represented by a relative position and a nucleotide symbol. The reference is here the genome assembly specified in the description lines. The relative positions should follow the indexing defined by the "1-indexed" header variable. The column quality contains the quality score of the read. According to the "redefinable" subtype adherence setting, adding columns to the end are allowed. In example file 7B, the "new" column is added. Also note that the "redefinable" setting allows the redefinition of any column as a "value" column, here the "quality" column.
A set of basic GTrack subtypes are available from [3].

References


Change log

v1.0.5 - 2012.07.04:

* Clarified the start position of the first element in a bounding region when using of ”fixed gap size”.

v1.0.4 - 2012.03.22:

* Small clarification of the use of the missing value character in vectors and lists.

v1.0.3 - 2012.01.16:

* Clarified the use of the missing value character in vectors and lists.

v1.0.2 - 2012.01.09:

* Fixed typo in the explanation of the ”value column” and ”edges column” header variables.

v1.0.1 - 2011.12.30:

* Rephrased the explanation of the ”End inclusive” header variable.
* Updated citation.
* Fixed some quotation marks and capitalization issues.

v1.0 - 2011.12.23:

* First public version, included as ”Additional file 1” in [1].
Appendix C

Matrix-based counting algorithm
**Algorithm 1** Algorithm for calculating the matrix of counts of points of track 1 falling inside segments of track 2, for all combinations of categories for both tracks

**Input:**
- $N$ is the number of base pairs in the region
- $P(c)$ is the positions of track 1 points for category $c$
- $S(c)$ is the positions of bps covered by track 2 segments for category $c$
- $C_P$ is the unique categories of track 1 (of type: valued points)
- $C_S$ is the unique categories of track 2 (of type: valued segments)

```plaintext
1 function CatCountPointsInSegsMatrix($N, P(c), S(c), C_P, C_S$)

2 \[ M \leftarrow [0]_{N \times (|C_P|+|C_S|)} \]

3 for $i \leftarrow 1$ to $N$ do

4     for $j \leftarrow 1$ to $|C_P|$ do

5         \[ c_P \leftarrow C_P[j] \]

6         if $i \in P(c_P)$ then

7             \[ M_{i,j} \leftarrow 1 \]

8     for $k \leftarrow 1$ to $|C_S|$ do

9         \[ c_S \leftarrow C_S[k] \]

10        if $i \in S(c_S)$ then

11            \[ M_{i,|C_P|+k} \leftarrow 1 \]

12 for $i \leftarrow 1$ to $N$ do

13     if $\forall j \in 1$ to $|C_P| + |C_S| : M_{i,j} = 0$ then

14         Remove row $i$ from $M$

15 \[ C \leftarrow [0]_{|C_P| \times |C_S|} \]

16 \[ N_{reduced} \leftarrow \frac{|M|}{(|C_P| + |C_S|)} \quad \triangleright \text{The reduced number of rows} \]

17 for $i \leftarrow 1$ to $N_{reduced}$ do

18     for $j \leftarrow 1$ to $|C_P|$ do

19         for $k \leftarrow 1$ to $|C_S|$ do

20            if $M_{i,j} = 1 \land M_{i,|C_P|+k} = 1$ then

21                \[ C_{j,k} \leftarrow 1 \]

22 return $C$
```
Listing 1: Python file CategoryPointCountInSegsMatrixStat.py, which implements algorithm 1

```python
# Copyright (C) 2009-2013, Geir Kjetil Sandve, Sveinung Gundersen and Morten Johansen
# This file is part of The Genomic HyperBrowser.
# The Genomic HyperBrowser is free software: you can redistribute it and/or modify
# it under the terms of the GNU General Public License as published by
# the Free Software Foundation, either version 3 of the License, or
# (at your option) any later version.
# The Genomic HyperBrowser is distributed in the hope that it will be useful,
# but WITHOUT ANY WARRANTY; without even the implied warranty of
# MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
# GNU General Public License for more details.
# You should have received a copy of the GNU General Public License
# along with The Genomic HyperBrowser. If not, see <http://www.gnu.org/licenses/>.

from gold.statistic.Statistic import Statistic, StatisticNumpyMatrixSplittable
from gold.statistic.RawDataStat import RawDataStat
from gold.track.TrackFormat import TrackFormatReq
from collections import OrderedDict
import numpy

class CategoryPointCountInSegsMatrixStat(MagicStatFactory):
    pass

class CategoryPointCountInSegsMatrixStatSplittable(StatisticNumpyMatrixSplittable):
    pass

class CategoryPointCountInSegsMatrixStatUnsplittable(Statistic):
    VERSION = '1.3'
    def __init__(self, region, track, track2, calcPointTotals=False, **kwArgs):
        if type(calcPointTotals) == str:
            calcPointTotals = eval(calcPointTotals)
        self.__calcPointTotals = calcPointTotals
        Statistic.__init__(self, region, track, track2, \  
        calcPointTotals=calcPointTotals, **kwArgs)

    def _clusterOverlapsKeepCopies(self, ystarts, yends):
        assert len(ystarts) == len(yends)
        segments = numpy.concatenate((ystarts, yends))
        segments.shape = (2, len(ystarts))
        prevEl = [None]

    def firstPass(self, el):
        if prevEl[0] is not None and prevEl[0][1] == el[0]: # Overlap
            ret = numpy.array([prevEl[0][0], max(prevEl[0][1], el[1])])
        else:
            ret = el
        prevEl[0] = el
        return ret

    def secondPass(self, el):
        if prevEl[0] is not None and prevEl[0][0] == el[0]: # Overlap after first pass
```

```python
    ret = numpy.array([el[0], prevEl[0][1]])
    else:
        ret = el
        prevEl[0] = el
    return ret
segments = numpy.flipud(numpy.apply_along_axis(secondPass, axis=0, 
                        arr=numpy.flipud(segments)))

    return segments

def _compute(self):
    xTv = self._children[0].getResult()
    yTv = self._children[1].getResult()
    xCats = xTv.valsAsNumpyArray()
    yCats = yTv.valsAsNumpyArray()
    xUniqueCats = numpy.unique(xCats)
    yUniqueCats = numpy.unique(yCats)
    xNumCats = len(xUniqueCats)
    yNumCats = len(yUniqueCats)
    numBps = len(self._region)

    # A boolean matrix with each bp as row and each category of the point track as columns
    # True means bp is a point
    xPosMatrix = numpy.zeros(shape=(numBps, xNumCats), dtype='bool')
    for i, cat in enumerate(xUniqueCats):
        xPosMatrix[[xTv.startsAsNumpyArray()[:, xCats == cat], i]] = True

    # A boolean matrix with each bp as row and each category of the segment track as columns
    # True means bp is in a segment
    yPosMatrix = numpy.zeros(shape=(numBps+1, yNumCats), dtype='int')
    for i, cat in enumerate(yUniqueCats):
        yPosMatrix[[yTv.endsAsNumpyArray()[:, yCats == cat], 
                    yTv.startsAsNumpyArray()[:, yCats == cat]]] = 1
    yPosMatrix[(yTv.endsForCat, i)] = 1
    # Both matrices are concatenated side by side
    posMatrix = numpy.concatenate((xPosMatrix, yPosMatrix), axis=1)
    del xPosMatrix
    del yPosMatrix

    # Bps (rows) with no hits in either track1 or track2 is removed
    reducedPosMatrix = \n        numpy.concatenate((numpy.logical_or.reduce(posMatrix[:, :xNumCats], axis=1), \n                           numpy.logical_or.reduce(posMatrix[:, xNumCats:], axis=1)))
    reducedPosMatrix.shape = (2, len(posMatrix))
    posMatrix = posMatrix[numpy.logical_and.reduce(reducedPosMatrix)]
    del reducedPosMatrix
    if len(posMatrix) == 0:
        return None

    # Result matrix
    matrix = numpy.zeros(shape=(xNumCats, yNumCats), dtype='uint32')
```
```python
    def addCount(row):
        matrix[numpy.ix_([row[:xNumCats], row[xNumCats:]])] += 1
        return True

    numpy.apply_along_axis(addCount, axis=1, arr=posMatrix)

    rows, cols = xUniqueCats, yUniqueCats

    if self._calcPointTotals:
        pointTotals = posMatrix.sum(axis=0)[:xNumCats]
        pointTotals.shape = [xNumCats, 1]
        matrix = numpy.concatenate((matrix, pointTotals), axis=1)
        cols = numpy.concatenate((cols, numpy.array(['Totals'])))

        return ('Result': OrderedDict([('Matrix', matrix), ('Rows', rows), ('Cols', cols)]))

    def _createChildren(self):
        self._addChild(RawDataStat(self._region, self._track, \
                            TrackFormatReq(dense=False, interval=False, \
                            val='category', allowOverlaps=True)))

        self._addChild(RawDataStat(self._region, self._track2, \
                            TrackFormatReq(dense=False, interval=True, \
                            val='category', allowOverlaps=True)))
```
