BRNO UNIVERSITY OF TECHNOLOGY
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

FACULTY OF INFORMATION TECHNOLOGY
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

DEPARTMENT OF INFORMATION SYSTEMS
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

SINGLE-CELL GENOTYPING

SINGLE-CELL GENOTYPOVÁNÍ

PHD THESIS
DISERTAČNÍ PRÁCE

AUTHOR **IVAN VOGEL**
AUTOR PRÁCE

SUPERVISOR **doc. Ing. JAROSLAV ZENDULKA, CSc.**
ŠKOLITEL

BRNO 2019

Abstract

Single-cell genotyping is a challenging part of genomics that deals with insufficient amount of DNA for analysis and methods that amplify the single cell DNA introduce bias in the data. This thesis maps the state of the art of bioinformatics analysis in genomics, particularly SNP array genotyping and proposes and implements original method for tackling the noise in the single-cell data. Moreover, few original algorithms for knowledge extraction from single cell data are presented and the functionality of the workflow is demonstrated on real data from products of female meiosis.

Abstrakt

Single-cell genotypovanie je náročnou oblasťou genomiky kvôli nedostatku DNA a metódy, ktoré DNA amplifikujú, dáta zašumujú. Táto práca mapuje aktuálne trendy v bioinformatickej analýze genomických dát, najmä genotypovanie SNP micročipov a navrhuje a implementuje pôvodnú metódu na odstránenie šumu zo single-cell dát. Ďalej prezentuje niekoľko originálnych algoritmov na získavanie znalostí zo single cell dát a funcionalita celého workflow je demonštrovaná na reálnych dátach - produktoch meiózy u žien.

Keywords

genotyping, bulk DNA, single cell, machine learning, discriminant analysis, recombination, SNP array

Klíčová slova

genotypování, hromadné DNA, jediná bunka, strojové učení, diskriminační analýza, rekombinace, SNP mikročip

Reference

VOGEL, Ivan. *Single-cell genotyping*. Brno, 2019. PhD thesis. Brno University of Technology, Faculty of Information Technology. Supervisor doc. Ing. Jaroslav Zendulka, CSc.

Single-cell genotyping

Declaration

I declare that I have prepared this dissertation thesis independently, under the supervision of doc. Ing. Jaroslav Zendulka, CSc. I listed all of the literary source and publications that I have used.

.....
Ivan Vogel
June 21, 2019

Acknowledgments

I would like to thank to my supervisor, doc. Zendulka, for great support throughout my studies. I would like to thank to doc. Kejnovský and prof. Hoffmann for being great mentors. I would like to thank to my partner Hero for her patience and love and my son Arman for giving me reason to smile every day. I would like to thank to my parents and my sister for their support and love.

Table of contents

Table of contents....1
1 Introduction....4
1.1 Motivation....4
1.2 Goals of the thesis....5
1.2.1 Research questions....5
1.3 Structure of this thesis....6
2 Genomics....7
2.1 Biology....7
2.1.1 DNA....7
2.1.2 Central dogma of molecular biology....8
2.1.3 Ploidy and homologous recombination....8
2.1.4 Structural variations in human genome....10
2.1.5 Heterogeneity and single-cell genomics....11
2.2 Workflow for processing genomic data....11
2.2.1 Material collection and DNA extraction....12
2.2.2 WGA and Library preparation....12
2.2.3 Microarrays....13
2.2.4 NGS....16
2.3 Conclusion....17
3 Data preprocessing....18
3.1 SNP array data preprocessing....18
3.1.1 Problem definition....18
3.1.2 Affine transformation....19
3.1.3 Quantile normalization....21
3.1.4 MA transformation....22
3.2 NGS data preprocessing....22
3.2.1 Introduction....22
3.2.2 Problem definition....22
3.2.3 Alignment strategies....23
4 Machine learning models for genotyping....25
4.1 Problem definition....25
4.2 Neural networks....25
4.3 Mixture models....28
4.3.1 Generic definition of a mixture model....28

4.3.2 Gaussian mixture model 28
4.3.3 Parameter inference and EM algorithm 29
4.3.4 Variational Bayes GMM 30
4.4 Kernel density estimation 31
4.5 Random Forest 32
4.6 Genotyping algorithms 34
4.7 Evaluation of performance 36
4.7.1 General evaluation of a classifier 36
4.7.2 Graphical representation of performance 37
4.7.3 Specific evaluation of genotyping 37
4.8 Single-cell genotyping 38
4.9 Conclusion 39
5 Novel algorithm for noise filtration 40
5.1 Introduction 40
5.2 Datasets of biological data 41
5.2.1 Groundtruth genotype from gDNA 41
5.2.2 Single cell genotyping with standard Illumina algorithm 42
5.3 Structure of noise in single cell 43
5.4 Training dataset 45
5.4.1 Feature transformation 45
5.4.2 Feature selection 46
5.4.3 Problem of imbalanced dataset 47
5.5 SureTypeSC 47
5.5.1 Basic Random Forest layer 48
5.5.2 Refinement using Gaussian Discriminant Analysis 48
5.5.3 Scoring function 51
5.6 Implementation of the algorithm 51
6 Validation and experiments with the model 53
6.1 Cross-validation 54
6.2 Validation on independent dataset 55
6.2.1 Validation curves and ROC-AUC score 56
6.2.2 Evaluation using confusion matrix 58
6.2.3 Matrix of posterior probabilities 61
6.2.4 Reduction of error rates 62
6.3 Experiments 63
6.3.1 Number of trees in Random Forest 63
6.3.2 Partitioning the input space for GDA 64

6.3.3 Effects of balanced and imbalanced training dataset 66
6.3.4 Effect of the size of the training dataset 66
6.3.5 Effect of the inter-layer threshold 67
6.4 Conclusion 68
7 Knowledge extraction 70
7.1 Recombination events in human oocytes 70
7.1.1 Introduction 70
7.1.2 Meiotic pathways and data models 70
7.1.3 Model I – reconstructing crossovers from Duos 73
7.1.4 Model II – reconstructing crossovers from Trios 74
7.2 Gene conversions 79
7.2.1 Introduction 79
7.2.2 Detection of GCs in Trios 80
7.3 Conclusion 80
8 Applications of SureTypeSC 82
8.1 Introduction 82
8.2 Improved crossover detection 82
8.2.1 Duos 82
8.2.2 Trios 83
8.3 Direct detection of gene conversions 84
8.3.1 GenCall 84
8.3.2 SureTypeSC 85
8.3.3 Validation with NGS data 85
8.4 Detection of copy number variants 87
8.4.1 Introduction 87
8.4.2 Results 87
8.5 Detecting subpopulations in the single cells 87
8.5.1 Introduction 87
8.5.2 Methodology 88
8.5.3 Results 89
8.6 Conclusion 90
9 Conclusion 91
9.1 Summary of contribution 92
9.2 Future perspectives 92
Bibliography 94
Publications 100

1 Introduction

1.1 Motivation

It has been over half of the century since Watson and Crick discovered the molecular structure of deoxyribonucleic acid (DNA). DNA, the code of life, codes the genetic information that we inherited from our ancestors and will partly pass down to our offsprings. Every single cell within our body contains nearly the same genetic information. However, every single individual (as a system of multiple cells communicating with each other) contains a collection of single nucleotide polymorphisms (SNPs) that distinguishes him from the rest of the population. One of the crucial tasks of nowadays computational biology is to shed light on this genetic diversity. A process of revealing a particular SNP is called SNP-typing or genotyping. Genotyping is feasible thanks to powerful algorithms relying on robust statistical methods assuming sufficiently strong signal that supports a genetic variant in a population. Invariant to the screening technology used, we need to operate with sufficient amount of biological material to obtain strong signal supporting our observation, This is usually not a problem, as, i.e. blood sample from an individual would likely contain sufficient amount of genetic material from multiple cells.

Although all the cells within one individual should in theory contain the same genetic information, there is a plethora of factors related to their function that can alter the DNA dynamically during their lifetime – i.e. epigenetics or *de-novo* mutations (Junker and van Oudenaarden, 2014; Shapiro et al., 2013). Therefore, there is an increasing motivation constantly driven by new scientific discoveries to look at the DNA on a single cell level. One big driver are cancer studies, where we know that cancer cells are highly heterogeneious due to increased mutagenesis. Knowing the precise genetic structure of a cancer cell can be crucial when assessing the therapy. Another example, where analysis on single cell level is of a great benefit, is examining structure of DNA and biological processes in highly complex tissues i.e. brain (Emery and Barres, 2008). At last, but not least, reproductive biology is a field where knowing the precise genetic information of a single cell is crucial. During creation of reproductive cells in human, the process called recombination shuffles the genetic information. This shuffling is unique per cell and helps to understand potential genetic disorders transmitted to offspring, causes of miscarriages etc. A very practical example, that is possible thanks to advanced medical technology, is in-vitro fertilization. During this procedure, a set of female eggs is monitored and genetically screened and only the eggs with the best fitness (in terms of likelihood of genetic diseases) will be used for in-vitro fertilization (Handyside et al., 1992; Friedler, 2012).

When analyzing data from multiple cells in a pool (termed bulk DNA), we accept that the signal we obtain is an average of all cells in the sample due to their heterogeneity. We illustrated

many examples, where this is not good enough. Thanks to sophisticated methods of molecular biology, we are now able to sort cells from a tissue, pick an individual cell and extract DNA. The amount of single cell DNA, unlike in bulk DNA is however very little for successful screening and therefore, the DNA has to be „copy-pasted". This process is possible, it is called whole-genome amplification, but it deteriorates the signal and can compromise the whole analysis by introducing bias to the data (Spits et al., 2006; Vanneste et al., 2012).

Machine learning has been successfully applied to many interdisciplinary fields including bioinformatics, biomedical research and medicine. Algorithms and statistical methods from this field of computer sciences many times improved the state of the art biological model by providing valuable and accurate predictions in situations, where searching through the whole space of possible results would be simply too costful or fatal (Prompramote et al., 2005).

SNP array has been an affordable technology for screening variants in DNA of an individual for over a decade and a package of hardware-software solution for revealing the genotype has shown high accuracy when bulk DNA is analysed, but not the single cell (Vanneste et al., 2012).

This work gives on overview of strategies for bioinformatics analysis of genomic data and then particularly focuses on algorithms for bulk DNA and single-cell genotyping. It pinpoints the specificities of the single-cell environment compared to standard bulk DNA processing and particularly researches the anatomy of noise and possibilities of improvement of genotyping. An original machine learning algorithm is proposed that tackles the noise problem. Additionally, few algorithms have been designed for single cell genomics, particularly area of reproductive biology, that, together with the novel machine learning solution, improve the current state of the art in bioinformatics of single cell and allow to look at fine genomic events at the single cell level.

1.2 Goals of the thesis

Given the motivation in the introduction, this work primarily focuses on investigating solutions for single-cell genotyping using data from SNP arrays. Based on the literature survey, there are satisfying solutions for genotyping of bulk DNA with SNP arrays, but none of them was specifically designed to genotype DNA from a single cell. Documented by multiple studies, the single cell data contains noise caused by amplification of the genetic material. This is a problem, because errors in the data can lead to false biological conclusions and therefore compromise the whole workflow. Standard genotyping algorithms fail, because they do not assume deteroriated signal caused by erroneous amplification. As a direct consequence, they allow both, high Type I and Type II errors.

1.2.1 Research questions

To solve the problem of reliable genotyping of single cell data, we can formulate following research questions:

1. Is it possible to describe pattern of noise in the SNP array single cell data?
2. Is is possible to design a machine learning method that would distinguish the good quality data from the noise and improve precision of a single-cell genotype?

1.3 Structure of this thesis

During his doctoral studies, the author of this thesis was member of two research groups[1] and the results presented here are author's contribution to their scientific outcome. The research groups actively supported the informatics research of the author with the exclusive data and this is reflected in the structure and the content of the work.

Chapter 2 will give an overview of the necessary terms and principles of the molecular biology. As the single-cell analysis shares many steps with standard analysis of bulk genomic data, we will present a generic workflow and pinpoint the specificities of the single-cell environment. While SNP array is the main source of data in this work, other approaches, namely next generation sequencing (NGS) is a valuable validation resource and will be therefore discussed as well. Chapter 3 then presents data preprocessing strategies for SNP arrays and NGS. With the remaining chapters, the work specializes on genotyping from SNP arrays and Chapter 4 therefore explains models that were previously used in the area of genotyping or principles of the models that will be utilized for single cell genotyping in later stage of this work. Here, we also demonstrate the limitations of using standard genotyping algorithms in the single cell environment. Chapter 5 then builds up on the state of the art methods and presents a design for a new method aimed for precise single cell genotyping. Chapter 6 validates the proposed methods and presents optimal parameters of the model as a result of performed experiments. Chapter 7 presents algorithms for knowledge extraction from the single cell data and Chapter 8 then demonstrates application of the novel machine learning algorithm on these knowledge extraction methods. Finally, Chapter 9 summarizes the contribution and proposes ideas for future research.

[1] Department of Plant Developmental Genetics at the Institute of Biophysics of the Czech Academy of Sciences, in Brno, Czech Republic and Center for Chromosome Stability, Department of Cellular and Molecular Medicine, University of Copenhagen, Denmark

2 Genomics

„*Genomics is the study of the full genetic complement of an organism (the genome). It employs recombinant DNA, DNA sequencing methods, and bioinformatics to sequence, assemble, and analyse the structure and function of genomes.*“[2]

Genotyping is detection of an individual’s genotype as a collection of alleles (variants of genes). While genotyping is the central topic of this thesis, it is necessary to put it into the context of genomics and explain underlying biological processes The collection of biological terms explained in this chapter will be used throughout the thesis. On top of this, a generic workflow for processing genomic data will be presented that serves as a visual guide for the subsequent chapters.

2.1 Biology

2.1.1 DNA

Virtually every cell of our body contains a molecule of DNA. DNA consists of two long polynucleotide chains (double helix) linked together by hydrogen bond. Each chain, or strand, is composed of four types of nucleotide subunits. Nucleotides are composed of a nitrogen-containing base and a five-carbon sugar, to which is attached one or more phosphate groups (Figure 2.1A). For the nucleotides in DNA, the sugar is deoxyribose (hence the name deoxyribonucleic acid), and the base can be either adenine (A), cytosine (C), guanine (G), or thymine (T). The nucleotides are covalently linked together in a chain through the sugars and phosphates, which thus form a backbone of alternating sugar–phosphate–sugar–phosphate (Figure 2.1B). Because it is only the base that differs in each of the four types of subunits, each polynucleotide chain in DNA can be thought of as a necklace: a sugar–phosphate backbonestrung with four types of beads (the four bases A, C, G, and T). These same symbols (A, C, G, and T) are also used to denote the four different nucleotides (Alberts et al., 2015). The sequence of nucleotides encodes the phenotype – an individual’s visible traits.

The set of DNA molecules is tightly packed into chromosomes and a complete set of chromosomes is genome. The human genome consists of 23 chromosomal pairs (Figure 2.1D). Each pair consists of homologous chromosomes (or homologs; one inherited from the mother and the other one from the father). Naturally, both homologs have the same order of genomic entities (genes) and are compatible, but they differ in some nucleotides. This gives rise to alleles – different variants of genes (Figure 2.1E). The first 22 pairs of chromosomes are called autosomal chromosomes and the

[2] https://www.nature.com/subjects/genomics

last pair is sex chromosomes. The type of pairing of the sex chromosomes defines the gender - X and X for female and X and Y for male.

2.1.2 Central dogma of molecular biology

Central dogma of molecular biology is a key principle describing the transfer of genetic information stored in DNA into proteins. It consists of three basic steps:

- Replication of DNA – is a natural process of DNA amplification, where the original DNA double-helix is first depleted and gives rise to two single stranded DNA molecules (ssDNA); both ssDNAs source as a template for DNA synthesis of a complementary strand to the template DNA. The output of this process is four strands of DNA (or two double strand DNA - dsDNA).
- Transcription from DNA to RNA – DNA is transcribed to various types of RNA. The RNA that is coding genes is messenger RNA as it carries the information about the gene to ribosome. The reader is refered to Wu et al., 2014 for a comprehensive overview of other types of RNAs (transfer RNA, long and small non-coding RNAs, etc).
- Translation of RNA into proteins – a portion of RNA (mRNA) is translated to amino-acid (three ribonucleotides build a codon that is translated into amino acid); a set of amino-acids is chained up into protein. Protein is a basic structural brick of a cell.

2.1.3 Ploidy and homologous recombination

As explained in Section 2.1.1, genomic DNA of human organism has 23 pairs of homologous chromosomes. Each homologous pair is created from parental chromosomes (green and yellow strands in Figure 2.2 representing maternal and parental DNA). The number of sets of chromosomes (number of homologs per chromosome) is often refered to as **ploidy** (Hartl, 2012). Human genome is diploid by default, as it consists of two copies of each chromosome, but human sex cells are haploid and merge to a diploid genome during fertilization. During gametogenesis (generation of sex cells), the amount of genetic material first doubles and then reduces (diploid genomic DNA to haploid gametes in Figure 2.2). This type of division is also called **meiosis**. The replicated chromosomes can undergo **recombination** – a process of reciprocal exchange of genetic material between homologous chromosomes (swap of green and yellow in Figure 2.2). Recombination is considered as one of the main drivers of human evolution (Rieger et al., 1968).

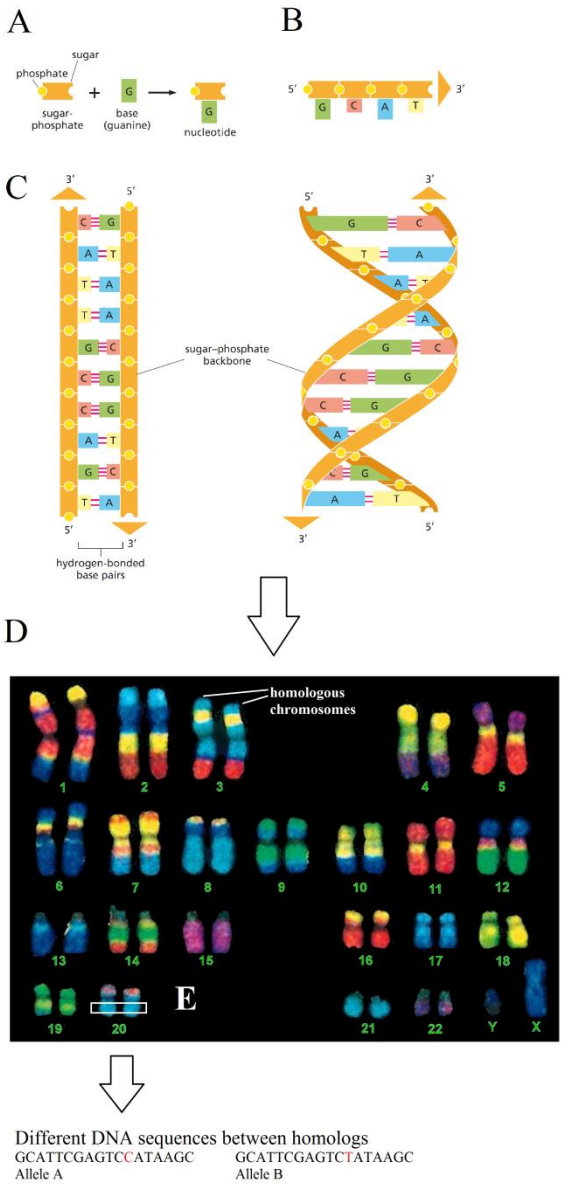

Figure 2.1. Structure and compounds of DNA. (A) Building blocks of DNA: sugar-phosphate and base build a nucleotide. (B) DNA strand composed of covalently linked nucleotides. (C) Double stranded DNA: the strands are linked by hydrogen bond .and adapt the typical double-helix structure of DNA. (D) DNA is packed into 23 chromosomal pairs.[3] (E) In each pair of chromosomes (homolog), the chromosomes have the same order of genes but the nucleotides can differ.

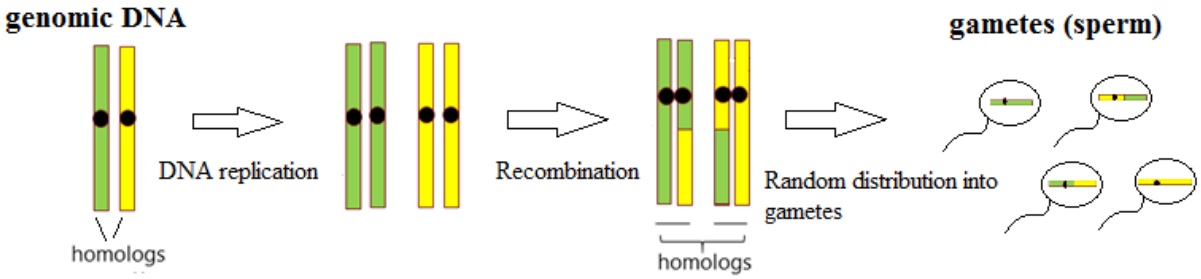

Figure 2.2. Meiosis and homologous recombination during gametogenesis. Genomic DNA is firstly replicated. The homologous chromosomes then undergo recombination – they swap part of their genetic material. This is then distributed into haploid cells.

[3] Picture taken from https://www.mun.ca/biology/scarr/FISH_chromosome_painting.html

2.1.4 Structural variations in human genome

The DNA of humans is identical in more than 99%. The rest accounts for structural variants (Feuk et al., 2006). Most of the variants are SNPs (Figure 2.3), which means that the difference is only in one nucleotide. The genomic position of the SNP is called genomic site or locus.

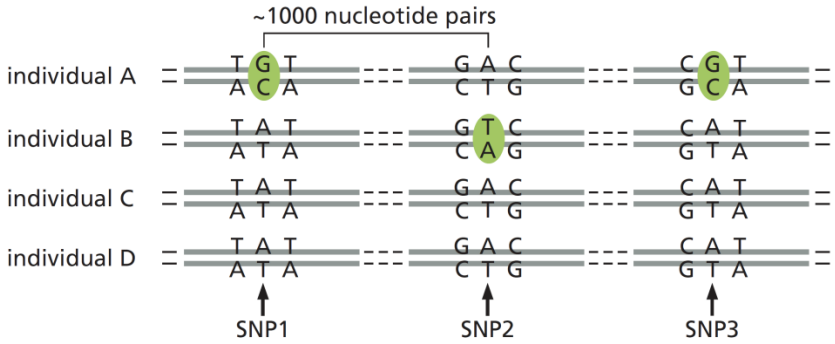

Figure 2.3. Illustration of a single nucleotide polymorphism. The individuals share most of the genotype. The differences between them are defined mostly by changes in a single nucleotide.

The genomic variants between individuals are interpreted using a common reference individual – this is usually a reference genome that has been previously established using a great amount of sequencing data and multiple bioinformatics tools to assure high quality.

Interestingly, variants that are present at more than 1% in the human population account for 90% of the variability (Collins et al., 1998). Most of these variable sites are biallelic, which means that only two variants of a gene are observed:

- Major allele A – corresponds to the reference genome
- Minor allele B – differs from the reference genome and is fixed at a rate 1% in the population or higher. This rate is called minor allele frequency (MAF).

Although the majority of the human genome shows no variation, the rest accounts for all variability including pathological phenotypes. The neighboring SNPs are usually inherited together – this gives an opportunity to reduce a full set of SNPs present in a population to tag SNPs that define a haplotype – this is a group of alleles that are inherited together from a single parent (green or yellow strands after recombination in Figure 2.2). The reduction to tag SNPs allows us to describe an individual using approximatelly 500,000 markers (The International HapMap Consortium, 2003).. The presence of tag SNPs and haplotype blocks gave rise to population studies that were aggregated in HapMap project (The International HapMap Consortium, 2003)..

The International HapMap project

The HapMap project collects the information about variability in the population (represented in tag SNPs and haplotype blocks) and draws conclusions about presence of certain phenotypes and diseases in subpopulations. HapMap screened all SNPs with minor allele frequency of 5 % or greater. The latest version of HapMap database incorporates 1.6 million SNPs from 1184 individuals from 11

global populations (The International HapMap Consortium, 2010). HapMap project with its huge statistical foundation is often considered as standard for quality assesment of SNP data (Montpetit et al., 2006) and a main resource for creating reference populations of SNPs with an optimal minor allele frequency (Illumina, Inc., 2010).

2.1.5 Heterogeneity and single-cell genomics

The heterogeneity as explained in Section 2.1.4 is not only present between individuals, but also at much finer scale, between tissues of a single individual and clones of cells from one tissue (as it is typical for i.e. . lymphocytes participating on the immune system of human) of an identical cell line (Figure 2.4). Two particular issues are specific to the single cell analysis:

- how to isolate a single cell, and
- how to increase the amount of DNA.

The latter is important because a single cell does not contain sufficient amount of DNA required by the screening technologies (Shapiro et al., 2013).

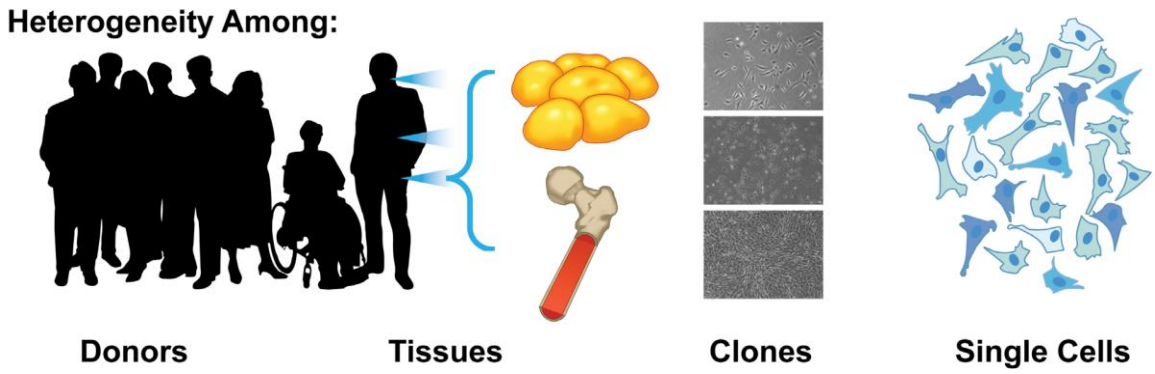

Figure 2.4. Heterogeneity is present at different levels – between individual donors of genetic information, between the tissues of a donor and clones of single cells from the same tissue. (McLeod and Mauck, 2017)

2.2 Workflow for processing genomic data

A generic workflow for processing genomic data is shown in Figure 2.5. The whole methodology can be dividied into two consecutive parts: `laboratory part` (Figure 2.5A) and `software part` (Figure 2.5B). The distinct processes from the laboratory part will be briefly presented in the subsequent text. It is important to understand that the single cell analysis requires WGA (Figure 2.5A) and to understand the principle and differences between microarray technology and NGS. The rest of the thesis (Chapter 3 onwards) will be dedicated to software part, as it is of the main interest of this work.

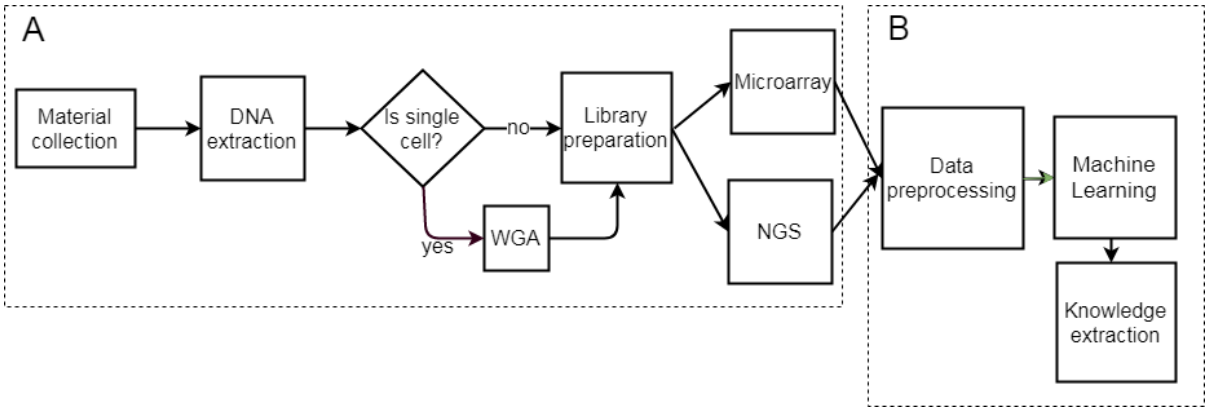

Figure 2.5. Workflow for processing of the genomic data. (A) Laboratory part involves distinct steps starting from material collection and DNA extraction. Should the material be from single cell, whole genome amplification is required. Subsequently, library of genomic data is created and screened by two commonly used technologies – microarrays or next-generation sequencing. (B) The signal from both microarray or NGS need to be preprocessed and then, a machine learning model is created from the data to predict primary information about the data (i.e. genotyping). Knowledge extraction then takes into account the genomic context and draws biological conclusions about the data (i.e. presence of recombination event)

2.2.1 Material collection and DNA extraction

Material collection is a broad term that involves isolating the target tissue in form of a biological sample. For single-cell environment, isolating the target biological samples requires isolating a single cell from a tissue. This can be currently performed by various laboratory procedures including fluroescence-activated cell sorting (FACS), magnetic-activated cell sorting (MACS), laser capture microdissection (LCM), manual cell picking and microfluidic. These methods differ in specific properties including robustness and accuracy and are discussed in detail in Hu et al. (2016).

DNA extraction is nowadays a routine process of molecular biology, supported by numerous laboratory kits and validation studies. The protocol includes cell lysis (cell membrane removal), protein denaturation and removal of other cell contaminants (Psifidi et al., 2015).

Both material collection and DNA extraction are nowadays very accurate processes and deliver purified DNA that undergoes subsequent amplification and library preparation.

2.2.2 WGA and Library preparation

WGA is a crucial step in the the single cell analysis (Figure 2.5) due to the critically low amount of DNA.We will first present polymerase chain reaction (PCR) as a "mother" of all amplification methods, followed by state of the art WGA methods. The criterion for a good performing whole genome amplification method is amplification evenness and low error rate. Evenness refers to the ability to evenly cover the whole genome (Blanshard et al., 2018).

PCR

PCR is an enzymatic assay that allows amplification of a specific DNA fragment. The DNA fragment is defined by special sequences – primers. Primers are complementary to the target sequence and

serve as extension point for the DNA polymerase . The DNA polymerase is crucial enzyme in the reaction that synthetizes many copies of the target DNA from the pool of prepared building blocks – nucleotides (Garibyan and Avashia, 2013).

PCR is a highly sensitive technique but it is highly exponential – therefore, some fragments would amplify much more frequently than the others which would negatively affect the coverage of the genome.

Whole genome amplification

The original WGA method was PCR based and is still used though its limited genome coverage. An example is improved degenerate oligonucleotide-primed PCR (improved DOP-PCR; Blagodatshik, et al., 2017). There are quasi-linear methods such as PicoPlex (Rubicon Genomics) or SurePlex (Illumina Inc.) and Multiple Annealing and Looping-Based Amplification Cycles (MALBAC; Yikon Genomics). They all have a "linear" phase, followed by a limited number of PCR-based cycles. MDA is a method based on isotermal reactions (SureMDA; Illumina Inc) and has been widely used due to relatively good coverage and low error rate of its polymerase (Blanshard et al., 2018). The most recent method is LIANTI, that uses repetitive DNA (transposons) to fragment the single-cell genome (Chen et al., 2017). The overview of the WGA methods with selected properties are displayed in Table 2.1.

Table 2.1. Overview of the WGA methods

WGA method	Genome Coverage (%)	SNP Accuracy	SNP FPR[a]	Amplification Principle	Amplification Enzyme
DOC-PCR	45	Low	2 ×10-4	Exponential	PCR DNA polymerase
MDA	87	High	$(1\text{-}2) \times 10^{-5}$	Exponential	Phi29 DNA polymerase
PicoPlex or MALBAC	73	Low	$(1\text{-}4) \times 10^{-4}$	Quasilinear	PCR DNA polymerase
LIANTI	95	High	$(2\text{-}5) \times 10^{-6}$	Linear	T7 RNA polymerase

[a] SNP false positive rate

Library preparation

Library preparation is the last step before processing the material with NGS or microarrays. The procedure involves DNA fragmentation into smaller, equally-sized pieces and ligation (attachment) of adapters. The adapter is a small sequence of DNA with known structure that acts as a barcode and enables categorization of the fragments of the analysed DNA molecules.

2.2.3 Microarrays

DNA microarray, also known as biochip or DNA chip, is a technology that allows parallel measurement of genetic information using hybridization principle. Hybridization is a biochemical reaction, where two compatible (complementary, Section 2.1) DNA strands attach to each other by

hydrogen bond to create duplex. The generic technology consists of a set of `probes` and `targets`. Probes are distributed on a solid surface and contain short fragments of DNA of known sequence. Target is the DNA of interest. The target DNA is usually fragmented into smaller pieces and fluorescently labeled to enable the detection. Once in contact with the probes on microarray, the labeled fragments specifically hybridize with the arranged probes and the signal is then measured and quantified (Figure 2.6). There are microarrays specifically designed to measure particular genetic information, in the subsequent text, we will focus on SNP arrays.

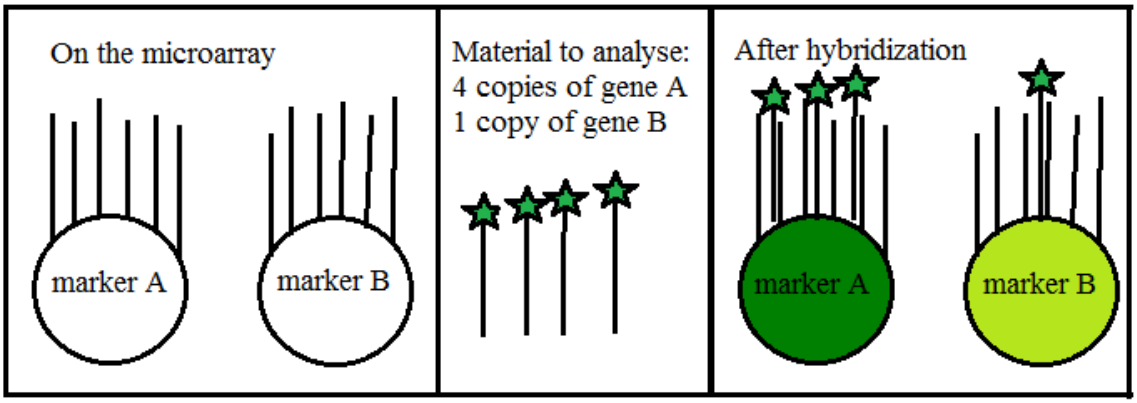

Figure 2.6. Principle of microarray. The microarray contains spots with short fragments attached to a bead (lines attached to marker A and marker B). Once hybridized with the fluorescently labeled material to analyse (target), the signal can be quantitatively assesed from the intensity of the fluorescence.

SNP arrays allow a parallel analysis of a complete set of tag SNPs (Section 2.1.4) in a cost efficient manner. There are two main technologies available in the market – SNP array from Affymetrix (Affymetrix, Inc., California, United States) and from Illumina (Illumina Inc., California, United States). These technologies share a common principle although they use different chemistries. The chip contains up to millions of micro-wells containing probes that are specifically designed to target a particular variant. The probes are fluorescently labeled and therefore the signal can be subsequently quantified, evaluated and converted to a genotype. Figure 2.7 illustrates their principle.

Affymetrix platform

Affymetrix is the first company that offered SNP array as a commercial product. Affymetrix SNP array can currently accommodate 1 Mio. Every SNP site is represented by a set of probes (probeset). Each probe is 25 nucleotides long. There are types of probes associated with allele A or B and each probe is complementary or nearly complementary to its target site. Each probeset has 24-40 probes, containing both perfect matches (PM) as well as mismatches (MM). The purpose of MM is to measure the background noise. The outcome of the raw intensity measure is a quartet of PMA, MMA, PMB and MMB. Using computational techniques, this quartet is then converted to AA, AB or BB (LaFramboise, 2009).

Illumina platform

Illumina platform called Illumina BeadChip (Illumina, Inc.) is using universal probes of 50 nucleotides. The actual genotyping takes place one nucleotide after the probe using single nucleotide extension assay (Illumina Infinium Assay protocol, Illumina Inc.). Depending on the target SNP, the complementary nucleotide carries either red or green fluorescent signal (A, T or G,C, respectively). The output of the Illumina technology is therefore one raw measurement for the A allele and one for the B allele at each SNP The number of SNPs and samples that can be analysed in parallel on one chip varies from 4 to 12 samples and from ~300k SNPs to 1 million SNPs depending on the specific product[4].

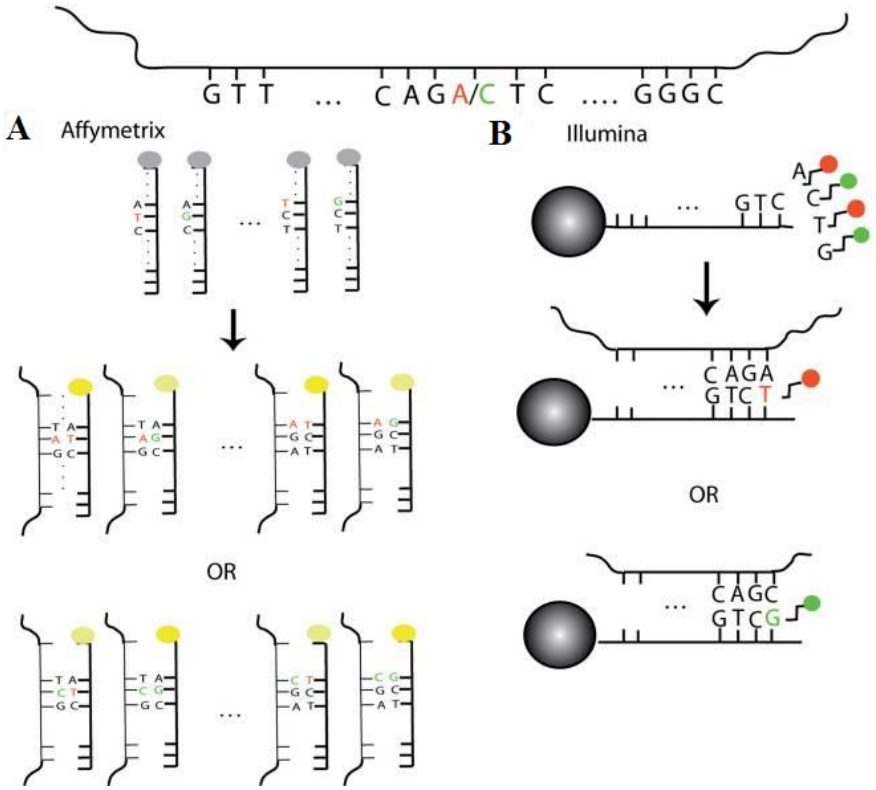

Figure 2.7. Principle of SNP array technology of two main chip producers on the market. (A) Affymetrix and (B) Illumina (LaFramboise, 2009). The SNP that is to be detected is in top and it is either an allele having A or C (A/C). The Affymetrix chemistry contains (A) probes for both alleles and therefore both types of probes will bind to the target, yet the probe with the correct allele more efficiently (all 25 bases are complementary). The quality of binding itself corresponds to the strength of the output signal. Illumina's probes (B) are universal and the actual SNP of interest is one nucleotide after the probe. Using the single nucleotide extension chemistry, a complementary nucleotide binds to the target SNP and emits a red or green signal

[4] Information and details about a particular SNP array technology available at www.illumina.com.

2.2.4 NGS

Sequencing in general, unlike microarray, that targets sequences that match the predefined probes, aims to determine the primary structure of DNA, RNA or other biomolecule. NGS is a high-troughput technology that allows analysis of millions of fragments of DNA in a parallel manner. The output of the technology is generally millions to billions short DNA reads (50 – 300 nucleotides) that correspond to random (or by a primer defined) position in DNA (Slatko et al., 2018). Two basic approaches in NGS will be presented: (a) sequencing by hybridization and (b) sequencing by synthesis (SBS).

Sequencing by hybridization

Sequencing by hybridization mostly resembles the microarray technology. Here, an array of DNA oligonucleotides of known sequence is used as a probe to hybridize to labeled fragments of the DNA to be sequenced After multiple rounds of hybridization and washing away the non-hybridized DNA, it is possible to determine whether the probe sequence contains the labeled data. This approach, similarly to microarrays, relies on specific probes and can target known SNPs or diseases, and it is often used in diagnostics (Slatko et al., 2018).

Sequencing by synthesis

Most SBS technologies are based on the following principle: individual DNA molecules to be sequenced are distributed to specific locations (well, chamber or solid substrate). DNA is then amplified in these locations. The DNA synthesis then involves incorporation of single nucleotides from a pool to the amplified DNA fragments.These nucleotides are specifically labeled per base and/or emit light when imaged to allow detection. Massive parallelism of this method allows to generate millions of DNA sequences in a sequence run and SBS is currently the prevalent technology in NGS. Two up-to-date SBS technologies are from Ion Torrent andIllumina (Slatko et al., 2018).

Ion Torrent is based on electrochemical principle and converts the nucleotide sequence into digital information directly, on a semiconductor chip. During DNA synthesis, when a correct nucleotide is incorporated, a hydrogen ion is released. This causes the change of pH and can be recorded as a change in voltage by a sensor. By sequentially flooding and washing with pool with only one type of nucleotide at a time, the voltage changes only if the appropriate nucleotide is incorporated (Slatko et al., 2018).

Illumina provides the currently leading technology in NGS. The sequencing is based on so-called bridge amplification. Briefly, fragments of DNA (500 nucleotides) are attached to a glass surface with both ends to create „bridges‘‘ and subsequently amplified 1000 times and more to create clusters. These clusters are then hybridized with fluorescently labeled nucleotides and the information about intensity is stored in a series of images.. The nucleotides are determined from the images using base-calling algorithms (Wright et al., 2017).

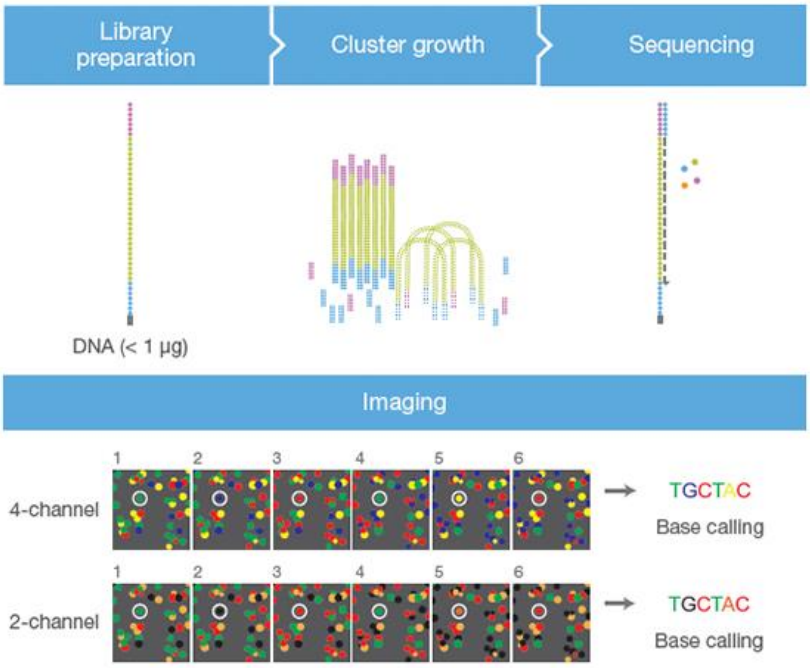

Figure 2.8. Principle of Illumina sequencing. Adapter sequences (in blue and violett) are attached to the sequenced DNA molecule to allow identification. The DNA is then guided to a flowcell, where it is amplified using bridge amplification. Elongating a DNA strand complementary to the target DNA molecule from a pool of prepared, fluorescently labeled nucleotides, determines the sequence of the molecule. The fluorescent dyes are captured – the dye strategy currently involves either 4 channel or 2 channel chemistry[5].

2.3 Conclusion

In this chapter, we presented the underlying biological principles and terms to understand the further text. In Section 2.2, we presented the generic workflow for analysing genomic data and described the laboratory processes and state of the art of the technology – microarrays (particularly SNP arrays) and NGS. These technologies share some characteristics in terms of massive parallelism and biochemical reactions. However, while microarrays require a predefined set of templates (probes) they are targeted for analysis of known, well defined markers. The genome coverage is entirely function of number of designed probes. NGS (particularly sequencing by synthesis) generally does not have this restriction and is aimed to sequence continuous fragments of the whole genome (or specific parts of the genome defined by PCR primers). They also differ in their output – while raw data of a SNP array is represented by set of signals (intensities, real values) for green and red channel (Illumina), the output of NGS is defined by set of short DNA sequences (reads, strings). The next chapter deals with strategies of preprocessing of both, SNP arrays and NGS.

[5] Image taken from Illumina, Inc, available at www.illumina.com

3 Data preprocessing

This chapter demonstrates strategies for preprocessing data coming from microarrays, namely SNP arrays and NGS. We assume the Illumina BeadChip SNP array technology (Section 2.2.3) as exclusively data from this technology will be used in the further text. The NGS (Section 2.2.4) preprocessing techniques generally do not depend on a particular vendor and therefore, the problem here will be defined in a more generic way.

3.1 SNP array data preprocessing

3.1.1 Problem definition

Every SNP *i* on an array is assigned a tuple of raw intensities d_i, defined as follows:

$$d_i = (x_i, y_i), \tag{3.1}$$

$$d_i \in A\mathbb{R}^2 \tag{3.2}$$

Where $x_i$and $y_i$corresponds to red and green raw signal intensities, respectively. $A\mathbb{R}^2$.is affine space which is a generalization of vector space. While vector space $\mathbb{R}^2$ is closed under operation of addition and scalar multiplication and point (0,0) is of a special significance (additive identity), the affine space does not hold a special significance point and the addition of two vectors is not defined (Bamberg, 1991). Intuitively, we can also introduce following fuzzy definition over the tuple d_i:

$$x_i = high \wedge y_i = low \rightarrow AA \tag{3.3}$$

$$x_i = low \wedge y_i = low \rightarrow BB \tag{3.4}$$

$$x_i = y_i \rightarrow AB \tag{3.5}$$

Where AA and BB correspond to homozygous genotype and AB is heterozygous genotype. The preprocessing of $d_i$involves two major steps:

1. Normalization
2. Background correction and outlier detection

Step 1 is important in order to perform comparisons across arrays by removing variance of non-biological origin and step 2 is important for correct interpretation of the data (Lamy et al., 2011). The motivation and need for data preprocessing is illustrated on inter-sample differences (Figure 3.1. In the following text, we will present few preprocessing strategies that tackle the problem of uneven signal between samples, as well as the *x* and *y* channels.

Notation

Consistent with previous definitions, we applied following notation for further explanation of the normalization procedures.

$$X = \{x_i; i \in < 1; N >\}$$

$$Y = \{y_i; i \in < 1; N >\}$$

$$x \in X$$

$$y \in Y$$

$$X_{sweep} \leftarrow \{(x, y);\ x \in random(X) \wedge y \in min(Y)\}$$

$$Y_{sweep} \leftarrow \{(x, y);\ x \in min(X) \wedge y \in random(Y)\}$$

Where N is the total number of markers on the chip, X_{sweep}, Y_{sweep} define sets of random selections of data boundary points for the x and y axis, respectively.

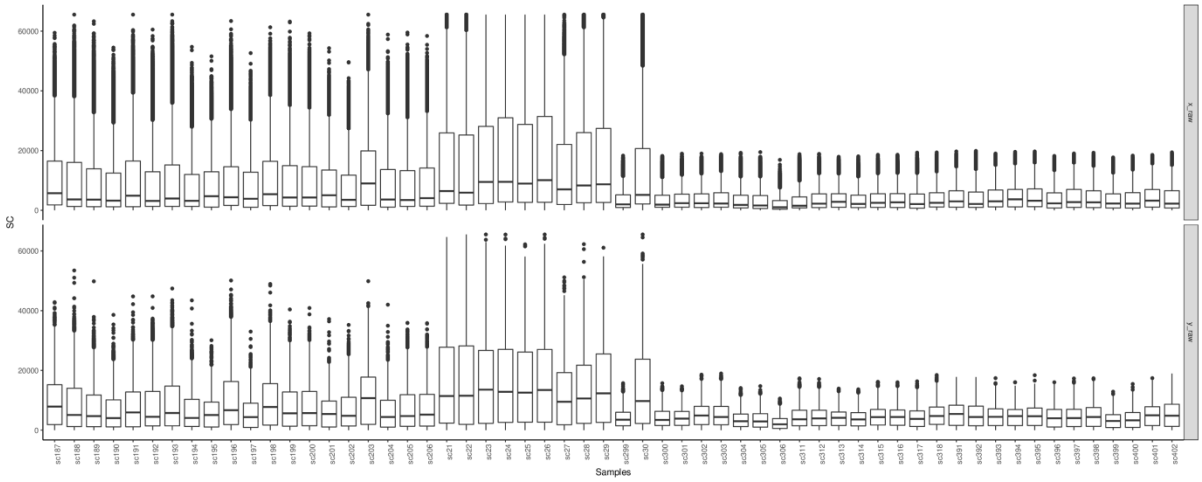

Figure 3.1. Boxplots of raw signal X (top) and Y (bottom) from multiple samples(x axis). Every boxplot contains all markers (nearly 300,000). Different properties of both, X and Y are apparent across the samples.

3.1.2 Affine transformation

The affine tranformation combines normalization and transformation of the data in one procedure.The purpose of the transformation is to translate and scale intensities that likely represent homozygous genotypes. The procedure (also refered to as 6 degree of freedom affine transformation, Kermani, 2008) consists of the following steps:

1. Outlier removal
2. Background estimation
3. Rotational estimation
4. Shear estimation
5. Scaling estimation

Outlier removal

The markers on the chip are divided into subbeadpools based on similar physical properties.The division into subbeadpools is predefined by the vendor. Within each subbeadpool, intensities are considered as outliers and are removed from the analysis,if:

- smaller than the 5th smallest value or 1st percentile of all SNPs, or

- larger than 5th largest value or 99th percentile of all SNPs

Background estimation

The goal of this step is to estimate the additive constant caused by background noise. We select 400 boundary points from X_{sweep} and Y_{sweep}. For these boundary points, linear functions are approximated and their intercept defines the new origin O':

$$O' = (\delta x, \delta y), \tag{3.6}$$

Where $(\delta x, \delta y)$ is the distance vector from the default (0,0) origin. The distance vector represents the shift of data caused by background noise. The data is therefore translated in regards to the new origin.

Rotational and shear estimation

The linear approximations of points from X_{sweep} and Y_{sweep} define the new x and y axis. The angle between the new and the original x-axis and the new and the original y-axis is measured and noted as parameter **Θ** (rotational angle) and **T** (shear), respectively.

Scaling estimation

X_{sweep} and Y_{sweep} are used again to calculate the scaling factor for the transformation of the points. Scaling factor $\boldsymbol{s_x}$ for the x-axis is defined as mean value of x-intensities from X_{sweep} and, analogously, $\boldsymbol{s_y}$ is defined as mean value of y-intensities from Y_{sweep} .

Normalization

The normalization process takes parameters calculated in the previous steps and applies Algorithm 1.

Algorithm 1: AffineNormalization

Input: *tuple d_i with x_i and y_i and parameters $\delta x, \delta y, \Theta, T, s_x, s_y$*

Output: *normalized intensities $x_{i,norm}$ and $y_{i,norm}$*

Apply pipeline in Eq. 3.7-3.14 and return $(x_{i,norm}, y_{i,norm})$

$$x_{i,1} = x_i - \delta_x \tag{3.7}$$

$$y_{i,1} = y_i - \delta_y \tag{3.8}$$

$$x_{i,2} = \cos\Theta \,.\, x_{i,1} + \sin\Theta \,.\, y_{i,1} \tag{3.9}$$

$$y_{i,2} = -\sin\Theta \,.\, x_{i,1} + \cos\Theta \,.\, y_{i,1} \tag{3.10}$$

$$x_{i,3} = x_{i,2} - \mathrm{T}.\, y_{i,2} \tag{3.11}$$

$$y_{i,3} = y_{i,2} \tag{3.12}$$

$$x_{i,norm} = x_{i,3}/s_x \tag{3.13}$$

$$y_{i,norm} = y_{i,3}/s_y \tag{3.14}$$

3.1.3 Quantile normalization

Different physical properties of the red and green dye (Section 2.2.3) cause unidentical statistical distribution of the x and y intensities and quantile normalization is a common method to use for the SNP array data to correct for this bias (Bolstad et al., 2003; Staaf et al., 2008). A need for a normalization method is illustrated in Figure 3.2. The principle is to split the two vectors of signal intensities into nearly equal sizes (quantiles) and then align them onto a diagonal in $\mathbb{R}^2$. The aligment procedure is done by substituting the original quantile value with the mean of the particular quantiles for signal x and y (Bolstad et al., 2003).

Formally, assuming $A\mathbb{R}^2$ with signal intensities X and Y, let $q_k = (q_{k,x}, q_{k,y})$ for $k = 1,..p$ be the vector of k-th quantiles for X and Y , $d = (\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}})$ is the unit diagonal and $p = |X| = |Y|$. We apply a projection formula in Eq. 3.15.

$$proj_d q_k = (\frac{1}{2}\sum_{j=x}^{\{x,y\}} q_{kj}, \frac{1}{2}\sum_{j=x}^{\{x,y\}} q_{kj}) \tag{3.15}$$

Algorithm 2 then illustrates the method. Note that this approach can be extended to n datasets, and therefore data from multiple microarrays can be compared and normalized to a common range (Bolstad et al., 2003).

Algorithm 2: QuantileNormalization

Input: *matrix M with dimensions $p \times 2$, where column 1 is dataset X and column 2 is dataset Y and $p = |X| = |Y|$*

Output: *matrix M' with normalized values*

1. *Create $M_s = sorted(M)$ by sorting every column of M*
2. *Take means across the rows of M_s and create M'_s*
3. *Get the normalized values by rearranging M'_s to have te the same order as the original matrix*

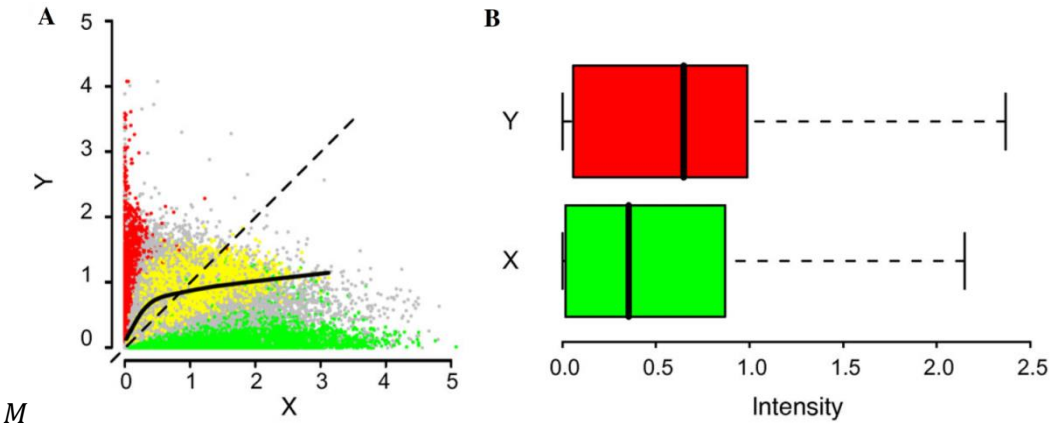

Figure 3.2. Statistical properties of X_{norm} vs. Y_{norm}. (A) Scatter plot of signal intensities normalized with affine transformation. Dashed line shows theoretical relationship, whereas the solid line is the approximated regression line indicating disproportion between x and y. Red, yellow and green correspond to homozygous BB, heterozygous AB and homozygous AA cluster (B) Boxplots of x and y indicating different distribution parameters. Taken from Staaf et al. (2008).

3.1.4 MA transformation

The MA transformation is useful in the situations when the variability between different microarray experiments and SNPs is high, which is also the case demonstrated in Figure 3.1. The idea is that instead of working with the signal values x and y directly, these are transformed to logarithmic difference (M) and logarithmic average (A). This transformation assumes that the differences in signals present in different microarray experiments and SNPs correlate with x and y. The MA transformation is an application of the Bland-Altman transformation (Bland and Altman, 1999) that has been used extensively in the analyses of gene expression data when intensity values for two channels are compared using microarrays.

Formally, we apply a linear-log transformation for every SNP i carrying a tuple of intensities (x_i, y_i) by calculating the values m_i and a_i, as follows:

$$m_i = \log_2(x_i) - \log_2(y_i) \tag{3.16}$$

$$a_i = \frac{1}{2}[\log_2(x_i) + \log_2(y_i)] \tag{3.17}$$

It has been previously shown, that m-feature has powerful discriminative ability to separate the three genotype clusters and is able to reduce variability between experiments and SNPs (Carvalho et al., 2007). The a-feature is a good general indicator of the signal quality (Ritchie et al., 2011).

3.2 NGS data preprocessing

3.2.1 Introduction

The outcome of the generic NGS technology millions of short sequences (or reads) coming from random loci in the genome. The parameter that defines the throughput is so called sequencing depth, that is calculated as LN/G, where L is read length, N is number of reads and G is size of the genome (Sims et al., 2014). Although there might be signal normalization steps involved in the NGS preprocessing, this is usually implemented in hardware an is not available for alteration. Therefore, the goal is to find the most probable origin of the sequence in the genome, also called hit or match. This process is called alignment. The alignment gives an overview of the distribution of the reads over the genome and is a fundamental step in bioinformatics of NGS. Prior to the alignment, adapter removal and quality check are applied to the NGS data (Wright et al., 2017).

3.2.2 Problem definition

Consider an alphabet $\Sigma = \{A, T, G, C\}$, where the symbols in Σ represent nucleotides (Chapter 2). Let $\bar{\Sigma} = \Sigma \cup \{-\}$, where – is symbol note in Σ called gap symbol.. Let $S = s_1, s_2, \ldots s_m$ and $T =$

$t_1, t_2 \ldots, t_n$ be two sequences over Σ. An alignment of sequences S and T is a two-row matrix A with entries in $\bar{\Sigma}$ such that:

1. The first (second) row contains the letters of S (T) in the original order
2. One or more gaps may appear between two consecutive letters of Σ in each row.
3. Each column contains at least one letter of Σ.

Following functions can be defined over matrix $\mathcal{A}$:

- A scoring function $s: \Sigma \times \Sigma \rightarrow \mathbb{Z}$ that assigns a score to each pair of nucleotides from Σ. Then, the matrix with values from all possible outputs of s is called a scoring matrix.
- A gap penalty function $\delta: k \rightarrow \mathbb{Z}^- \cup \{0\}$ that assigns a non-positive integer value to every gap of length $1 \leq k \leq \max(m, n)$ in an alignment
- An alignment score is the sum of scores of each pair of symbols

An example of $\mathcal{A}$ is in Figure 3.3

$$\mathcal{A} = \begin{pmatrix} A & - & T & A & C & - & T & G & G \\ - & G & T & C & C & G & T & - & G \end{pmatrix}$$

Figure 3.3. Example of sequence alignment

Formal definitions were adapted form Chao and Zhang, (2009) and Orlova (2010).

3.2.3 Alignment strategies

The goal of the alignment is to find the configuration of $\mathcal{A}$ that maximizes the alignment score. This is generally an NP hard problem and can be solved by dynamic programming implemented in the Neddleman-Wunsch algorithm (Needleman and Wunsch, 1970) if the global alignment is desired (i.e. aligning two sequences of similar sizes) or Smith-Waterman algorithm (Smith and Waterman, 1981), if the task is to find similar regions between two sequences (local alignment). These algorithms have quadratic time complexity (Krane, 2002).

The problem of NGS mapping means alignment of multiple sequences (`reads`) to a target (`reference`) genome. This involves millions of alignment matrices, which is computationally not tractable with dynamic programming. For these reasons, several heuristic approaches have been developed starting from BLAST (Altschul et al., 1990), BLAT (Kent, 2002) to Bowtie (Langmead et al., 2009) and BWA (Li and Durbin, 2009).

The reference genome is often not available and the sequencing depth is not high enough to perform *de novo* assembly. In this case, clustering methods are applied, which is particularly advantageous in case of repetitive elements (Novak et al., 2010). The abundance of the repetitive element in the genome in combination with a clustering method allows to create a consensus

sequence even though the coverage is low. The consensus then can be annotated and manipulated as an average element representing the population and subsequently aligned against other macromolecules sequenced by NGS (i.e. mRNA to measure transcription of transposons and small non coding RNAs to explain silencing of transposons; Kubat et al., 2014). Another strategy is to maintain the genomic clusters of repetitive elements and use secondary NGS dataset to refine the description of the original clusters and genomic abundance and subsequently measure their transcription profiles (Steflova et al., 2013). Working directly with the genomic cluster allows to capture variability of the whole population of a particular repetitive element family.

4 Machine learning models for genotyping

While the previous chapter discussed general data preprocessing in genomics of DNA, this chapter will specifically focus on computional models used for genotyping of SNP array. Referring back to the generic workflow (Figure 2.5), these algorithms correspond to the process Machine learning. Firstly, we present computational models that are either commonly used in genotyping or will be used further in the practical part of this work. All these methods fall in the machine learning area and, therefore, their inputs and outputs will be defined in a generic manner, in notation conventional for machine learning. Then, known implementations of computational models presented at the beginning of this chapter will be briefly demonstrated. Subsequently, evaluation metrics that will be used for validation in the later stages of this work will be presented. Finally, we will show the performance of selected models in the single cell environment.

4.1 Problem definition

Let $\mathcal{D}$ be a labeled set of input-output pairs $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^{N}$ called training set and N is the number of training examples. Each input x_i is a D-dimensional vector of features. Output y_i is a response variable, that can be categorical or nominal ($y_i = \{1 \dots C\}$) or real-valued. When y_i is categorical, the problem is known as classification, when y_i is real-value, the problem is called regression. When C = 2, then the problem is binary classification, if $C > 2$, it is multiclass classification. The goal is to learn a mapping from input x to output y called supervised machine learning (Murphy, 2012).

Let $\mathcal{D} = \{(x_i)\}_{i=1}^{N}$ be, consistent with previous definition, set of inputs without an assigned response variable. The task is to find pattern in the data is called unsupervised machine learning (Murphy, 2012).

Genotyping is a mapping function $\mathcal{F}$ in a multiclass classification problem $C = 3$

$$\mathcal{F}: \{d_i\}_{i=1}^{N} \rightarrow \{AA, BB, AB\} \quad (4.1)$$

Where d_i is tuple of intensities as defined in Chapter 3, Eq 3.1.

4.2 Neural networks

Neural network is a computational model inspired by biological network of neurons in human brain. A multilayer neural network consists of a set of connected input/output layers and n hidden layers with associated weights (Figure 4.1A). The hidden and output layers consist of units shown in Figure 4.1B. The learning process involves adjusting the weights to enable prediction of the correct label for the input data (Han et al., 2011).

Backpropagation algorithm

Backpropagation algorithm is a learning procedure that iteratively aims to minimize the error between the value predicted by the neural network and the actual label y_i. The algorithm consists of the following steps:

1. Initialize the weights to small random numbers
2. For each feature vector $x_i \, in \, \mathcal{D}$:
   - Pass x_i to the input layer. Every unit of the input layer passes the received feature to the associated unit in the hidden layer (Figure 4.1A). Each unit j in the hidden layer then calculates a linear combination of its inputs as follows:

     $$I_j = \sum_i w_{ij} O_i + \theta_j \tag{4.1}$$

     Where w_{ij} is the weight of connection from unit i in the previous layer to unit j, O_i is the output of unit i from the previous layer; and θ_j is the bias of the unit that acts as threshold that varies the activity of the unit. I_j is then passed to the activation function (Figure 4.1B) that is a logistic or sigmoid function. The output O_j of the unit is then calculated as follows:

     $$O_j = \frac{1}{1 + e^{-I_j}} \tag{4.2}$$

     The activation function maps the input to the interval <0,1>. The values are then passed to the output layer (assuming one hidden unit, otherwise to the next layer of hidden units). Units of the output layer process the values using the same Eq. 4.1 and 4.2.
3. The error is backpropagated by updating the weights and biases to reflect the prediction error.

   For each output unit j calculate Err_j:

   $$Err_j = O_j(1 - O_j)(y_i - O_j) \tag{4.3}$$

   Where O_j is the output value of the unit j and y_i is the label of the training example x_i.

   For each hidden layer unit j, the Err_j is calculated as weighted sum of the units in the next layer:

   $$Err_j = O_j(1 - O_j) \sum_k Err_k w_{jk} \tag{4.4}$$

   Where w_{jk} is the weight of the connection between hidden layer unit j and unit k in the next layer and Err_k is error of the unit k.
4. The weights and biases are updated in order to reflect the propagated errors using following equations:

   $$\Delta w_{ij} = l Err_j O_i \tag{4.5}$$

   $$w_{ij} = w_{ij} + \Delta w_{ij} \tag{4.6}$$

   $$\Delta \theta_j = l Err_j \tag{4.7}$$

   $$\theta_j = \theta_j + \Delta \theta_j \tag{4.8}$$

Where Δw_{ij} is change in weight w_{ij}, $\Delta\theta_j$ is change in bias θ_j and l is learning rate – a constant from interval <0,1>.

5. The algorithm terminates if one of the following conditions is met:
   - All Δw_{ij} are below a certain very small threshold, which means that further updating of the weights had little impact on quality of the model
   - The percentage of misclassified training examples is below certain threshold
   - Pre-specified number of iterations has expired

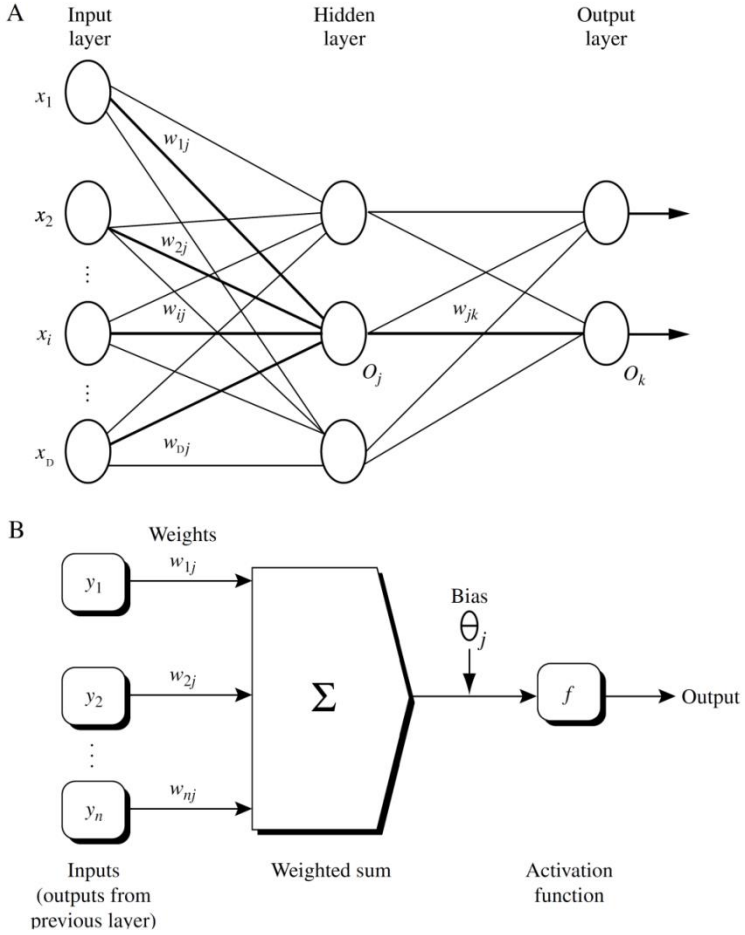

Figure 4.1. Feed-forward neural network with one hidden layer and its unit. The topology of the network (A) demonstrates fully connected nodes of a neural network with input compatible with feature vector x from $\mathcal{D}$ with D dimensions. (B) Scheme of a hidden or input unit. The values (output from previous layer) linearly combined and then mapped y an activation function to the output. Taken from Han et al., 2011.

The topology presented here corresponds to multilayer feed-forward neural network, where all nodes are fully connected. It has been previously shown that this type of neural network can approximate any continuous function (Hornik et al., 1989). In general, the topology of the network and type of the activation function are largely determined by the application domain (Mago and Bhatia, 2012).

Although training time of the neural networks is comparatively long and the interpretability (system of weights, hidden units and number of training parameters) is often difficult, they deal very

well with noisy continuous data and are able to classify patterns on which they have been not trained. Normalization of the data is crucial prior to the training (Han et al., 2011).

4.3 Mixture models

Mixture models are probabilistic models falling into category of unsupervised learning methods. They assume subpopulations in the input data that can be modeled by a simpler probabilitic distribution. The total distribution of the model is then composed by linear combination of the simpler models. Mixture models are viewed as soft-clustering methods. Soft in this context means that a membership of a training example to a cluster is defined by a probabilistic density function and is not fixed as in case of hard-clustering (i.e. k-means).

4.3.1 Generic definition of a mixture model

Formally, having training example $(x_i) \in \mathcal{D}$ without response variable (Section 4.1), a mixture model is defined with following parameters and equations (Murphy, 2012):

- latent or indicator variable $z_i \in \{1, \ldots, K\}$ representing discrete latent state indicating the membership of x_i to one of K subpopulations called mixing components or base distributions; note that if element $z_k = 1$, then all other elements are 0
- we note the likelihood $p(x_i|z_i = k) = p_k(x_i)$, where p_k is *k*-th base distribution of any type
- the overall model is then

$$p(x_i|\,\theta) = \sum_{k=1}^{K} \pi_k p_k(x_i|\,\theta) \tag{4.9}$$

  Where π_k is mixing weight that satisfies $0 \leq \pi_k \leq 1$ and $\sum_{k=1}^{K} \pi_k = 1$ and θ is set of parameters of the mixture model.

- consistent with the initial definition of indicator variable z_i, we can incorporate it in the formula:

$$p(x_i|\,\theta) = \prod_{k=1}^{K} [\pi_k p_k(x_i|\,\theta)]^{z_k} \tag{4.10}$$

4.3.2 Gaussian mixture model

Gaussian mixture model (GMM) is the most widely used mixture model. Here, every mixing component *K* is modelled with the Gaussian distribution. Consistent with the generic form of mixture model in Eq. 4.10, for GMM we assume ($D = 1$):

- $p_k \sim N(\mu_k, \sigma_k^2)$, where μ_k is mean and σ_k^2 variance of component *k* of the Gaussian (normal) distribution N. We call this univariate normal distribution

- The complete form of the univariate GMM is then as follows:

$$p(x_i|z) = \prod_{k=1}^{K} N(x_i|\mu_k, \sigma_k)^{z_k} \quad (4.11)$$

In the practical analysis, however, D > 1 and therefore, we have to substitute variance σ_k for covariance matrix Σ_k. Covariance matrix is a generalization of variance for multiple dimensions displaying variability between the features of the training example. The distribution of component *K* is called multivariate Gaussian distribution and bivariate Gaussian distribution for *D=2* The joint distribution of $p(x_i)\_$ is given by $p(z)p(x_i|z)$, and the marginal distribution of x_i is then obtained by summing the joint distribution over all possible states of z to give (Eq. 4.12):

$$p(x_i) = \sum_{z} p(z)p(x_i|z) = \sum_{k=1}^{K} \pi_k N(x_i|\mu_k, \Sigma_k) \quad (4.12)$$

4.3.3 Parameter inference and EM algorithm

We assume N training examples from $\mathcal{D}$ with D dimensions, that we model with mixture of multivariate Gaussians The data can be represented by an $N \times D$ matrix. The indicator variable will be denoted as matrix Z with dimensions $N \times K$. Coming from the Gaussian mixture distribution Eq. 4.4, we can define a log-likelihood function:

$$\ln p(X|\pi, \mu, \Sigma) = \sum_{n=1}^{N} \ln \left\{ \sum_{k=1}^{K} \pi_k N(x_n|\mu_k, \Sigma_k) \right\} \quad (4.13)$$

The objective is to maximize this function. Calculating the maximum analytically is computationally extensive. The Expectation Maximization (EM; Dempster et al., 1977) algorithm offers an effective solution by iterativelly approximating the parameters of the mixture components to maximize the outcome of the log-likelihood function. In respect to definitions in Section 4.3.2 and assuming multivariate Gaussian mixture model, the parameters are mean, component weight and covariance.

The EM algorithm consists of two steps: Expectation step (E) and maximization step (M). These steps are repeated iteratively until convergence is reached or predefined number of iterations is exceeded. We define a responsibility γ of a component k. This is posterior probability that a random data point belongs to component k. Then, the EM algorithm is as follows:

1. Initialize the means μ, covariances Σ and the mixture weights π for every component k. This can be done arbitrarily or by pre-clustering using a hard-clustering method like k-means (Bishop, 2006)
2. **E-step**. Calculate the responsiblities based on current parameters:

$$\gamma(z_{nk}) = \frac{\pi_k N(x_n|\mu_k, \Sigma_k)}{\sum_{j=1}^{K} \pi_j N(x_n|\mu_k, \Sigma_k)} \quad (4.14)$$

3. **M-step.** Evaluate the parameters based on the responsibilities calculated in step 2.

$$\mu_k^{new} = \frac{1}{N_k}\sum_{n=1}^{N}\gamma(z_{nk})\,x_n \tag{4.15}$$

$$\Sigma_k^{new} = \frac{1}{N_k}\sum_{n=1}^{N}\gamma(z_{nk})(x_n - \mu_k^{new})(x_n - \mu_k^{new})^T \tag{4.16}$$

$$\pi_k^{new} = \frac{N_k}{N} \tag{4.17}$$

   Where $N_K = \sum_{n=1}^{N}\gamma(z_{nk})$.

4. Evaluate the log-likelihood using Eq. 4.13. Check for convergence, if not satisfied, go to step 2.

The convergence is usually defined as numerical difference between log likelihood outcomes of two adjancent iterations or a maximum number of iterations.

4.3.4 Variational Bayes GMM

In the previous definition of Gaussian Mixture, we assume that we known the number of components K a priori (Section 4.3.1 and 4.3.2). Parameter K is, however, not known in many practical problems. Variational Bayes GMM is a model that solves this problem by employing statistical distribution over the parameters of GMM. Assuming infinite K, the posterior distribution of the latent variable z can be defined by Bayesian statistics (Eq. 4.18) and the log marginal probability of the observations with integral over z (Eq. 4.19):

$$p(z|x,\theta) = \frac{p(x,w|\theta)}{p(x|\theta)} \tag{4.18}$$

$$\log p(x|\theta) = \log \int p(z,x|\theta)dz \tag{4.19}$$

Analytical compuation of integral in Eq. 4.19 is computationally intractable and therefore optimization algorithm of variational inference is applied . Similarly to EM algorithm (Section 4.3.3), algorithm of variational inference approximates the exact solution. The distribution over the parameters of the model θ is often modelled with Dirichlet process (Blei and Jordan, 2006). Therefore, the model in principle defines distribution over distribution (Teh, 2010). Assuming so called full model (where every component K has its covariance matrix Σ_k), we can define following distributions over the parameters of the model:

$$\pi_k \sim Beta(\gamma_{k,1}, \gamma_{k,2}) \tag{4.20}$$

$$\mu_k \sim N(\nu_{\mu_k}, I) \tag{4.21}$$

$$\Sigma_k \sim Wishart(\alpha_k, B_k) \tag{4.22}$$

$$z_i \sim Discrete(\nu_{z_i}) \tag{4.23}$$

Where $Beta$ distribution is parametrized by two shape parameters, $Wishart$ distribution is generalization of gamma distribution, where α_k is the degree of freedom and B_k is the scale matrix. For the probability density functions and further details on these distributions, we refer the reader to Krishnamoorthy (2006). Similarly to the EM algorithm, the algorithm iterates over x_i from $\mathcal{D}$ and updates the parameters of the distributions. The details on the update rules and statistical inference of the algorithm can be found elsewhere (Bishop, 2006; Blei and Jordan, 2006). The convergence of the algorithm is defined by computing so called variational lower bound (Blei and Jordan, 2006) – a strategy implementing the Kullback-Leibler (KL) divergence that measures the difference between two probability distributions. Here, we assume approximation of the real distribution (called reference) with the distribution estimated by variational inference. The variational bound for log likelihood of observing training examples x from $\mathcal{D}$ is then defined as the sum of all KLs for all parameters over all training samples and components (Blei and Jordan, 2006). Note that while in theory, the number of components K is infinite, there is often a restriction on this parameter in the implementation by defining the maximal number of components. However, after the parameter convergence of the variational inference algorithm, some component weights π_k might be close to zero, which practically eliminates them and maintains the good component number adaptability of this type of GMM (Pedregosa et al., 2011).

4.4 Kernel density estimation

Kernel density estimation, unlike Gaussian mixture, is a non-parametric density estimation, where no assumption about functional form of data distribution is made. The idea of the algorithm is that we treat $x_i \in \mathcal{D}$ as an indicator of high-probability density in its vicinity. The probability density at a random point depends on the distance of this point from x_i.

Formally, let $x_1, \ldots, x_n \in \mathcal{D}$ be an independent and identically distributed training example with an unknown density f. The kernel density estimation of f is then:

$$\widehat{f_h}(x) = \frac{1}{nh}\sum_{i=1}^{n} K\left(\frac{x - x_i}{h}\right) \tag{4.24}$$

Where $K()$ is a kernel and h is bandwith (smoothing parameter). Kernel is a non-negative real-valued integrable function (Han et al., 2011). Gaussian function with mean of 0 and a variance of 1 is frequently used as kernel:

$$K\left(\frac{x - x_i}{h}\right) = \frac{1}{2\pi} e^{\frac{(x-x_i)^2}{2h^2}} \tag{4.25}$$

There are other types of kernels used in machine learning, a comprehensive review can be found in Genton (2001).

4.5 Random Forest

Random Forests (Breiman, 2001) belong to a group of ensemble classifiers. Ensemble classifiers combine multiple (simpler) classifiers to create an improved composite model. Random Forest consists of system of decision trees that all contribute to final classification and k decision trees play the role of simple classifiers M_1 ... M_k in the composite model (Figure 4.2).

Decision tree is a graph based structure where the inner nodes are tests on a particular feature and the leaves represent the predicted class. As illustrated in Figure 4.3, the algorithm for creating a decision tree consists of system of questions in every node that, based on the answers, aims to perfectly split the training data into nearly homogenous classes. Two metrics are broadly used to measure the degree of inhomogeneity or inpurity: **entropy** and **Gini index.**

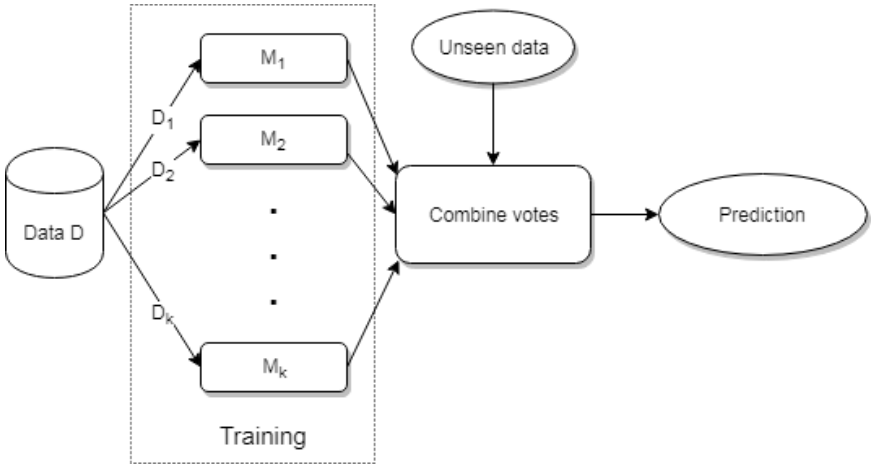

Figure 4.2. Composite classification model (adapted from Han et al., 2011). The training data D is split up into several sets that are trained on simpler models (M_1 - M_k). When applying the composite model to unseen data, the votes of simpler models are combined into a single vote resulting in prediction.

Consistent with the previous definitions, let $\mathcal{D}$ be the set of training examples having to be classified into c classes and p_i the fraction of items from $\mathcal{D}$ belonging to class i:.

- The entropy I (Eq. 4.26) is an indicator of impurity.Impurity is minimized if a single $p_i = 1$ and the rest is 0 and maximized if all the p_i*'s* are equal.

$$I = \sum_{i=1}^{c} p_i \log p_i \tag{4.26}$$

- The Gini index G (Eq. 4.27) is zero, if $\mathcal{D}$ contains only one class.

$$G = 1 - \sum_{i=1}^{c} p_i^2 \tag{4.27}$$

In context of the decision tree and selection of the node for the split, the training algorithm always seeks to minimize the weighted average of the I of the resulting children nodes:

- if a certain test on feature divides the training examples into k different subsets $\mathcal{D}_i, \ldots, D_k$, then the decision about the next test on feature (children node) is done by finding the minimum of $\sum_{j=1}^{k} \frac{\mathcal{D}_j}{\mathcal{D}} \times I(D_j)$;
- the splitting operation is done recursivelly and the training dataset is split into smaller subsets. To prevent overtraining, the procedure often finishes if there is no increase (or increase smaller than threshold) on purity of the subsets;
- alternatively, the tree can be built completely until no further division of any subtree is possible. Then, however, a procedure of tree pruning is necessary to prevent overtraining. Tree pruning deletes the unecessary nodes by collapsing or other techniques (Kearns and Mansour, 1998).

Random forest, as a composite system of decision trees that increases accuracy and prevents overfitting (Breiman, 2001), can be built using two distinct approaches:

- Bagging, and
- Forest-RC

Bagging is a procedure that assigns random subset of training data to k decision trees with replacement (some samples may occur more than once). Every decision tree is therefore trained on a subset of data and eventually on a subset of input features. The final membership of a sample to a class is decided by majority vote of all decision trees.

Forest-RC uses random linear combinations of input features instead of existing attributes. This is useful when there are only few attributes available as this reduces the correlation between the trees in the forest (Han et al., 2011).

Random Forest implements an built-in training validation metrics called out of bag (OOB) estimate or OOB score. It is calculated as prediction performance of trees that were not included for training of the current bootstrap of data.

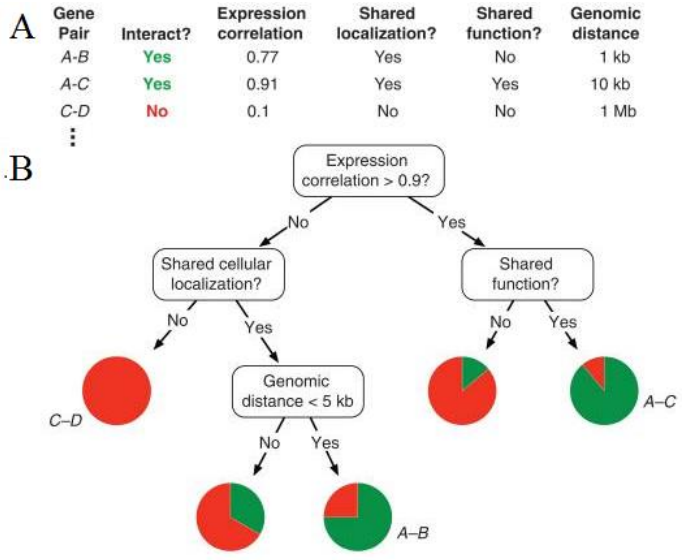

Figure 4.3. Illustration of a decision tree on a biological program of protein-protein interaction prediction. (A) Training data set with 4 input features, one response variable (Interact) and metadata column (Gene Pair). (B) Creation of

the decision tree by systematic browsing of the feature space. The pie charts indicate fractions of labeled data reaching the particular branch (Kingsford and Salzberg, 2008).

4.6 Genotyping algorithms

In relation to presented computational models in Sections 4.2 and 4.3 and normalization methods presented in Section 3.1, this section will give an overview of published implementations that use the aforementioned principles for genotyping. First, algorithms for genotyping Illumina SNP arrays (Section 2.2.3) will be discussed. Most of the algorithms have a complex statistical backround and, based on the strategy of genotyping, they can be divided into `within sample` and `reference based` models. Within sample models rely on unsupervised learning and do not require a reference dataset. On the other hand, the reference based implementations are based on some underlying genotype population that serves as reference – this is usually HapMap (Section 2.1.4).

GenCall

The GenCall algorithm is part of Illumina's genotyping module. First, raw intensities are normalized by employing affine transformation on the data (Section 3.1.2). The polar coordinates R, Θ are fitted into predefined genotype clusters with their centroid predicted by a neural network. R is a sum of normalized intensities (x_{norm} and y_{norm}, Section 3.1.2) and Θ is a product of the affine transformation defining the rotational angle during the normalization. The neural network was trained on bulk DNA from the Phase I HapMap Project (International HapMap Consortium, 2005) for three global populations with minor allele frequency (MAF) >0.05. The topology of the neural network is not known, as GenCall is a proprietary algorithm. The samples from HapMap form a centroid as the default position to which the genotype is inferred by determining the nearest cluster (Figure 4.4). A confidence score, known as the GenCall score (GC), is assigned to every SNP and is used as a measure of quality. As a default, genotypes with GC values lower than 0.15 are considered false positives. The user has an option to generate new cluster files ('recluster') using the user-derived data, however this step requires sufficient coverage of all three genotypes for every SNP – according to Illumina's documentation this is approximately 100 individuals per locus per population (Illumina, 2014). Illumina's algorithm and its default configuration are optimized for bulk DNA.

GenoSNP

GenoSNP (Giannoulatou et al., 2008) is using quantile normalization (Section 3.1.2) to normalize the raw intensities between the red and green channels. The normalized values for each SNP i are then converted to log scale using following formula:

$$\mathrm{x_{i,log}} = \log_2(x_{i,norm} + 1) \quad (4.1)$$

$$\mathrm{y_{i,log}} = \log_2(y_{i,norm} + 1) \quad (4.2)$$

Where $x_{i,norm}$ and $y_{i,norm}$ are output of quantile normalization for SNP i. A four component mixture of t-distribution is fitted to the data. Three components correspond to the three genotypes and the forth component is to capture outliers. GenoSNP is using two different algorithms to infer the parameters of the mixture model: Expectation-Maximization and Variational Bayes (Section 4.3). GenoSNP does not use any reference population to evaluate the data – the analysis is purely based on a within sample strategy. GenoSNP gives posterior of a called genotype as a measure of confidence.

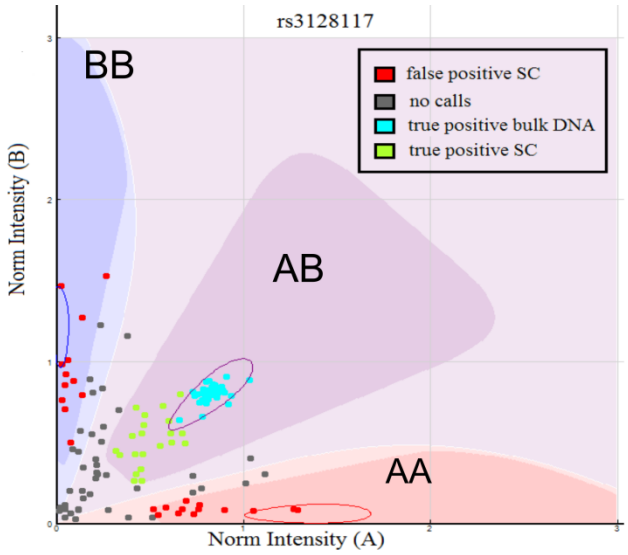

Figure 4.4. Visual output of GenomeStudio that is implementing the GenCall algorithm. Three genotype clusters inferred from HapMap are apparent with their elliptically shaped centroids. The points represent genotypes for a particular SNP locus rs3128117. False positives single cell calls are apparent as they fall into wrong cluster due to suboptimal signal intensities.

Illuminus

Illuminus (Teo et al., 2007) is using normalization from GenCall and furthermore converts the normalized intensities to strength s and contrast c defined as follows:

$$s = log(x_{i,norm} + y_{i,norm}) \quad (4.3)$$

$$c = \frac{x_{i,norm} - y_{i,norm}}{x_{i,norm} + y_{i,norm}} \quad (4.4)$$

Where $x_{i,norm}$ and $y_{i,norm}$ are output of affine transformation for SNP i (Section 3.1.2). Illuminus is using three-component mixture of multivariate truncated t distributions where the components correspond to genotypes AA,BB and AB. An Expectation-Maximization procedure is used to fit the parameters of the model. Similarly to GenoSNP, Illuminus does not use reference population to asses the final score of the genotypes.

CRLMM

In the program CRLMM, the raw intensities are first quantile normalized (Section 3.1.3) using distributions from training samples from HapMap. Subsequently, log ratios (M) of intensities and average log intensities (A) are calculated per SNP basis (MA transformation, Section 3.1.4). A three

component model is fitted to the data and furthermore, a spline curve is approximated for every component. The bias is corrected according to the fitted spline (Carvalho et al. 2007). Subsequently, a hierarchical model is created per SNP basis to account for potential probe effects. Similarly to the bias correction step, there is one Gaussian component for every genotype. The mean and standard deviation of a particular SNP are derived from the training HapMap samples.

4.7 Evaluation of performance

Evaluation of the computational models used in genomics, particularly in genotyping is discussed from two perspectives: machine learning theory and genotyping. While genotyping is a multiclass problem defined in Section 4.1, the evaluation evalutes the results as a binary classifier – either is the genotype correctly genotyped or not.

4.7.1 General evaluation of a classifier

We assume a binary classifier. The training examples from $\mathcal{D}$ are labeled as positive (P; $y_i = 1$)or negative (N; $y_i = 0$). In context of genotyping, the SNP that has been correctly resolved belongs to the positive class and the SNP that has been mistyped belongs to the negative class. The main class in the context of genotyping is P. If classified on labeled data, we can come across the following cases:

- True positives (TP) – positive examples that were correctly classified as such
- True negatives (TN) – negative examples that were correctly classified as negative
- False positives (FP) – negative examples that were incorrectly classified as positive
- False negatives (FN) – positive examples that were incorrectly classified as negative

The relationship between predicted data and the labels can be also represented using a confusion matrix (Table 4.1.)

Table 4.1. Confusion matrix.

		Prediction		
		Yes	**No**	**Total**
Actual class	**Yes**	TP	FN	P
	No	FP	TN	N
	Total	P‘	N‘	P + N

In the next chapters, we operate with following performance metrics:

- Recall, sensitivity or true positive rate (TPR): $\frac{TP}{TP+FN} = \frac{TP}{P}$
- Precision: $\frac{TP}{TP+FP}$
- Specificity: $\frac{TN}{N}$
- Accuracy: $\frac{TP+TN}{P+N}$

- False positive rate (FPR): $\frac{FP}{N} = \frac{FP}{TN+FP}$ = 1 - Specificity

Precision and recall can be combined into a single measure called $F_\beta - score$:

$$F_\beta = \frac{(1+\beta^2) \times precision \times recall}{\beta^2 \times precision \times recall} \quad (4.5)$$

If $\beta = 1$ then it is harmonic mean of precision and recall, noted as $F_1 - score. Beta$ decides about the weight of precision and recall, other, commonly used beta values are F_2 (where recall is weighted twice as much as precision) and $F_{0.5}$ (where precision is weighted twice as much as recall).

4.7.2 Graphical representation of performance

Visual representation of the performance is useful for comparison of multiple classifiers. The most common method is Receiver operating characteristics (ROC) curve that displays trade-off between TPR and FPR. To plot an ROC curve, the classifier needs to output the probability for the predicted class. Applying various thresholds, one can calculate TPR and FPR for a particular threshold and display it as a point in the XY scatter plot. Interpolation of these points gives rise to ROC curve. The area under the curve is another measure of the quality of the classifier. The larger the area under the curve, the better is the performance of the classifier – it is able to pick up most of the positive samples by minimizing the rate of incorrectly classified negative samples as positive (FPs).

The other type of an evaluation curve is Precision-Recall curve. It displays the trade-off beween precision and recall or sensitivity. Precision-recall curve is more informative than widely used ROC curve in case of dealing with imbalanced data (Saito et al., 2017). This is illustrated in Fig. 4.6. While ROC curve (Figure 4.5A) shows exactly same characteristics for both balanced and imbalanced dataset with 1000 positive examples, precision-recall curve is sensitive to the amount of negative examples (Figure 4.5B). This is due to the fact that a negative example is much more likely to be misclassified as the number of negative examples increases (Figure 4.5B), which affects precision.

4.7.3 Specific evaluation of genotyping

There are metrics specific to the area of genotyping. Some of them either overlap or are identical with the metrics commonly used in the machine learning theory.

Call rate

Called genotype is a genotype that met the quality criteria (i.e. defined threshold) of a particular genotyping algorithm. In machine learning theory this is a positive prediction, that corresponds to the sum of TP and FP. No-call (NC) is a genotype that failed the quality control. In terms of classification, this is reffered to as negative prediction, which corresponds to the sum of TN and FN.

Allele drop-in

Allele drop-in (ADI) is an erroneous heterozygous call. This means, that a homozygous allele (AA or BB) was incorrectly classified as heterozygous allele (AB). This type of error is a FP.

Allele drop-out

Allele drop-out (ADO) is an erroneous homozygous call. A heterozygous allele (AB) was incorrectly classified as homozygous allele.. This type of error is similarly to ADI considered as FP.

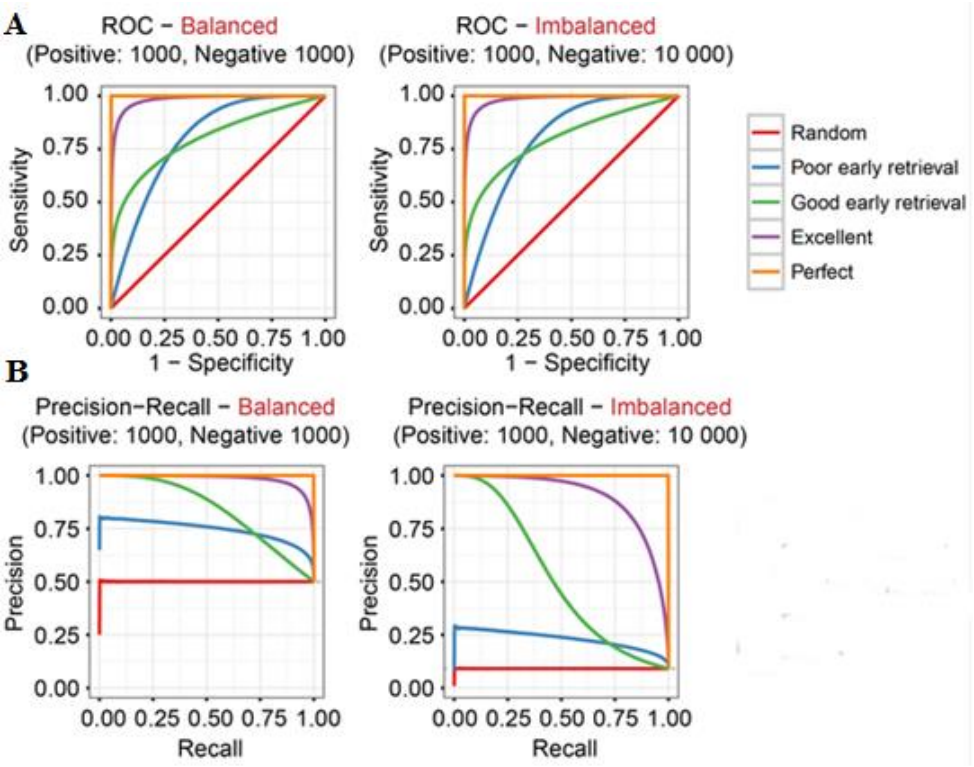

Figure 4.5. Illustration of ROC curve (A) and Precision-Recall curve (B) on balanced and imbalanced dataset[6]

4.8 Single-cell genotyping

As shown in Section 4.6, there is couple of tools available for genotyping of SNP array data. To the best of our knowledge, there is not a specialized algorithm available for the single cell genotyping from SNP arrays. Zamani Esteki et al. (2015) performed evaluation of two genotyping algorithms (GenoSNP and GenCall) in the single cell environment and attempted to adjust them to single cell data. An important summary of this is shown in Figure 4.6, where. Zamani Esteki et al. analysed the homozygous and heterozygous genotypes separately. As both methods give a score or measure of confidence of a genotype, they were systematically increasing the thresholds to acchieve better accuracy. The results suggest that while GenoSNP has generally higher call rate, it also suffers from

[6] adapted from https://classeval.wordpress.com/simulation-analysis/roc-and-precision-recall-with-imbalanced-datasets/

lower accuracy compared to GenCall, particularly for the heterozygous calls. The authors of the study decided to proceed with GenCall due to better overall properties. Adjusting the threshold of the GenCall algorithm to the single cell envorinment came at cost of significant data loss (call rate ~60 % at accuracy below 90 % , Figure 4.6). To give a frame of reference, validation studies on bulk DNA indicate both accuracy and call rate above 99% on average for all genotyping algorithms presented in this chapter (Ritchie et al., 2011).

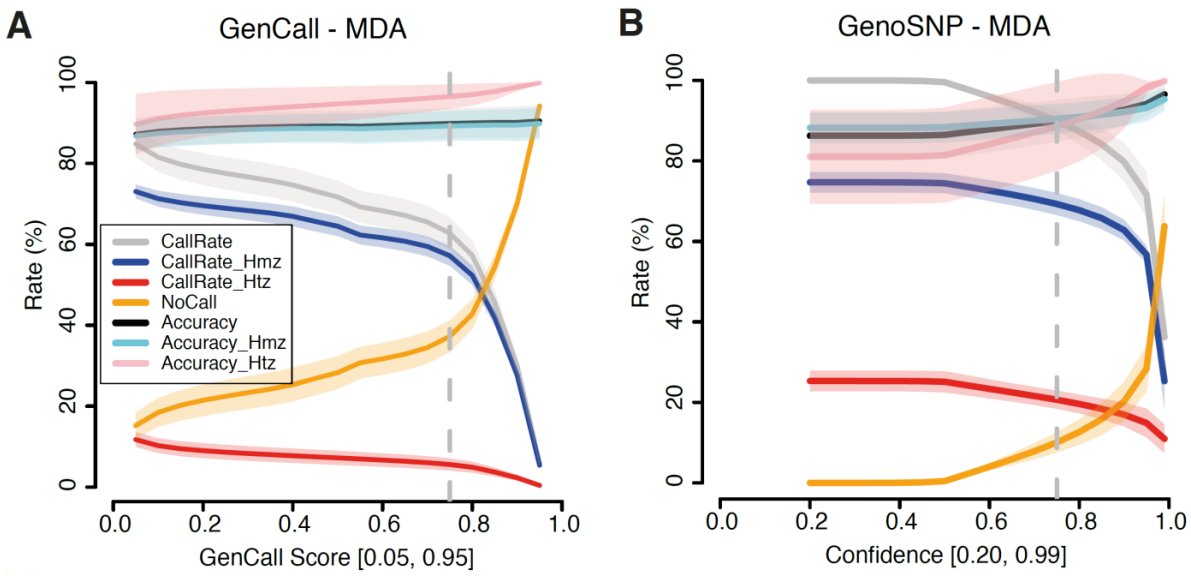

Figure 4.6. Comparison of performance of GenCall (A) and GenoSNP (B) on the single cell data amplified by MDA. The dashed vertical line shows the actual threshold that was selected by Zamani Esteki et al. for the genotyping as tradeoff of accuraccy and call rate. Hmz means homozygous, Htz is heterozygous.

4.9 Conclusion

In this chapter, we summarized the existing genotyping algorithms for Illumina SNP arrays and their underlying models (Sections 4.2, 4.3 and 4.6). We defined evaluation procedures (Section 4.7) and highlighted the suboptimal performance of the standard algorithms on single cell data compared to bulk DNA (Section 4.8).

5 Novel algorithm for noise filtration

5.1 Introduction

This chapter describes original algorithm that removes noisy genotypes from single-cell data caused by whole genome amplification. It largerly presents results from the published study by Vogel et al. (2019). To understand the noise, it is crucial to define a reference population of single cells where we know the ground truth. First, the creation of the reference dataset of single cell population is described, followed by data preprocessing and description of the prototype, workflow and implementation. As we work with biological data, it is important to reflect the biological and computational perspective on the data in the terminology:

- term `sample` for a biological entity – this is usually one cell from the dataset;
- term `training example` (or `testing example`) wherever we refer to the input of the algorithm (usually a single SNP) and it is consistent with the definition of $\mathcal{D}$ in Chapter 4.
- term `dataset` refers to collections of samples

To remind the reader, we operate with two categories of biological data. **Bulk genomic DNA (gDNA)** is characterized by sufficient amount of genetic material (pool), has great support in genotyping tools (Section 4.6) and gives very precise genotype estimation. On contrary, **single cell DNA (scDNA)** is characterized by very small amounts of DNA (from one cell) and needs to undergo a single cell path of the workflow for processing genomic data (Figure 2.5). This protocol causes deterioration of the signal and random erroneous genotypes at the output. The bottleneck of the workflow is illustrated in Figure 5.1.

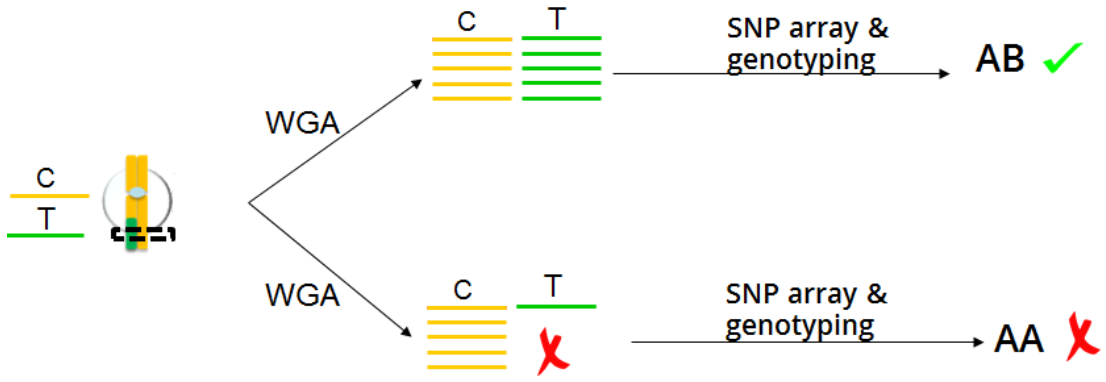

Figure 5.1. WGA – cause of errors in the single cell genotyping. The heterozygous locus in the dashed rectangle is correctly amplified with nearly equal amounts of both of the alleles (top) and then correctly genotyped as AB. The bottom branch illustrates erroneous whole genome amplification, where one of the alleles has suboptimal signal. This causes wrong detection of homozygous genotype.

5.2 Datasets of biological data

We operated with two human cell lines, GM12878 and GM7228 that we obtained from the NIGMS Human Genetic Cell Repository at the Coriell Institute for Medical Research, New Jersey, USA. Additionally, we obtained bulk DNA samples for GM7224 and GM7225 that are parents of GM7228 (Table 5.1). We used standard single-cell processing protocol with MDA (Section 2.2.2; SureMDA, Illumina Inc., California, USA) and Infinium Karyomapping Assay Kit (Illumina Inc., California, USA). We used GenomeStudio v2.0.2 software with Genotyping Module v1.9 (Illumina Inc., California, USA) for genotype calling. The details of the laboratory protocol can be found in Vogel et al. (2019).

Table 5.1 Overview of the SNP array data

Inidividual	Types	# of samples
GM7228	{gDNA; scDNA}	{9, 58}
GM7224	{gDNA}	{5]
GM7225	{gDNA}	{5}
GM12878	{gDNA;scDNA}	{5,45}

5.2.1 Groundtruth genotype from gDNA

We used parental information and the consensus approach to create a high confidence genotype for individual GM7228 and consensus approach and sequencing information to create high confidence genotype for GM12878. The strategy for both is illustrated in Figure 5.2.

In order to exclude potential miscalls, only genotypes matching the parental inheritance pattern were accepted for gDNA (Table 5.2). Furthermore, as multiple samples per individual were available, only genotype calls that were 100 % concordant were accepted. There was no family information for the individual GM12878 in the SNP array data. However, GM12878 is part of the Platinum Genome Sequencing Projects (Eberle et al., 2017) and therefore, high confidence sequencing data were available for this individual. Due to filtering of variants that were not concordant, did not follow the inheritance pattern or did not match the sequencing reference, we excluded 2.1 % of the autosomal SNPs from GM12878 and 4.2 % of the autosomal SNPs from GM07228. These numbers of retained SNPs are high and confirm that, although stringent criteria applied for the filtration, a large number of variants from the bulk gDNA are of a high quality.

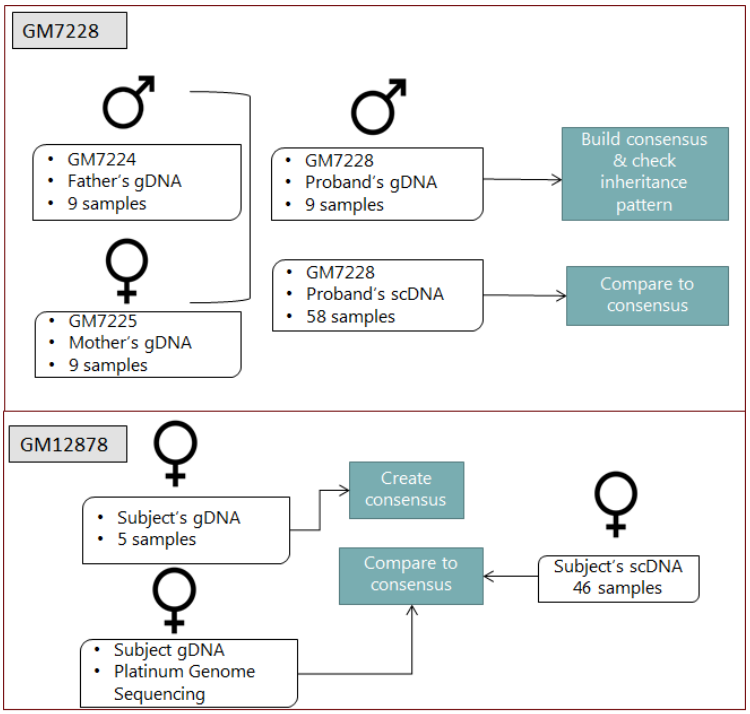

Figure 5.2. Strategy for creating high confidence groundtruth genotype and comparison to scDNA to create reference population of single-cell data. For the individual GM7228 there were parental genomes available (GM7224 and GM7225). For the individual GM12878, there were results of deep next-generation sequencing analysis (Platinum Genome Sequencing, Eberle et al., 2017) available.

Table 5.2 Rules for evaluating the correct genotypes based on parental information

mother (GM07224)	father (GM07225)	proband (GM07228)
AB	AB	{AB,AA,BB}
AA	AA	{AA}
BB	BB	{BB}
AB	BB	{AB,BB}
AB	AA	{AA,AB}
AA	BB	{AB}

5.2.2 Single cell genotyping with standard Illumina algorithm

Reliable high confidence genotypes (`reference`) described in the previous subsection allowed us to compare the single cell genotypes to the reference (Figure 5.2) and evalute the accuracy of the genotyping of single cell data cell by cell (more than 28.7 million SNP genotypes). We used two strategies to check the quality:

- minimal filtration using GenCall score 0.01 (QC001) that passes virtually all variants (including the strongly biased ones) and allows us to see the full error pattern in the data, and
- standard filtration using GenCall score 0.15 (QC015) that is recommended by Illumina.

Table 5.3 gives a detailed overview of the results. Homozygous genotypes (AA, BB) and heterozygous genotypes (AB) were analyzed separately and then together (`all`). Using the standard

QC cutoff (QC015), 73% SNPs (20.9 million) were correctly genotyped according to the reference, and 8% SNPs (2.36 million) were false positives. 19% SNPs were rejected, having failed to fall

Table 5.3. Detailed overview of the analysed single cell datasets

		QC001[a]											
Data set	**Region**	**+ (M)**	**+ (M %)**	**+ (SD)**	**+ (SD %)**	**- (M)**	**- (M%)**	**- (SD)**	**- (SD%)**	**NC (M)**	**NC (M%)**	**NC (SD)**	**NC (SD%)**
7228	AB	43,226	7.8	7,633	1.4	1,949	0.4	1,583	0.3	27,389	5	3,586	0.65
	AA,BB	174,821	31.7	4,424	0.8	28,333	5.1	6,971	1.3				
	all	218,048	39.5	11,047	2	30,282	5.5	7,970	1.4				
12878	AB	39,897	7.2	8,799	1.6	1,990	0.4	1,638	0.3	21,364	3.9	3,835	0.7
	AA,BB	183,543	33.2	4,585	0.8	29,732	5.4	8,139	1.5				
	all	223,441	40.5	12,823	2.3	31,721	5.7	9,427	1.7				
	Total	22,925,096 (79.8%)				3,215,551 (11.2%)				2,571,343 (9%)			
		QC015[b]											
Data set	**Region**	**+ [a] (M)**	**+ (M %)**	**+ (SD)**	**+ (SD%)**	**- (M)**	**- (M %)**	**- (SD)**	**- (SD %)**	**NC (M) [e]**	**NC (M%)**	**NC (SD)**	**NC (SD%)**
7228	AB	30,382	5.5	7,038	1.3	666	0.1	564	0.1	54,500	9.9	7,432	1.4
	AA,BB	168,863	30.6	7,029	1.3	21,309	3.9	6,314	1.1				
	all	199,245	36.1	13,087	2.4	21,975	4	6,634	1.2				
12878	AB	27,357	5	8,075	1.5	715	0.1	665	0.1	49,343	8.9	8,394	1.5
	AA,BB	176,238	31.9	8,102	1.5	22,874	4.1	7,355	1.3				
	all	203,595	36.9	15,730	2.9	23,589	4.3	7,831	1.4				
	Total	20,921,603 (72.9%)				2,359,645 (8.2%)				5,430,742 (18.9%)			

[a]+ are true positives (correctly classified by GenCall algorithm following specified threshold), - are false positives (misclassified by the GenCall algorithm); M is mean, SD is standard deviation across all cells in the particular dataset

within the genotype clusters defined by bulk DNA genotypes from HapMap (illustration in Figure 4.4). To capture the variability between the cells, we also displayed the data in per cell format with mean value (M) and standard deviation (SD, Table 5.3). The true positive rate was higher when we used a minimal QC (0.01) compared to the standard QC of GenCall (40%, SD=2% and 36%, SD=2%, respectively, for cell line GM07228 and 40%, SD=2% to 37%, SD=3% for GM12878. These differences in true positive rates are statistically significant ($p<0.0001$, Figure 5.3). In total for both datasets, the GenCall algorithm rejects about 7% of correctly genotyped SNPs from WGA DNA and increases precision by 3%.

5.3 Structure of noise in single cell

Systematic comparison of the scDNA with the reference reveals presence of noise (Table 5.3). To test whether the noise creates a distinct pattern, we performed following steps:

1. We randomly selected 10,000 SNPs from the single cell data GM07228 and extracted the normalized (affine-transformed) itensities.

2. We further transformed the intensities using MA transformation (Section 3.1.4, Figure 5.4A).
3. We estimated density function separately for erroneous calls and correct calls on the M and A values using bivariate normal kernel (Section 4.4)
4. The results (Figure 5.4) indicate that, as expected, the correctly genotyped SNPs build three clusters corresponding to AA, BB and AB genotypes. The noisy data also builds three clusters correspondingto the transition between AB and AA or AB and BB (two clusters of allele drop-outs; ADO) and one cluster with allele drop-ins (ADI). The data suggests good separability of the errorneous clusters from the correct clusters since the centers of the clusters are non-overlapping.

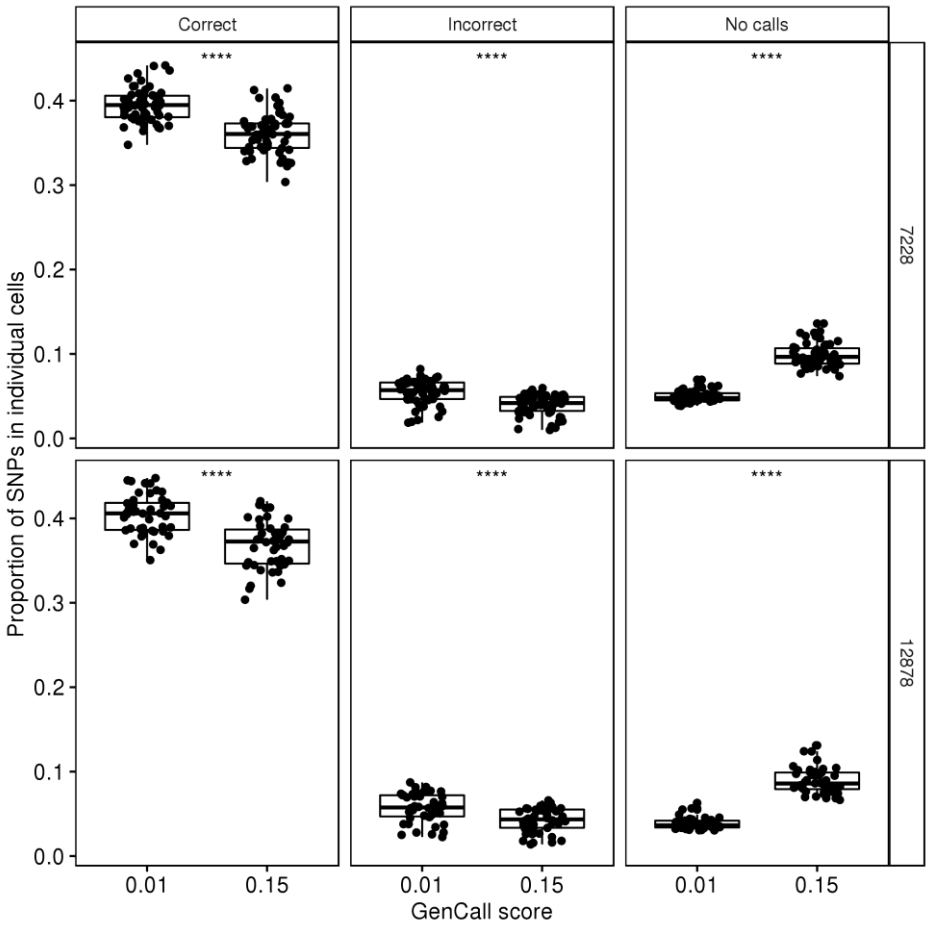

Figure 5.3. Rates of correct calls, incorrect calls and no-calls for 58 individual cells of GM7228 and 46 cells of GM12878. Each dot represents one cell. The proportions of SNPs within each single cell, that were correctly typed, incorrectly typed, or not typed (rejected due low GenCall score) are shown in the plots from left to right. Paired t-test was used to draw differences between cells typed with GenCall score 0.01 and GenCall score 0.15. The number of asterisks above the graps corresponds to level of significance of the difference (**** corresponds to p-value<0.0001)

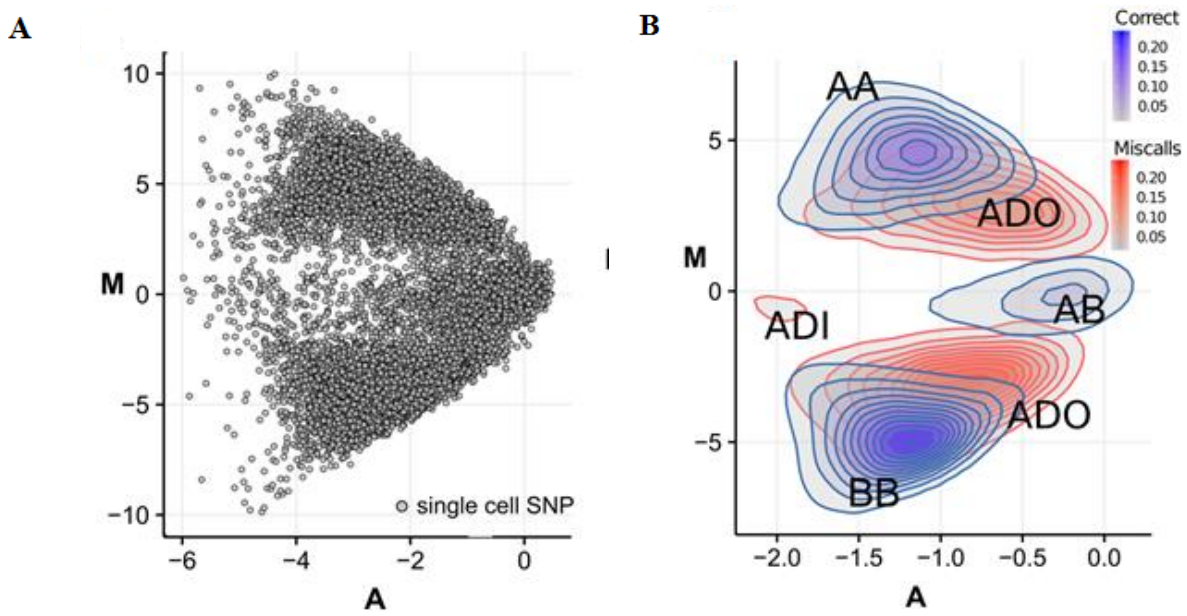

Figure 5.4. MA plot of 10,000 randomly selected SNPs from single cell data. Every dot corresponds to one variant from single cell (A). 2D density function with bivariate normal kernel was applied on the data to reveal the erroneous clusters (B). Red clusters correspond to mistyped SNPs (noise), blue clusters show true signal. ADO clusters mark erroneous transitions from heterozygous to homozygous genotype and ADI cluster marks erroneous transition from homozygous to heterozygous genotype.

5.4 Training dataset

5.4.1 Feature transformation

The MA transformation in Figure 5.4 illustrates feasible input features for training. As partly discussed in Section 3.1.4, these features contribute to generalization of the problem and decrease the intragroup variability. We confirm this on fraction of our single cell data by systematically analysing statistical properties of raw intensities without any normalization, intensities normalized with affine transformation and intensities normalized with MA transformation (Figure 5.5). It is aparent, that lacking normalization shows great variability between the samples (Figure 5.5A). The affine transformation partly brings the data into comparable scale, however the variability is still relatively high (Figure 5.5B). Finally, the MA transformation on the top of the affine transformation displays the data from multiple samples in a common scale, descreases the variability and allows to merge SNPs from various single cell samples into one bigger dataset (Figure 5.5C). The effect of MA transformation can be explained as follows: Samples and the red and green channels analyzed in different labs or on different chips accumulate certain bias (other than bias from the whole genome amplification). Instead of analysing the signals independently (displaying and analysing red and green signal separately), the MA transformation cancels out the lab and sample specific bias and allows to display the true signal in a logarithmic scale.

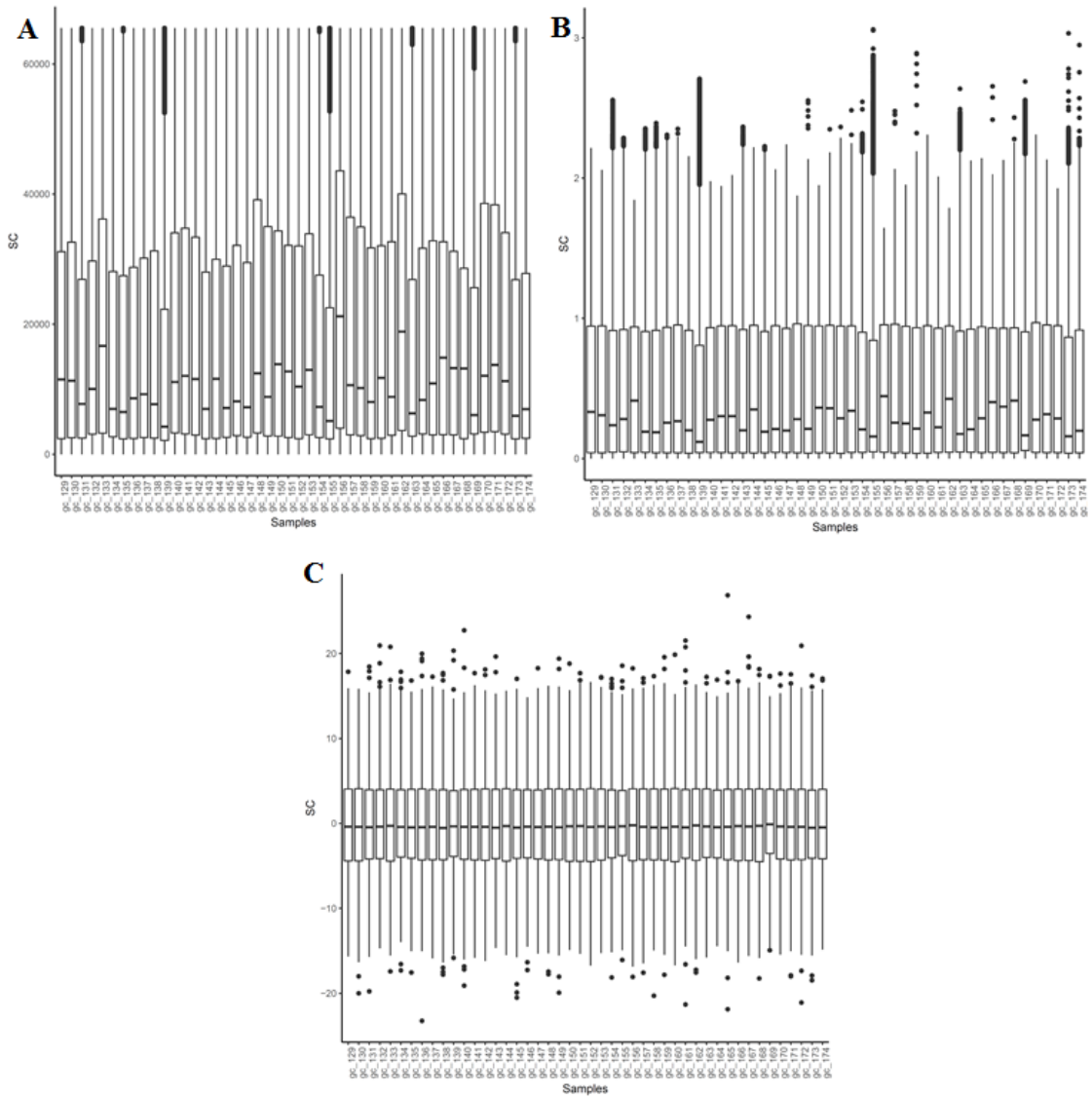

Figure 5.5. Boxplots of single cell intensities without any normalization (A), X channel after affine transformation (B) and M value after MA transformation of X and Y channel (C).

5.4.2 Feature selection

We compared our single cell datasets to the reference genotype. More specifically, for every candidate single-cell call for SNP i and sample s we assigned a label $l_{i,s}$: $l_{i,s} \in \{True, False\}$, depending on the match or mismatch with the corresponding reference genotype call. The training dataset is then a set of of triplets $(m_{i,s}, a_{i,s}, l_{i,s})$, where $(m_{i,s}, a_{i,s})$ are input features and $l_{i,s}$ is the output feature. Note that we omit sample index s in further explanation, as we do not distinguish between the origins of SNPs in the training data set. We included all autosomal single cell calls with GenCall score above 0.01 (QC001) totaling 14,403,139 SNPs for training (GM07228) and 11,737,508 SNPs for validation (GM12878). Lowering the GenCall score threshold for accepting a SNP allowed us to include potentially poorly amplified SNPs and to capture the full error pattern.

5.4.3 Problem of imbalanced dataset

Table 5.3 suggests that the training data is highly imbalanced with positive class being the majority class. The positive class is our target class, and the analysed single cell datasets reflect the `real world` class ratio. Neverthless, as we operated with sufficient amount of data we performed downsampling of the positive datasets to obtain same amount of positive and negative samples. We discuss this issue and perform further experiments in Chapter 6.

5.5 SureTypeSC

SureTypeSC is a novel machine learning method that combines nonparametric (in terms of statistical distribution) supervised method embodied in Random Forest (RF) that adapts to the noise in the single cell. RF is trained on the reference dataset described in Section 5.2. The fitted RF then estimates the regions of noise and high quality SNPs in unseen data. These regions are then formalized using a parametric model. This is a second layer of the algorithm and consists of a system of Gaussian mixtures called Gaussian discriminant analysis.

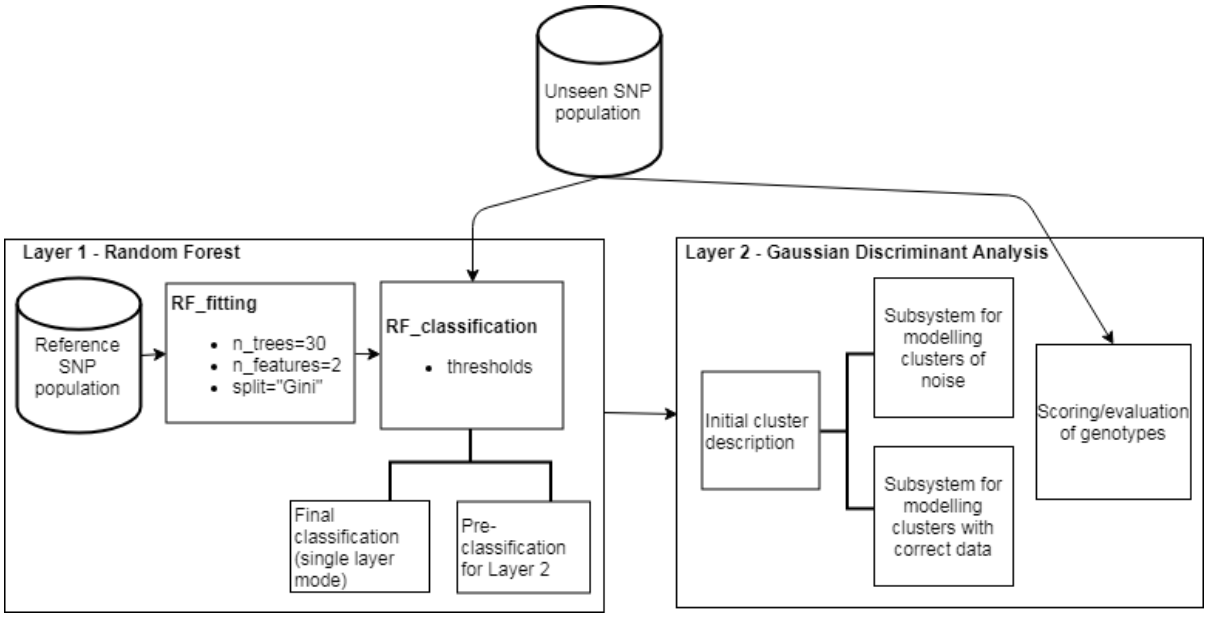

Figure 5.6. Scheme of prototype for SureTypeSC based on machine learning with two layers organized in cascade.

The prototype of SureTypeSC is schemed in Figure 5.6. Classification of unsen SNP data is first performed with Layer 1 (RF trained on reference data). RF in the single layer mode directly classifies the unseen data. In the standard, two layered mode, Random Forest estimates the positive and negative classes for the further improvement with Layer 2. Layer 2 accepts the classification from Layer 1 and creates an initial cluster estimate, which is then iteratively refined. Subsequently, the genotypes from the unseen SNP population are evaluated again with the fitted model from Layer 2. Figure 5.6 also illustrates parameters of the model, that can be subsequently subjected to experiments.

This is i.e. threshold for the positive class defined for RF. This parameter influences the input of the Layer 2 and therefore the output of the final classification. Experiments with this and other parameters of the model are described and discussed in in Chapter 6. The following two subsections discuss both layers of the algorithm in a greater detail.

5.5.1 Basic Random Forest layer

We chose Random Forest classifier for the initial training and classification. The kernel density estimation in Section 5.3 suggests that the function that separates the erroneous clusters from the clusters of correct data (red and blue contours in Figure 5.4B) is non-linear. The ensemble nature of Random Forest has by definition the ability to fit different trees to different parts of the input space and therefore mimic a non-linear separating function that, in theory, can increase the classification accuracy. We implemented the Random Forest algorithm in scikit package (Pedregosa et al., 2011) and for the basic implementation, we adjusted following parameters of the Random Forest algorithm:

- the number of trees was increased from 10 to 30; according to experiments in Oshiro et al. (2012), a theoretical upper limit is 128 trees and further increase in number of trees does not contribute to higher accuracy. However, our data suggest that forests with more than 30 trees contribute minimally to the accuracy of the model but increase the size of the model substantially; this is discussed and supported by experiments with the model data in Chapter 6.
- the number of features to consider when looking for the best split was set to two (maximum)

5.5.2 Refinement using Gaussian Discriminant Analysis

The second layer of the algorithm is a Gaussian Discriminant Analysis (GDA) that formalizes the genotype clusters obtained from the RF step and potentially improves the classification. GDA models positive and negative class separately using GMM and defines a scoring function that discriminates the data based on their affinity to positive and negative class. The general concept of GDA was adapted from Ng (2019) and Hastie et al. (2016).

Let $D = \{x_j | j = 1 \dots N, x_j \in \mathbb{R} \times \mathbb{R} \times \mathbb{G} \times \widehat{\mathbb{L}}\}$ denote a set of N SNPs that were classified by the trained RF, where $\mathbb{G} = \{\text{AA}, \text{AB}, \text{BB}\}$, $\widehat{\mathbb{L}} = \{\text{T}, \text{F}\}$. Therefore, $x_j = (m_j, a_j, g_j, \widehat{l_j})$ is a quadruplet of the logarithmic difference, logarithmic average, genotype predicted by GenCall (QC 0.01) and class prediction by RF at the *j*-th SNP. We assume that both the positive (T) and negative (F) classes, which are represented by pairs $d_i = (m_i, a_i)$, are drawn from mixtures of multivariate normal distributions and define the following system of Gaussian discriminants:

$$\hat{L} \sim Bernoulli(\lambda) \tag{5.1}$$

$$p(\hat{l}) = \lambda^{\hat{l}}(1-\lambda)^{1-\hat{l}} \tag{5.2}$$

$$p(d_j | \hat{l} = T) = p(d_j | \Theta_{\text{T}}) = \sum_{k=1}^{3} \alpha_{T,k} \phi(d_j | z_{T,k}, \theta_{T,k}) \tag{5.3}$$

$$p(d_j|\hat{l} = F) = p(d_j|\Theta_F) = \sum_{k=1}^{3} \alpha_{F,k}\phi(d_j|z_{F,k}, \theta_{F,k}) \quad (5.4)$$

Where:

- λ denotes probability $p(\hat{l} = T|d_j)$
- ϕ is multivariate normal density function with parameters θ_k (with mean μ_k and covariance matrix Σ_k)
- z_k is an indicator variable that denotes the genotype class, where $z_k \in \mathbb{G} \times \widehat{\mathbb{L}}$
- α_k is the mixture component weight representing the probability that a random tuple (m_j, a_j) was generated by component *k*.

The complete set of parameters for the presented Gaussian discriminants is given as $\Theta_{\widehat{\mathbb{L}} \in \{T,F\}} = \{\alpha_{\widehat{\mathbb{L}},1}, \ldots \alpha_{\widehat{\mathbb{L}},3}, \theta_{\widehat{\mathbb{L}},1} \ldots \theta_{\widehat{\mathbb{L}},3}\}$. Decomposing the definition of z_k, we obtain following list of all components lying in two mixture models:

- a cluster of true heterozygous SNPs (AB_{TRUE})
- a cluster of false heterozygous SNPs (AB_{FALSE})
- a cluster of true homozygous SNPs (AA_{TRUE})
- a cluster of false homozygous SNPs (AA_{FALSE})
- a cluster of true homozygous SNPs (BB_{TRUE})
- a cluster of false homozygous SNPs (BB_{FALSE})

We use maximum likelihood estimation to estimate the parameters Θ_L. The log-likelihood function $\mathcal{F}$ for classes from $\widehat{\mathbb{L}}$ is defined as follows:

$$\ln \mathcal{F}(\Theta) = \sum_{j=1}^{N} \ln p(d_j|\Theta_{\hat{L}}) \quad (5.5)$$

We use an Expectation Maximization algorithm (Dempster et al., 1977) to find optimal solution for $\Theta_{\hat{L}}$ of the positive and negative class that maximize their log-likelihood function (Eq. 5.5). As shown in Section 4.3.3, the EM algorithm is divided into an Expectation-Step (E-Step) and a Maximization-Step (M-Step). These are run in iterations separately for the positive and negative classes until convergence is reached. $N_{\hat{l}}$ is total number of SNPs in the particular class.

EM algorithm for the cluster and class refinement:

For every SNP i with label $\hat{l}$ in $\widehat{\mathbb{L}}$:

1. E-step - calculate membership weights - probabilities of (m_i, a_i) belonging to a cluster k using either initialization parameters $\Theta_{\hat{l}}^{init}$ if this is the first iteration, otherwise $\Theta_{\hat{l}}^{t+1}$

$$\omega_{i,k,\hat{l}} = p\left(z_{i,k,\hat{l}} = 1 \middle| d_i, \Theta_{\hat{l}}^{t}\right) =$$
$$= \frac{\Phi\left(d_i \middle| z_{i,k,\hat{l}}\theta_{k,\hat{l}}\right) \cdot \alpha_{k,\hat{l}}}{\sum_{m=1}^{K} \Phi\left(d_i \middle| z_{i,m,\hat{l}}, \theta_{m,\hat{l}}\right) \cdot \alpha_{m,\hat{l}}} \quad \text{for } 1 \leq k \leq 3, 1 \leq i \leq N. \tag{5.6}$$

2. M-step
   - Calculate new component weights for the next iteration

$$\alpha_{k,\hat{l}}^{t+1} = \frac{\sum_{i=1}^{N_{\hat{l}}} \omega_{i,k,\hat{l}}}{N_{\hat{l}}} \tag{5.7}$$

   - calculate new means for the next iteration

$$\mu_{k,\hat{l}}^{t+1} = \frac{\sum_{i=1}^{N_{\hat{l}}} \omega_{i,k,\hat{l}} d_i}{\sum_{i=1}^{N_{\hat{l}}} \omega_{i,k,\hat{l}}} \tag{5.8}$$

   - calculate new covariances for the next iteration:

$$\Sigma_{k,\hat{l}}^{t+1} = \frac{\sum_{i=1}^{N_{\hat{l}}} w_{i,k,\hat{l}} \cdot (d_i - \mu_{k,\hat{l}}^{t+1})(d_i - \mu_{k,\hat{l}}^{t+1})^T}{N_{k,\hat{l}}} \tag{5.9}$$

   at the end of the M-step we obtain new parameter estimates Θ^{t+1}

3. Calculate log likelihood using Equation 5.5 and if the relative change in the overall likelihood is smaller than a threshold, halt. Otherwise proceed with the E-step with parameters from Θ^{t+1}.

After the parameters of both classes have been estimated by the EM algorithm, they are subjected to a second run. Here, the class membership $\hat{L}$ is hidden from the algorithm and every SNP *i* is evaluated for both Gaussian discriminants using the following formula:

$$\left(score_{i,T}, score_{i,F}\right) = [\ln p(d_i|\Theta_{\hat{T}}), \ln p(d_i|\Theta_{\hat{F}})] \tag{5.10}$$

The final classification (membership to a positive or a negative class) is determined by higher value from the pair $(\mathrm{score}_{\mathrm{i,T}}, score_{i,F})$.

5.5.3 Scoring function

The key role of a genotyping algorithm is to report the likelihood of a certain genotype in form of a score or a posterior probability. Besides GenCall having its own scoring scheme, we used the following equations to estimate the probability of a certain SNP being correctly genotyped:

1. Random Forest: the score of a genotype of the *i*th SNP is given as a proportion of the trees in the forest that voted for a particular genotype being correct:

$$score_{i,RF} = p(l_i = T|d_i) \tag{5.11}$$

2. The scoring strategy of SureTypeSC is inferred from its second layer (GDA) as the class-conditional posterior probability of a genotype falling into positive class T:

$$score_{i,RF-GDA} = \frac{e^{score_T} \times p(T)}{\sum_{Z\in\{T,F\}} e^{score_Z} \times p(Z)} \tag{5.12}$$

5.6 Implementation of the algorithm

To summarize the implementation of the methodology, we genotyped 58 single cells from GM07228 using standard Illumina genotyping workflow (GenomeStudio) and generated high confidence genotypes generated from the bulk DNA from the trio (paternal, maternal and son- GM07228; Section 5.2). We set up a parallel branch for testing 46 single cell genotypes (subject to Chapter 6) from cell line GM12878 (Section 5.2). The intensities were stored in the Genotype Call Files (*.gtc), the information about SNPs and probe content was stored in BeadPool Manifest file (*.bpm). The cluster files (*.egt) carry the reference information for each locus. All files were available prior to the single-cell genotyping analysis (the intensity files are the final product of the BeadChip scanning procedure and *.egt and *.bpm are available online from Illumina). We then exported the data from GenomeStudio, indexed using chromosomal position (chromosome and position) and SNP ID (id) in a dataframe structure from pandas library (McKinney, 2010) . The MA-transformed data were fitted to the two-layered machine learning model and validated using both independent and cross-validated datasets (Chapter 6). The machine learning core is represented by an original algorithm aimed towards noise removal in single cell data called SureTypeSC. The first layer is represented by Random Forest (Section 5.5.1), whereas the second layer is a system of Gaussian mixtures (Section 5.5.2) called Gaussian Discriminant analysis. The machine learning methods were implemented using package scikit (Pedregoza et al., 2011).

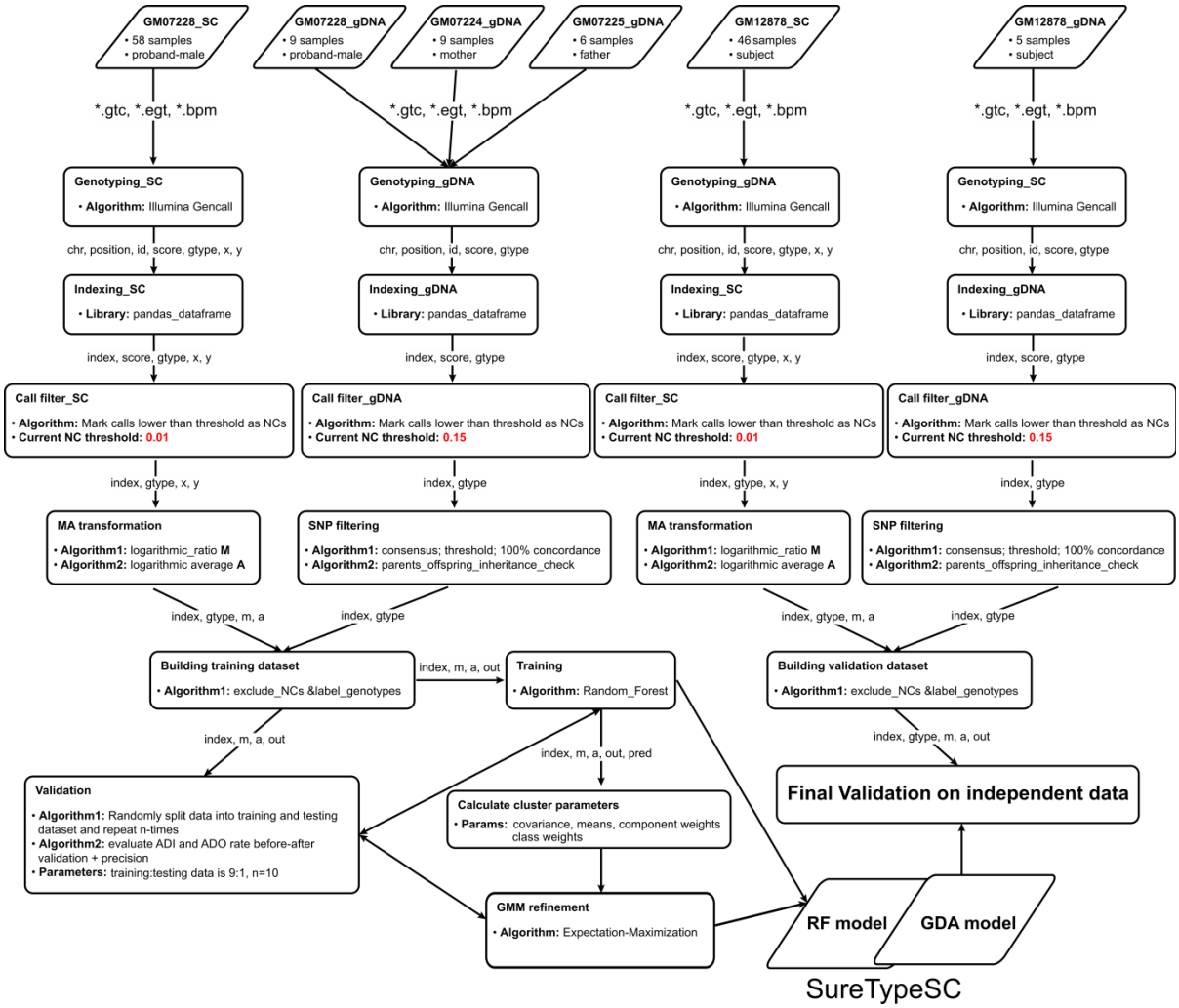

Figure 5.7. Flowchart for single cell analysis of the presented data, filtration and validation.

6 Validation and experiments with the model

This chapter presents the validation process for the proposed algorithm presented in Chapter 5. The validation is carried out in two ways – using cross-validation and on independent dataset that was created from a different individual than the training dataset. The results of the validation part (without experiments in Section 6.3) were published in Vogel et al. (2019).

We analyse the performance of SureTypeSC, but also its constitutive single layers. We therefore denote, consistently with the previous explanations, the single layers by their acronyms (RF and GDA) and the cascade solution by their combination (RF-GDA). To pinpoint the differences in performance and to show the improvements SureTypeSC achieves in the single cell domain, we included the performance of GenCall in all validation analyses. GenCall represents the current state of the art and has been used in multiple single-cell genotyping analyses (Ottolini et al., 2016; Zamani Esteki et al., 2015; Handyside et al., 2010). As we operate with multiple single-cells (46 or 58 cells; Section 5.2), we calculate mean values and standard deviation or confidence intervals to capture the variability of the results and perform statistical tests, whenever applicable, to show the significance of the measured differences in the performance. We always evaluate the heterozygous and homozygous genotypes separately. Having reliable method for heterozygous loci improves detection power of many knowledge extraction algorithms from single cell data (subject to Chapter 7 and 8).
We perform validation procedures from two categories (Section 4.7):

- procedures that capture the whole spectrum of classification outcomes by changing the classification threshold (visual representation and ROC-AUC score);
- validation procedures that operate with fixed classification thresholds; these procedures mimic the real applications where we have to decide for a particular cutoff to retrieve the information about rejecting or accepting a particular genotype; we used accuracy,precision,recall and F1-score (defined in Section 4.7)

We discuss the antagonistic relationship of precision and recall and how different thresholding strategies can be beneficial for achieving high recall or precision of the algorithm.

The second part of the chapter deals with different experiments with the model. As pointed out in Chapter 5, SureTypeSC includes parameters that can be adjusted and would theoretically influence the performance of the classification. Based on the results of the performed experiments, we will discuss whether these parameters can further improve the classification.

Table 6.1 summarizes validation strategies and metrics carried out in this chapter. The reader is refered to particular section for details.

Table 6.1. Overview of the validation strategies

Atribute/Strategy	Cross validation	Validation (standard genotyping)	Validation (high precision genotyping)	Experiments
Training	GM7228	GM7228	GM7228	GM7228
Testing	GM7228	GM12878	GM12878	GM12878
Parameters[a]	default thresholds	threshold Table 6.5	thresholds in Table 6.7 and Table 6.8	per experiment
Categories	het and homo	het and homo	het and homo	per experiment
Tested algorithms	GenCall, SureTypeSC	GenCall, SureTypeSC	GenCall, SureTypeSC	layers of SureTypeSC
Metrics	precision,recall,F1-score, accuracy, ROC-AUC score	precision,recall,F1-score, accuracy, ROC curve, ROC-AUC score, Precision-Recall curve	matrix of posterior probabilities, ADI, ADO and call rate	ROC-AUC score
Comparison	mean and CI across 10 folds	mean and CI across 46 cells, paired t-test	mean and CI across 46 cells	per experiment
Main objective	consistency check, comparison to the state of the art and between layers of SureTypeSC	comparison to the state of the art and between layers of SureTypeSC	maximize precision	test impact of various parameters on performance
Section	6.1	6.2	6.2.3-6.2.4	6.3

[a] by default, various threshold were tested, other parameters were experimented in the experimental section

6.1 Cross-validation

The cross-validation divides the input dataset into *m* parts, whereas *m-1* parts (folds) are used to build the model and the remaining part is used for testing. This partitioning is done *m* times, randomly.

The classification dataset for genotyping of single cell is always imbalanced – the target class is the correctly genotyped SNPs and is the majority class. On contrary, the mistyped SNPs represent the minority class. We therefore used stratification to ensure that every fold contains both correctly genotyped and mistyped SNPs with nearly equal ratio. We chose *m=10*. To tackle the imbalance problem, we always balanced the training fold by down-sampling the correctly genotyped SNPs. We evaluated the performance of every testing fold and scored the genotypes of all algorithms using the GenCall score (Section 4.6) or Eq. 5.11 and 5.12 for SureTypeSC (Section 5.5).

Table 6.2. Results of the cross-fold validation on dataset GM7228[a]

Genotype			homozygous		
Algorithm/Metrics[b]	**precision**	**recall**	**ROC-AUC**	**accuracy**	**F1-score**
GDA	0.93±0.001	0.86±0.009	0.82±0.003	0.82±0.007	0.89±0.005
RF	0.94±0.002	0.73±0.006	0.79±0.002	0.73±0.005	0.82±0.004
RF-GDA	0.93±0.001	0.86±0.008	0.82±0.004	0.82±0.006	0.89±0.004
GenCall	0.89±0.002	0.89±0.005	0.66±0.005	0.81±0.005	0.89±0.003
			heterozygous		
	precision	**recall**	**ROC-AUC**	**accuracy**	**F1-score**
GDA	0.99±0.001	0.87±0.006	0.89±0.009	0.86±0.005	0.92±0.003
RF	0.99±0.001	0.82±0.007	0.88±0.005	0.82±0.006	0.90±0.004
RF-GDA	0.99±0.001	0.87±0.004	0.89±0.006	0.86±0.003	0.92±0.002
GenCall	0.98±0.002	0.70±0.002	0.75±0.003	0.70±0.001	0.82±0.001

[a] values are mean proportions over 46 cells ± confidence interval
[b] score of 0.5 was used for GDA, RF, and RF-GDA and default (recommended) score 0.15 for GenCall

We performed the classification using fixed classification thresholds (0.5) and by Illumina recommended classification threshold for GenCall (0.15). Consistent with random sampling of the SNPs, the mean performances of all algorithms have narrow confidence intervals (at 95%), which suggests that the algorithms are invariant to SNP selection (Table 6.2). Greater variability between folds would mean that some SNPs are easier/harder to classify than the others. The results indicate that RF-GDA and GDA outperform GenCall in all metrics and for both, homozygous and heterozygous genotypes, except for the recall for homozygous genotypes. While ROC-AUC is, however, lower for homozygous GenCall then for the other algorithms, this could be likely an issue of suboptimal thresholding for SureTypeSC. Thresholds will be adjusted in the next section. It is also apparent that the parametric model (GDA) outperforms RF except for the precision for the homozygous calls, where RF achieves higher score. However, there is virtually no difference between GDA and the cascade model RF-GDA in this test.

6.2 Validation on independent dataset

In this performance analysis, we used the SNP genotypes obtained from 58 single cells from cell line GM07228 for training and the SNP genotypes obtained from 46 single cells from a different cell line, GM12878 (Table 5.3), for testing. The genotyping data from the testing set were obtained at an independent time, with different batches of WGA reactions and genotyping arrays. This avoids systematic errors introduced by the chemistry used to obtain the genotypes. We used both, metrics that describe the classifiers' performance at various cutoffs (`Validation curves and ROC-AUC score`), as well as metrics that statically describe the performance at particular score cutoff (`Static evaluation`). The performance between homozygous and heterozygous regions may vary - we therefore analysed them separately. In the last part of the section, we conclude how SureTypeSC

contributes to error reduction in terms of allele drop outs and allele drop ins. We summarize the datasets involved in training/testing and the scoring strategy of genotypes in Figure 6.1.

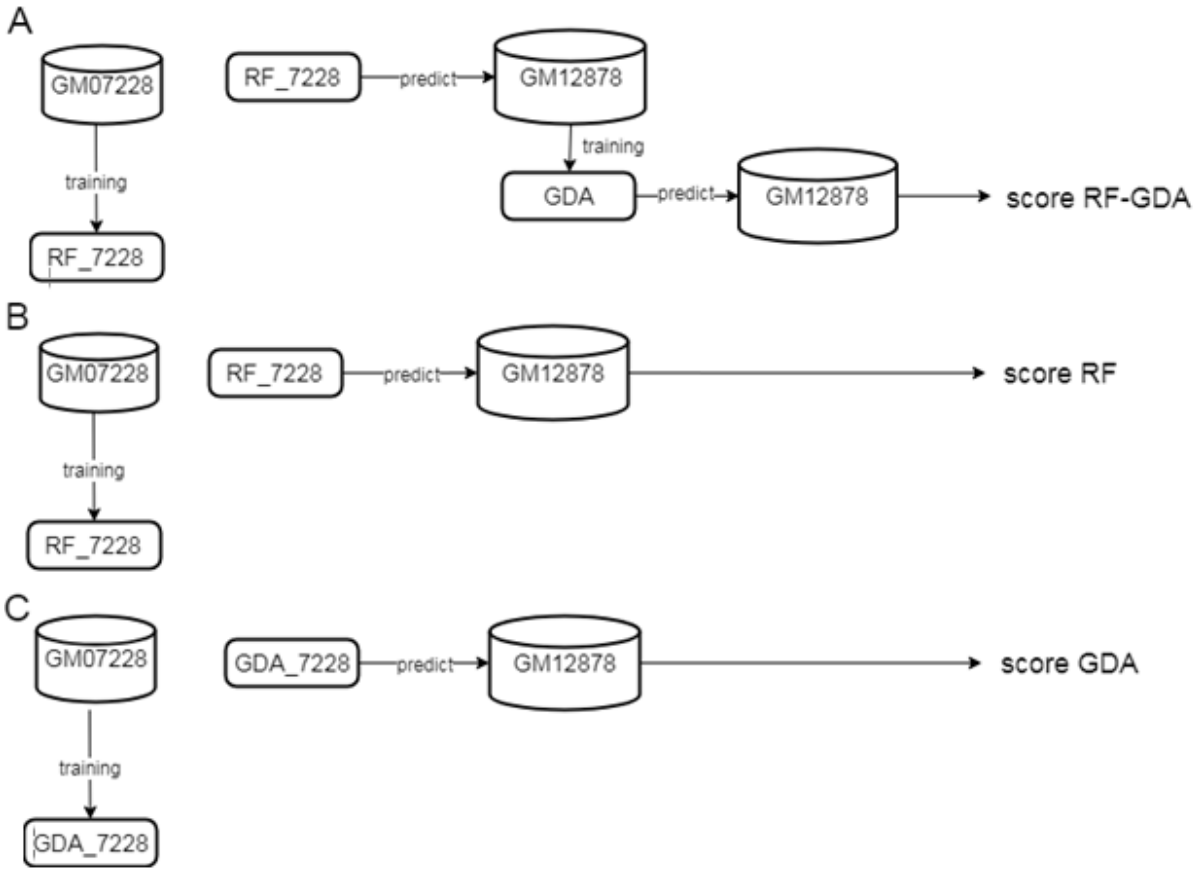

Figure 6.1. Training and testing strategy for SuretypeSC. (A) The RF was trained on ground truth data from GM07228 and used to predict the values of GM12878. The GDA was used to fit the predicted values of GM12878 (B) The RF was trained on the ground truth data from GM07228 and used for prediction and scoring on the testing data, GM12878. (C) The GDA trained on ground truth from GM07228 and prediction and scoring took place on the testing data, GM12878.

6.2.1 Validation curves and ROC-AUC score

We first demonstrate the performance of the algorithms using ROC and Precision Recall curves. These metrics gave us visual insight into overall performance of the classifiers, invariant to the score cutoffs used. For the heterozygous calls, RF-GDA outperforms all tested algorithms, which is also quantified by the ROC-AUC score (Figure 6.2, Figure 6.3 and Table 6.3). While GenCall achieves a 74% ROC-AUC score on average, this is increased to 86%, 87% and 92% for RF, GDA and RF-GDA, respectively (Table 6.3). The mean differences of the ROC-AUC scores are at $p < 0.001$ (Table 6.4). For the homozygous regions, the RF outperforms GenCall at all points of the ROC and Precision-Recall curves, which is supported by the increase in the ROC-AUC score from an average of 67% (GenCall) to 81% for the RF (Table 6.3 and Figure 6.2A). This is further increased with the GDA or RF-GDA (both 83%, Table 6.3). Interestingly, at a precision of approx. 93%, the RF curve crosses that of the GDA and RF-GDA and recalls more true positive homozygous calls (Figure 6.3). This suggests that the RF alone might be a good option if higher recall is required at the costs of lower precision, which is nevertheless higher than GenCall in the homozygous regions. GenCall

crosses the Precision-Recall curve of the RF-GDA at a precision around 88% and recalls more true positives (Figure 6.3A). This is, however, very close to a recall of 100%, which also means accepting all calls without any filtration.

Table 6.3 ROC-AUC score of the genotyping algorithms on independent dataset GM12878[a]

GenCall		RF		GDA		RF-GDA	
het	homo	het	homo	het	homo	het	homo
0.74±0.01	0.67±0.015	0.86 ± 0.004	0.81 ± 0.012	0.87 ± 0.005	0.83 ± 0.013	0.92 ± 0.004	0.83 ± 0.012

[a] values are mean proportions over 46 cells ± confidence interval at 95%

Table 6.4 Analysis of the differences in ROC-AUC score using paired t-test.

		heterozygous				homozygous			
Alg1	**Alg2**	**MDE[a]**	**upper[b]**	**lower[c]**	**p-value**	**MDE**	**upper**	**lower**	**p-value**
GenCall	GDA	-0.137	-0.122	-0.151	1.04E-22	-0.165	-0.161	-0.170	5.14E-48
	RF	-0.127	-0.115	-0.140	2.60E-24	-0.142	-0.137	-0.148	1.07E-42
	RF-GDA	-0.182	-0.166	-0.198	2.03E-26	-0.164	-0.159	-0.169	7.00E-46
RF-GDA	GDA	0.045	0.048	0.042	1.33E-34	-0.002	-0.001	-0.003	2.06E-04
	RF	0.054	0.058	0.051	6.07E-32	0.021	0.022	0.021	1.51E-41
RF	GDA	-0.009	-0.006	-0.013	1.56E-07	-0.023	-0.022	-0.025	1.24E-32

[a] mean of differences of algorithm1 and algorithm2 from 46 cells of GM12878
[b] lower bound of the 95% confidence interval
[c] upper bound of the 95% confidence interval

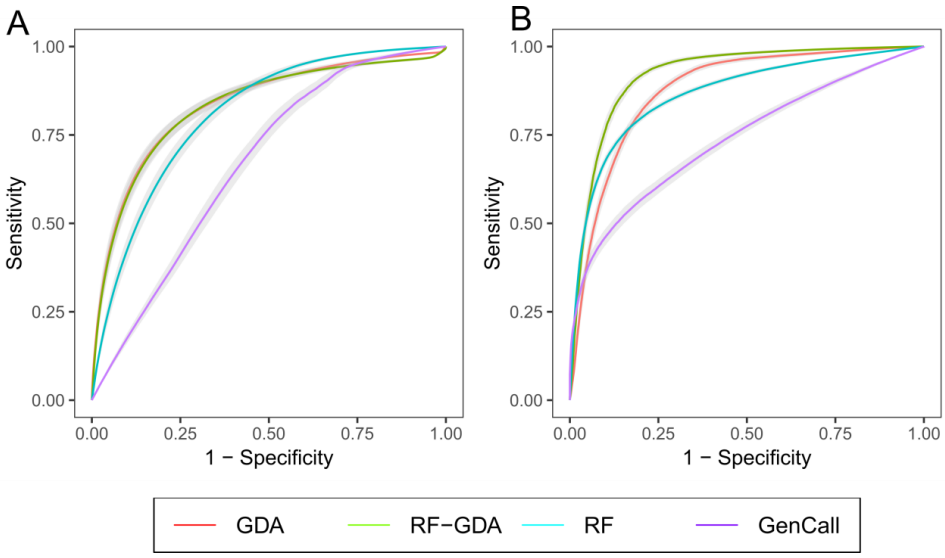

Figure 6.2. ROC curve for homozygous (A) and heterozygous (B) calls.

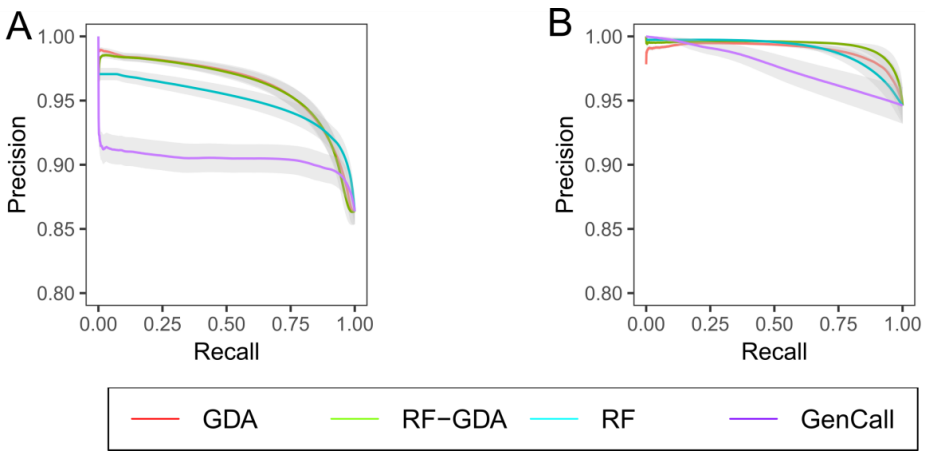

Figure 6.3. Precision-recall curve for homozygous (A) and heterozygous (B)

6.2.2 Evaluation using confusion matrix

Evaluation using confusion matrix (Section 4.7.1) requires classification threshold to be established prior to the evaluation to determine the membership of a SNP to positive or negative class. Similarly to cross validation (Section 6.1), we selected default (and recommended) classification threshold for the GenCall algorithm. For RF, GDA and RF-GDA, we selected thresholds that emphasize on the differences in performance and compared the measurements using paired t-test. Results show that GenCall recalls 68% of the true positive heterozygous genotypes at precision of 97% (Table 6.5). The RF-GDA has 84% recall and achieves average precision of 99% and thus outperforms GenCall in both precision and recall. Having similar precision than RF-GDA, single layers RF and GDA recall fewer true positive heterozygous genotypes (Table 6.5).

High precision and recall are reflected in high harmonic mean of precision and recall (F1-score) for the RF-GDA (Table 6.5) and high rate of correctly classified SNPs (accuracy, Table 6.5). GenCall recalls 96% of the true positive homozygous genotypes on average at precision 89%. At similar recall, the RF alone increases precision by 2.5% ($p < 0.0001$; Table 6.6).

Table 6.5 Performance of the genotyping algorithms on independent dataset GM12878[a]

Alg. / Metrics	GenCall [b]		RF		GDA		RF-GDA [g]	
	het	homo	het [c]	homo [d]	het [e]	homo [f]	het	homo
accuracy	0.68 ± 0.01	0.86 ± 0.012	0.71 ± 0.013	0.88 ± 0.008	0.63 ± 0.009	0.85 ± 0.01	0.84 ± 0.014	0.85 ± 0.01
F1-score	0.8 ± 0.01	0.92 ± 0.007	0.82 ± 0.012	0.93 ± 0.005	0.76 ± 0.011	0.91 ± 0.007	0.91 ± 0.011	0.91 ± 0.007
precision	0.97 ± 0.01	0.89 ± 0.009	0.99 ± 0.001	0.91 ± 0.008	0.99 ± 0.001	0.92 ± 0.008	0.99 ± 0.001	0.92 ± 0.008
recall	0.68 ± 0.01	0.96 ± 0.005	0.7 ± 0.017	0.96 ± 0.001	0.61 ± 0.013	0.9 ± 0.005	0.84 ± 0.017	0.9 ± 0.006

[a] values are mean proportions over 46 cells ± confidence interval at 95%; [b] GenCall score threshold 0.15; [c] Random Forest score threshold 0.6 and [d] 0.15 ; [e] Gaussian Discriminant Analysis score threshold 0.8 and [f] 0.5 ; [g] RF-GDA score threshold 0.15

GDA and RF-GDA further improve precision, but at the cost of recall. Both methods achieve an average precision of 92% at 90% recall for the homozygous calls (Table 6.5). Recalling fewer true positives at higher precision causes a drop in the F1-score for GDA and RF-GDA. This is because recall declines much quicker than the precision increases (Figure 6.3A). The effect of lower recall

from the GDA and RF-GDA is also mirrored in the lower accuracy. As GDA and RF-GDA have higher precision, they are also more likely to reject correct SNPs, thereby decreasing the number of true positives.

Table 6.6. Analysis of statistical differences between the tested algorithms on independent dataset GM12878 with paired t-test

		region	heterozygous				homozygous			
Alg1	**Alg2**	**metrics**	**MDE[a]**	**upper[b]**	**lower[c]**	**p-value**	**MDE**	**upper**	**lower**	**p-value**
GenCall	GDA	accuracy	0.045	0.05	0.04	3.51E-21	0.016	0.019	0.013	1.52E-15
		F1 score	0.040	0.041	0.038	2.53E-43	0.014	0.015	0.013	1.28E-29
		precision	-0.025	-0.016	-0.034	3.17E-06	-0.033	-0.031	-0.035	5.53E-37
		recall	0.064	0.066	0.062	2.19E-48	0.063	0.064	0.062	3.60E-57
	RF	accuracy	-0.035	-0.033	-0.037	1.26E-34	-0.019	-0.014	-0.024	2.19E-09
		F1 score	-0.021	-0.020	-0.023	5.01E-30	-0.010	-0.007	-0.013	2.01E-08
		precision	-0.025	-0.016	-0.034	2.11E-06	-0.019	-0.018	-0.020	8.40E-36
		recall	-0.020	-0.015	-0.026	1.36E-09	0.000	0.005	-0.006	8.69E-01
	RF-GDA	accuracy	-0.169	-0.166	-0.172	8.44E-56	0.015	0.017	0.012	3.68E-14
		F1 score	-0.112	-0.110	-0.114	1.72E-55	0.013	0.014	0.012	1.35E-27
		precision	-0.026	-0.016	-0.036	3.07E-06	-0.035	-0.034	-0.038	5.50E-35
		recall	-0.162	-0.155	-0.168	1.22E-40	0.064	0.066	0.063	1.31E-49
RF	GDA	accuracy	0.080	0.085	0.075	8.24E-33	0.035	0.038	0.032	3.28E-28
		F1 score	0.061	0.063	0.059	8.16E-44	0.024	0.026	0.021	1.12E-24
		precision	0.000	0.000	0.000	1.64E-01	-0.014	-0.014	-0.015	3.24E-37
		recall	0.084	0.089	0.080	3.88E-37	0.063	0.069	0.058	2.73E-26
RF-GDA	GDA	accuracy	0.213	0.221	0.206	2.95E-43	0.001	0.002	0.001	7.31E-06
		F1 score	0.152	0.154	0.150	1.65E-60	0.001	0.001	0.000	1.10E-02
		precision	0.001	0.001	0.001	2.73E-05	0.002	0.003	0.002	3.60E-13
		recall	0.226	0.231	0.220	3.06E-51	-0.001	0.000	-0.002	4.90E-02
	RF	accuracy	0.133	0.137	0.130	3.35E-51	-0.033	-0.030	-0.036	1.42E-26
		F1 score	0.091	0.093	0.089	4.51E-51	-0.023	-0.021	-0.026	6.09E-23
		precision	0.001	0.001	0.000	2.52E-03	0.017	0.018	0.016	1.21E-32
		recall	0.141	0.143	0.139	2.50E-59	-0.065	-0.058	-0.071	1.86E-24

[a] mean of differences of algorithm1 and algorithm2 from 46 cells of GM12878; [b] lower bound of the 95% confidence interval; [c] upper bound of the 95% confidence interval

Evaluation of the single layers

The two-layered architecture, RF-GDA, generally outperforms its constituent single layers (RF or GDA alone). Combining the RF and GDA together is particularly advantageous in the heterozygous regions, where the RF-GDA performs better in all metrics (Figure 6.4). For the homozygous calls, the RF-GDA performs better than single RF and GDA in precision (mean difference 1.7% and 0.2% for RF and GDA, respectively, $p < 0.0001$, Table 6.6). However, the single GDA has better ROC-AUC score, which is 0.2% higher in the GDA than in RF-GDA ($p < 0.001$,Table 6.4). The ROC curves in Figure 6.2 and Precision-Recall curves in Figure 6.3 confirm that the difference is minor, since the RF-GDA and GDA largely overlap. Collectively, the benefits of the two layered RF-GDA compared

to its single layers is the maximized precision and recall for the heterozygous calls. This is due to sensitivity of the EM algorithm to outliers, which are effectively reduced in the RF step (Vogel et al., 2019). There is a further benefit in the maximized precision in the homozygous calls at the relatively modest loss of true positive calls. Interestingly, the cascade approach of RF-GDA was not beneficial when tested on cross validated data (Section 6.1). We assume that this is due to different training and testing performance of the GDA layer. Cross validation uses data from the same dataset (yet different folds) for training and testing. It is therefore likely that the probability distribution of the data in the folds is nearly identical. When, however GDA trained and tested on completely different datasets with possible shift in the distribution between the training and testing data, the performance of GDA is suboptimal compared to RF-GDA, particularly for heterozygous calls. The results therefore suggest that RF-GDA is more adaptable and less sensitive to overfitting compared to single GDA.

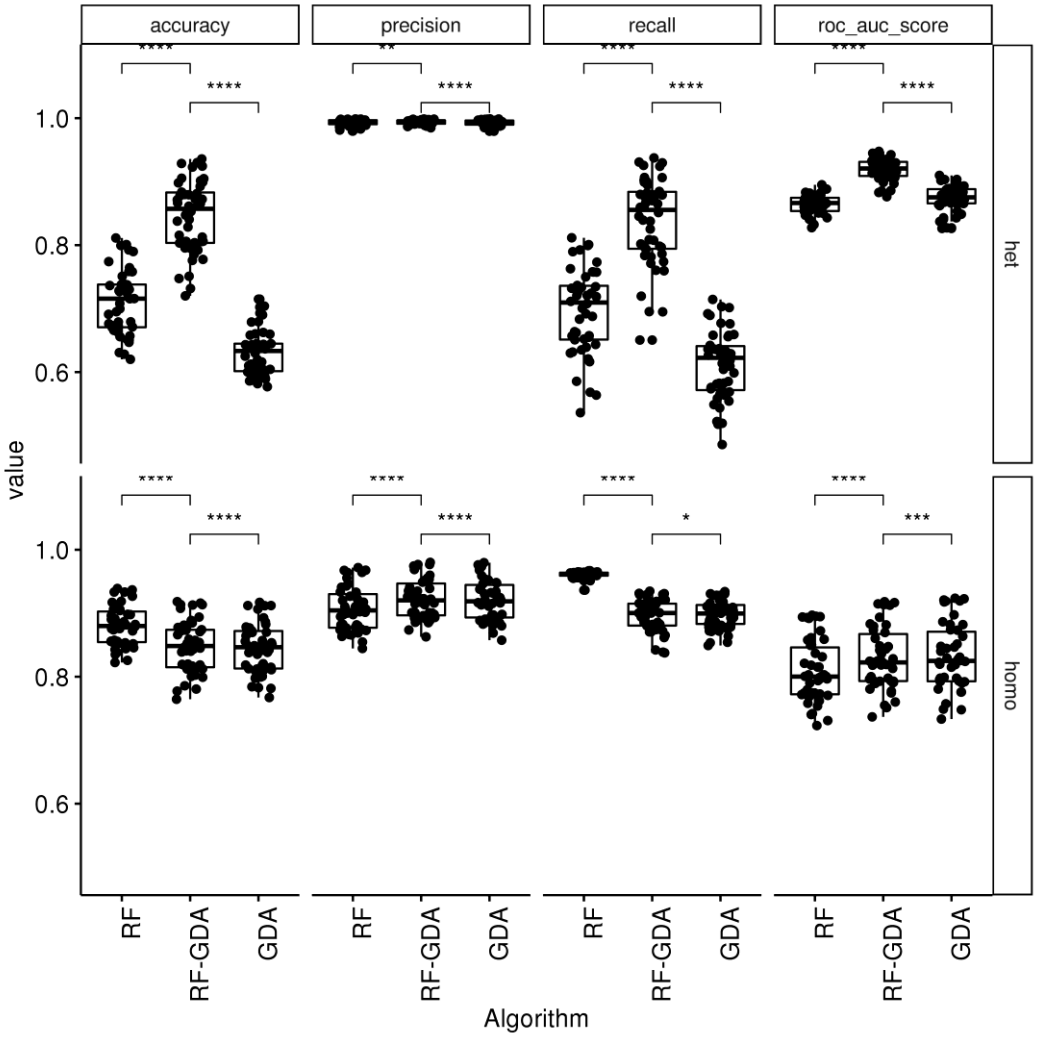

Figure 6.4. Comparison of performance of the single layers (RF, GDA) vs. a combined two layered architecture (RF-GDA) on 46 cells from GM12878. Each dot represents one cell. The pairwise statistics for the 46 single cells was performed using paired t-test. Each asterisk represents level of significance.

6.2.3 Matrix of posterior probabilities

Our observations suggest that SureTypeSC can effectively improve precision of both homozygous and heterozygous SNPs (on average, 99% for heterozygous calls and 92% for homozygous calls, Table 6.5). Having a high precision is a crucial assumption for many single cell analyses as Chapter 8 will discuss. Precision can be further improved at the cost of recall, particularly for homozygous SNPs, as Figure 6.3 suggests. We therefore adjusted both SureTypeSC and GenCall for high precision, recalling ~47% of the true positive SNPs. To compare their performance, we developed a simple statistical toolkit that shows a detailed view of confidence in AA, BB or AB calls using a transition matrix of posterior probabilities (Table 6.7).

Table 6.7 Precision rates with GenCall and SureTypeSC on single cell line from GM12878[a]

			QC001[b]		
REF / SC	**AA**	**AB**	**BB**	**NC**	**Call_rate**
AA	**0.851 ± 0.01**	0.008 ± 0.001	0 ± 0	0.084 ± 0.01	0.364 ± 0.003
AB	0.149 ± 0.01	**0.946 ± 0.012**	0.129 ± 0.008	0.57 ± 0.01	0.151 ± 0.007
BB	0 ± 1e-04	0.046 ± 0.01	**0.87 ± 0.008**	0.076 ± 0.01	0.407 ± 0.002
NC	0 ± 0	0 ± 0	0 ± 0	**0.27 ± 0.01**	0.077 ± 0.003
			GenCall[c]		
REF / SC	**AA**	**AB**	**BB**	**NC**	**Call_rate**
AA	**0.895 ± 0.01**	0 ± 1e-04	0 ± 0	0.268 ± 0.002	0.18 ± 0.002
AB	0.105 ± 0.01	**0.993 ± 0.004**	0.089 ± 0.009	0.385 ± 0.004	0.035 ± 0.003
BB	0 ± 0	0.007 ± 0.004	**0.911 ± 0.009**	0.312 ± 0.003	0.203 ± 0.002
NC	0 ± 0	0 ± 0	0 ± 0	**0.035 ± 4e-04**	0.583 ± 0.007
			GDA[d]		
REF / SC	**AA**	**AB**	**BB**	**NC**	**Call_rate**
AA	**0.967 ± 0.01**	0.002 ± 2e-04	0 ± 0	0.226 ± 0.002	0.189 ± 0.001
AB	0.033 ± 0.01	**0.995 ± 8e-04**	0.022 ± 0.003	0.37 ± 0.003	0.065 ± 0.005
BB	0 ± 0	0.003 ± 5e-04	**0.978 ± 0.003**	0.37 ± 0.004	0.148 ± 0.006
NC	0 ± 0	0 ± 0	0 ± 0	**0.034 ± 6e-04**	0.599 ± 0.01
			RF[e]		
REF / SC	**AA**	**AB**	**BB**	**NC**	**Call_rate**
AA	**0.949 ± 0.007**	0.002 ± 2e-04	0 ± 0	0.266 ± 0.003	0.172 ± 7e-04
AB	0.051 ± 0.007	**0.995 ± 8e-04**	0.038 ± 0.005	0.328 ± 0.005	0.09 ± 0.007
BB	0 ± 0	0.003 ± 5e-04	**0.962 ± 0.005**	0.37 ± 0.003	0.158 ± 0.004
NC	0 ± 0	0 ± 0	0 ± 0	**0.035 ± 6e-04**	0.58 ± 0.01
			RF-GDA[f]		
REF / SC	**AA**	**AB**	**BB**	**NC**	**Call_rate**
AA	**0.972 ± 0.004**	0.002 ± 2e-04	0 ± 0	0.252 ± 0.004	0.17 ± 0.001
AB	0.028 ± 0.004	**0.996 ± 6e-04**	0.022 ± 0.003	0.326 ± 0.005	0.089 ± 0.008
BB	0 ± 0	0.003 ± 4e-04	**0.978 ± 0.003**	0.389 ± 0.004	0.134 ± 0.006
NC[e]	0 ± 0	0 ± 0	0 ± 0	**0.034 ± 7e-04**	0.607 ± 0.013

[a] elements of the table show confidence (precision) rates (and ± confidence intervals at 95%) of a particular SC genotype (column) being genotyped as in reference (row), the last column shows the call rates in the single cell data; NC is no call; [b] GenCall with score threshold 0.01;[c] GenCall at high precision (score threshold 0.87);[d] Gaussian Discriminant Analysis at high precision (score threshold 0.9); [e] Random Forest at high precision (score threshold 0.79); [f] cascade Random Forest and Gaussian Discriminant analysis at high precision (score threshold 0.75); [g]incidence transitions from AA,AB and BB in SC to No Calls (NC) in Reference is always 0; this is due to quality check of the reference - SNPs that are not concordant within the replicates are set to NC and automatically set to NC in the single cell as well

Elements of transition matrix of posterior probabilities $P(g_{ref} \mid g_{sc})$, where g_{ref} is the reference call (in columns) and g_{sc} is the single cell call (in rows); posterior probability in this context is a confidence measure of genotype g_{sc} having a truth value of g_{ref}. $P(g_{ref} \mid g_{sc})$ is calculated using following equation:

$$P(g_{ref} | g_{sc}) = \frac{\sum_i g_{ref,i} = g_{sc,i}}{\sum_i g_{sc,i} = g_{sc}} \quad (7.1)$$

Table 6.7 shows that compared to GenCall, RF-GDA achieves major improvements of 8% and 7% confidence of AA and BB, respectively, and an improvement of 0.3% in confidence of an AB genotype. We also analysed the performance of the single layers. These outperform GenCall as well, but achieve lower precision then RF-GDA.

6.2.4 Reduction of error rates

Incorrect genotype calls arise predominantly from imbalances in the allele frequencies generated during the chemical reaction when the whole genome is amplified. The deviation from a 1:1 allele ratio of heterozygous SNPs can lead to allele drop out (ADO). Analogously, mistyping of a homozygous SNP results in allele drop in (ADI). We calculated the ADO and ADI rates for GenCall and SureTypeSC at high precision using the transition matrices (Table 6.7) and following formulas:

- $ADI = P(AA|AB) + P(BB|AB)$
- $ADO = P(AB|AA) \times P(\mathrm{AA}|\mathrm{hom}) + \mathrm{P(AB|BB)} \times \mathrm{P(BB|hom)}$

Calculation of ADI is straightforward. For calculation of ADO, we have calculate proportion of homozygous single-cell AA calls ($P(\mathrm{AA}|\mathrm{hom})$) and proportion of homozygous single cell BB calls ($\mathrm{P(BB|hom)}$) and multiply these with the corresponding elements of the transition matrix.

Results show that at a call rate (Section 4.7) of 42% for GenCall and 39% for SureTypeSC, GenCall is able to decrease ADI 7 times and SureTypeSC 12.5 times compared to minimal filtering (GenCall QC0.01). The ADO rate is decreased 1.5 times by GenCall and 5.6 times by SureTypeSC (Table 6.8). Although SureTypeSC outperforms GenCall and minimizes the error incidence, the loss of data is inevitable (call rate 39%, Table 6.8).

Table 6.8 Allele drop-in, allele drop-out and call rate with all tested algorithm at high precision

	Min. QC[a]	GenCall[b]	RF[c]	GDA[d]	RF-GDA[e]
ADI	0.05±0.01	0.007±0.003	0.005±0.0007	0.005±0.0007	0.004±0.0005
ADO	0.14±.009	0.096±0.01	0.045±0.006	0.03±0.004	0.025±0.004

Call rate	0.92±0.003	0.42±0.01	0.42±0.01	0.4±0.01	0.39±0.01

[a]GenCall score threshold 0.01,[b]GenCall score threshold 0.87, [c] RF score threshold 0.7, [d] GDA score threshold 0.9,[e] RF-GDA score threshold 0.75

6.3 Experiments

We performed a set of experiments with the model to show the sensitivity towards the parameters and quantity of the training data used. The experiments are summarized in

Table 6.9. Most of the parameters that revealed as optimal by these experiments, were implemented in the program and validated in Sections 6.1 and 6.2. This is clearly distinguished in Table 6.9. We used ROC-AUC score as objective function for measuring the performance. The advantage of ROC-AUC score, as mentioned earlier in the text, is that it captures the overall performance of the classifier and is not dependent on the classification threshold. We also used out of bag score (OOB, Section 4.5) in one of the experiments to evaluate the training procedure of RF.

If not stated otherwise, the default parameters for the experiments were:

- Number of trees: 30
- Training dataset: GM7228
- Testing dataset: GM12878
- Number of input training cells: 58
- Ratio of positive and negative samples: 1
- Data for GDA-model creation: aggregated chromosomes 1-22
- Inter-layer threshold for RF-GDA: 0.5

Table 6.9 Overview of the experiments performed on the model

Category	Parameters tested	Layers tested			Validated[a]
		RF	GDA	RF-GDA	
Robustness	• Number of trees	yes	no	no	yes
Robustness	• Partitioning of the input data	no	no	yes	yes
Sensitivity to size of the training data	• Number of cells included in the training dataset	yes	no	yes	no
Sensitivity to quality of the training data	• Ratio of positive and negative samples in the training dataset	yes	yes	yes	yes
Sensitivity to quality of the training data	• Threshold of the RF-score for the GDA-layer	yes	no	yes	yes

[a] indicates whether optimal parameters revealed by the experiment were implemented and validated in Section 6.1-6.2

6.3.1 Number of trees in Random Forest

We tested how the number of trees *n* in the Random Forest influence the classification performance. We started with *n=5* and incremented up to 100 trees and evaluated the ROC-AUC score in every

iteration. The results (Figure 6.5) suggest that with first few *n*, RF benefits from adding every extra tree to the forest. Figure 6.5 shows that for *n=5*, the average ROC-AUC score is aroud 0.785. For *n=30*, that we chose for the implementation, the ROC-AUC score is ~0.81. Note that from *n=25*, the contribution to the improvement of the ROC-AUC fades and the ROC-AUC score stabilizes at *~0.815* with *n=75*. Based on the testing data, we can conclude that *n=30* gives a good tradeoff between performance and complexity of the Random Forest.

6.3.2 Partitioning the input space for GDA

In the cascade configuration, the output of RF is used to feed the GDA layer (Figure 5.6). As of default, we create one aggregate model for all analysed chromosomes. We were interested how succesful the classification would be if we trained a separate model per chromosome. We therefore ran a comparative analysis where we first classified all chromosomes using one aggregate GDA model (noted as `aggregate`) and then performed an independent run where we fitted a separate model per each chromosome (noted as `single`). We then listed the performance per chromosome (Figure 6.6). We compared the groups analysed with the `aggregate` and `single` model using paired t-test (Table 6.10). Collectivelly, Figure 6.6 and Table 6.10 show that for the majority of the chromosomes (14), aggregate model gives slightly better results. These differences are statistically significant (p-value < 0.0001), however, very small (Table 6.10). As creating one aggregate model is computationally more efficient than creating a separate model per chromosome, it is advantageous to use the aggregate strategy unless some chromosome suffer from aneuploidy (wrong number of chromosomes due to cancer or error in the cell division). Aneuploidy would cause deviation of the signal caused by factor other than MDA (Section 2.2.2).

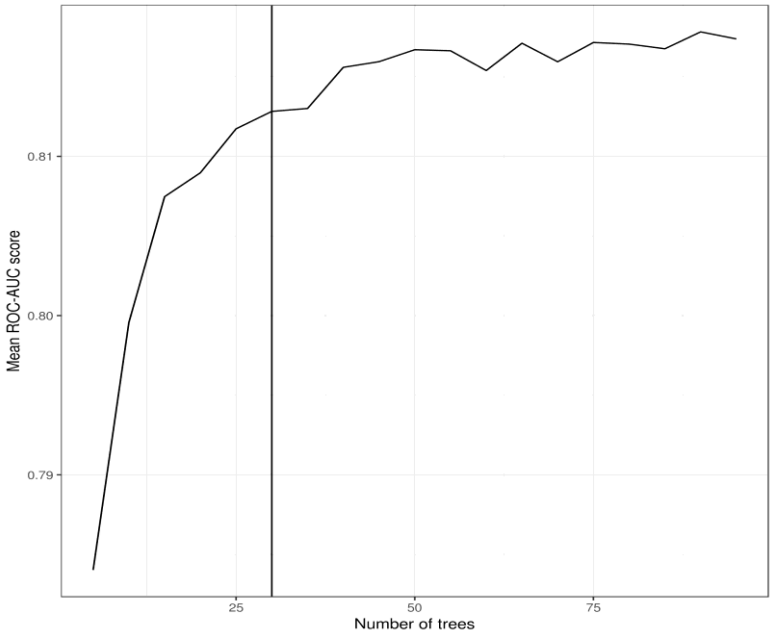

Figure 6.5. Relationship of size of the Random Forest (number of trees) and the performance of the classifier. X axis shows number of trees, Y axis shows mean ROC-AUC score across 46 cells from the testing individual GM12878. The vertical line demonstrates the number of trees in the current implementation (30)

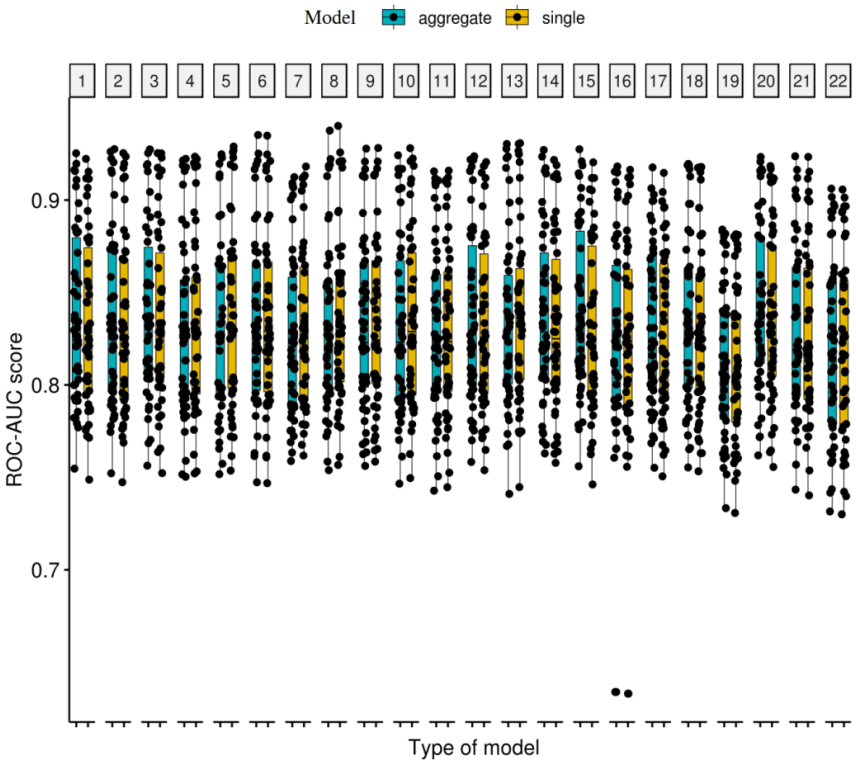

Figure 6.6. ROC-AUC score over 46 cells from GM12878. Every dot represents one cell and every dotplot represents results of an aggregate or single model.

Table 6.10. Results of paired t-test between aggregate and single[a]

Chromosome	MDE[b]	Direction of difference[c]
1	0.005	+
2	0.004	+
3	0.004	+
4	-0.002	-
5	-0.003	-
6	0.001	+
7	-0.004	-
8	-0.003	-
9	-0.001	-
10	-0.004	-
11	-0.001	-
12	0.004	+
13	-0.002	-
14	0.005	+
15	0.010	+
16	0.003	+
17	0.005	+
18	0.002	+
19	0.002	+
20	0.006	+
21	0.002	+
22	0.001	+
Total		14+;8-

[a] measurements were performed across 46 cells from GM12878
[b] mean difference estimate from paired t-test
[c] + means `aggregate` model has better score, - means `single` model has better score

6.3.3 Effects of balanced and imbalanced training dataset

In the original implementation the dataset is balanced prior to the training. We were interested how using the original, highly imbalanced dataset (roughly 9:1 in favor of the positive class) affects the performance. We visualised the results using ROC and Precision-Recall curves. As shown in Figure 6.7 balancing the dataset is advantageous for the first layer of the algorithm (RF) in the heterozygous area of the genome (Figure 6.7A,C), but, surprisingly, does not affect the GDA layer or affects it only minimally. This is probably related to the size of the input training dataset, which is big enough (58 cells) to accumulate sufficient amount of information for the minority class. It will be interesting to observe how sensitive the model is to the size of the input (training) dataset and how this affects the performance of the second layer. The problem of sufficient amount of training data is discussed in the following subsection.

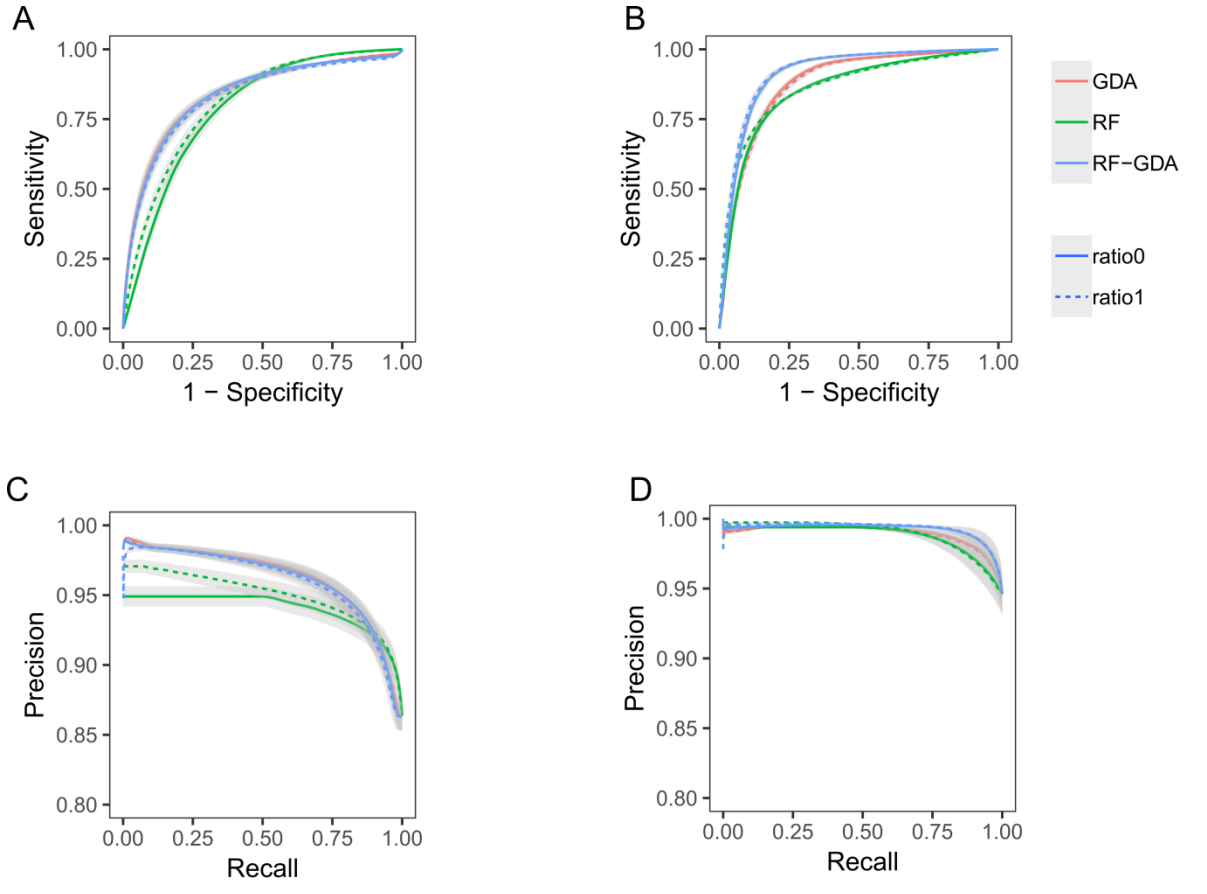

Figure 6.7. Comparison of performance of RF, GDA and RF-GDA using balanced training data (dashed; ratio1) and imbalanced original data (solid line; ratio0). (A) ROC curve for homozygous SNPs; (B) ROC curve for heterozygous SNPs; (C) Precision-Recall curve for homozygous SNPs; (D) Precision-Recall curve for heterozygous SNPs; The bands along the curves represent confidence intervals at 95% over 46 cells of GM12878.

6.3.4 Effect of the size of the training dataset

Here, we measured how number of cells in the training dataset affect the training and testing performance. The training performance was measured for Random Forest using out of bag (OOB)

score. The testing score was assesed as in the previous experiments – using ROC-AUC score. Similarly to Section 6.3.1, where we were systematically increasing number of trees, in this experiment, we were systematically increasing the number of input cells for the training algorithm (interval from $n \in < 1; 58 >$). Figure 6.8 suggests that system with only few cells is relatively unstable – for the first few cells, every addition of a cell to the training algorithm causes shift of the mean score. The effect is similar for ROC-AUC score of the validation data (GM12878; blue curve in Figure 6.2) and for OOB during the training (training on GM7228; green curve). The GDA in the second layer is dependent on the results of the RF and therefore, as expected, ROC-AUC score of RF-GDA has similar trend as ROC-AUC of RF on its own. As the number of cells in the system increases, the changes between the iterations decrease and the system stabilizes at ~40 cells. As the model was originally trained on all available cells (56), these results suggest that lowering the number of cells in the training data will not harm the performance.

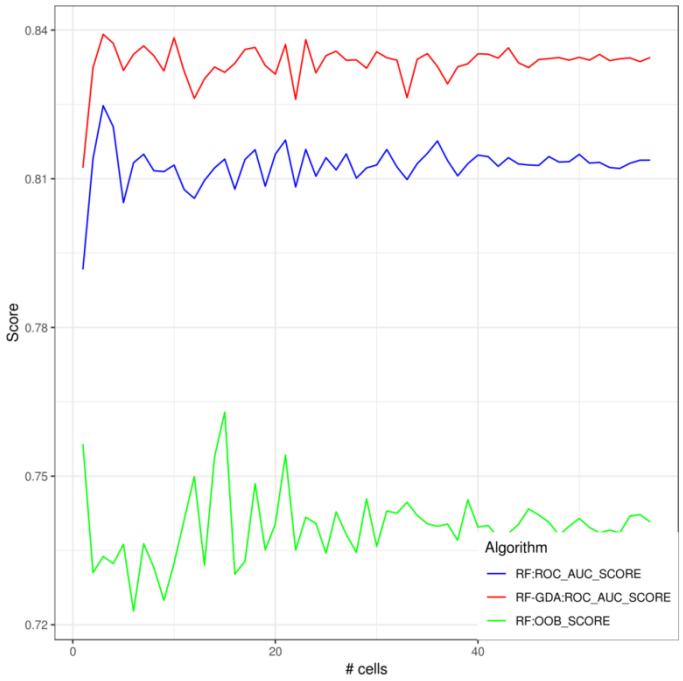

Figure 6.8. Relationship of training dataset size (in number of cells) and the mean ROC-AUC score over 46 testing cells from GM12878.

6.3.5 Effect of the inter-layer threshold

As described previously, the GDA-layer accepts data processed by the RF-layer in the form of positive and negative class (Section 5.5.2). This approach is beneficial as shown in the validation part (Section 6.2). As by default, the positive class for fitting with GDA consists of SNPs that have RF score at least 0.5 and the rest is considered as negative class. We were interested how the RF-score (termed as inter-layer threshold) influences the performance of the GDA layer and therefore changed this parameter systematically. The results are shown in Figure 6.9. As the inter-layer threshold increases, the amount of relevant positive examples for the GDA-input decreases and the amount of

relevant negative examples increases. The peak performance is in interval of inter-layer threshold <0.5; 0.7>. After this interval, the amount of positive samples is likely insufficient to fit the GDA layer and the performance of RF-GDA declines. RF labeled the testing data GM12878 with an average score of 0.66. This number falls into the interval of peak performance. Selecting average RF-score for the inter-layer threshold can be therefore a good rule of thumb for setting the optimal performance of the cascade classifier.

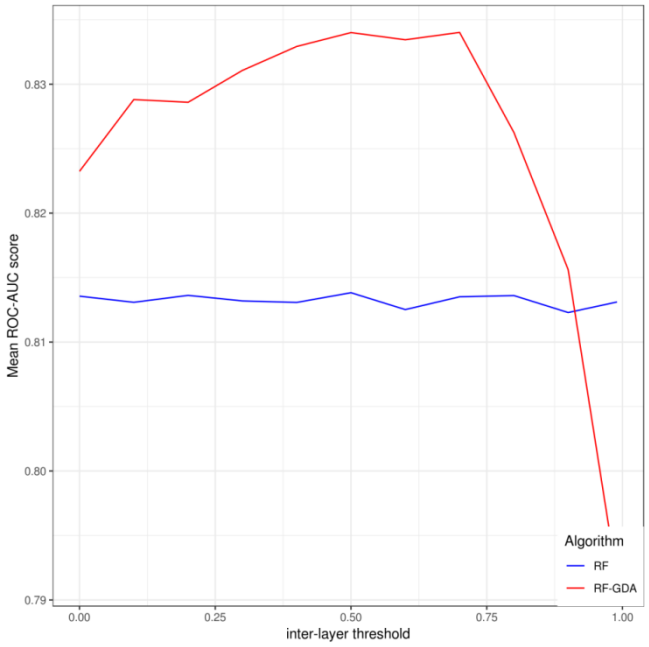

Figure 6.9. Relationship of the threshold for classification output of RF (inter-layer score) and the performance of the consequent GDA layer.

6.4 Conclusion

In the validation part, we have shown that our algorithm can increase accuracy of single-cell genotyping for heterozygous SNPs from 68% to 84%, by increasing both, recall (also from 68% to 84%) and precision (from 97% to 99%). We improve precision for homozygous SNPs from 89% to 92%. This is at the cost of recall. We then discuss how even higher precision can be achieved for single cells by adjusting the classification threshold. The applications for these findings will be explained in Chapter 8.

In the second part of the chapter, we performed experiments with the model, that show:

- Increasing the number of trees in the first layer benefits the overall performance; performance of the RF layer stabilizes with ~50 trees and we do not observe further improvement of the RF layer; after 30 trees, we observe only minor improvements and therefore use this number in the implementation as an optimal cutoff between performance and complexity
- We show that under assumption that we work with normal, healthy samples (i.e. constant and correct numbers of input chromosomes), it is beneficial to aggregate all chromosomes in one

GDA model. Splitting them can locally (for some chromosomes) contribute to slightly better classification performance, however, at the cost of higher complexity and running times (creating separate models per each chromosome).

- Balancing the training dataset benefits the homozygous SNPs in the RF-layer. As RF-layer due its good performance in the homozygous regions can be potentially used independently, this finding has practical implications for the application part of this work
- The number of cells for training of the system contribute to better performance. The training and testing performance fluctuate with the first few cells and then stabilize at ~40 cells. Decreasing the number of cells from 58 to 40 will therefore speed up the learning and likely not harm the performance.
- Choosing the mean RF-score value for the inter-layer threshold gives optimal performance of RF-GDA. Increasing the threshold beyond this negatively affects the performance as the GDA likely does not obtain enough truly positive SNPs to create the model.

7 Knowledge extraction

As presented in the generic workflow in Figure 2.5, creating a model over the intensity data and genotyping is not a final step of the analysis. The whole workflow would have zero practical impact without proper interpretation of the data and analysis of the context. We call this process knowledge extraction and will present few algorithms falling into this category. The presented algorithms are either optimization of the previously published concepts or present original solutions for knowledge extraction – this is clearly distinguished in the text.

7.1 Recombination events in human oocytes

7.1.1 Introduction

We will refer to the biological introduction, particularly to Section 2.1.3, but will elaborate on female reproductive cells. As presented previously, human genome has 23 pairs of homologous chromosomes and each homologous pair has been created from parental chromosomes (green and yellow strands in Figure 7.1A representing maternal and parental DNA). Oocytes are female reproductive organs that are generated during process called female gametogenesis in ovaries. During gametogenesis, the amount of genetic materials first doubles to give rise to primary oocyte (Figure 7.1B). The parental genetic information in the primary oocyte can then undergo recombination between homologous chromosomes (Section 2.1.3) .

Recombination is illustrated as exchange of green and yellow stretches of DNA in Figure 7.1B. The genetic information inherited from the same parent (here, continuous stretch of green or yellow) is called haplotype block. During the first stage of meiotic divison, the homologous chromosomes segregate and give rise to two different cells – oocyte[7] and polar body (PB1, Figure 7.1C). The sister chromatids segregate upon fertilization or artificial activation of oocyte in the lab (Models II-1 and II-2, respectively, Figure 7.1D and Figure 7.1E; Ottolini et al. 2015). During this process, PB2 is extruded (Figure Figure 7.1D, E).

7.1.2 Meiotic pathways and data models

Thanks to advanced laboratory technologies, we are able to obtain data for most of the pathways (models) depicted in Figure 7.1 using the generic workflow (Figure 2.5) and SNP arrays (Section 2.2.3). Analogously to Chapter 5, we can identify two categories of data – accurate information from bulk genomic DNA (Figure 7.1A) and noisy genotype information from single cell DNA represented

[7] We use the term oocyte and egg interchangeably.

by the products of female meiosis (Figure 7.1C,D,E). Depending on whether the cell is diploid or haploid, there are several constraints on the data. The notation for genotypes is consistent with the notation used in Chapter 4.

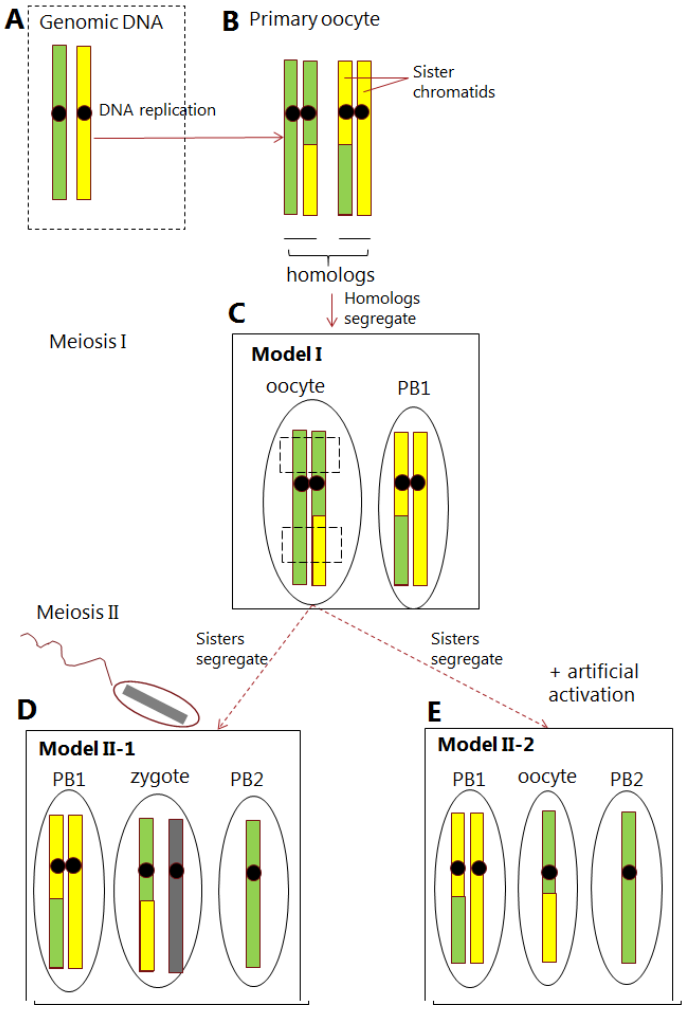

Figure 7.1. Generation of human oocytes. The genomic DNA (A) is replicated and undergoes recombination (B). The homologous chromosomes segregate and give rise to two different cells – oocyte and polar body (C). The sister chromatids segregate upon fertilization (D) or artificial activation of oocyte in the lab (E).

Genomic DNA

Genomic DNA represents diploid information. As this is combination of alleles from maternal and paternal genome, we can observe both heterozygous and homozygous calls. Let G_{gDNA} note the genomic DNA. Then:

$$G_{gDNA} \in \{AA, BB, AB, NC\},$$

where $\boldsymbol{NC}$ corresponds to NoCall or missing value, $\boldsymbol{AA}$ and $\boldsymbol{BB}$ present the homozygous genotypes and $\boldsymbol{AB}$ is the heterozygous genotype.

Model I

Both, PB1 and oocyte are diploid at this stage of meiosis which implies the same constraints on the genotypes as it is in case of the genomic DNA (both homozygous and heterozygous calls are possible). Let $\boldsymbol{G_{PB1}}, \boldsymbol{G_{oo}}$ define the PB1 and oocyte genotypes, then:

$$G_{PB1} \in \{AA, BB, AB, NC\},$$
$$G_{oo} \in \{AA, BB, AB, NC\},$$

Model II-1

At this stage of meiosis, PB1 and zygote are diploid. However, PB2 is haploid and therefore only homozygous calls should appear in this data. Let $\boldsymbol{G_{PB1}}, \boldsymbol{G_{PB2}}$ and $\boldsymbol{G_{zy}}$ define the genotypes for PB1, PB2 and zygote, then following constraints are applied on the genotypes:

$$G_{PB1} \in \{AA, BB, AB, NC\},$$
$$G_{PB2} \in \{AA, BB, NC\},$$
$$G_{zy} \in \{AA, BB, AB, NC\},$$

Model II-2

This model is feasible thanks to artificial activation of oocyte which mimics the presence of sperm. Yet, the sperm genotype is not present, the oocyte remains haploid. Similarly to Model II-1, this implies following rules for the genotypes:

$$G_{PB1} \in \{AA, BB, AB, NC\},$$
$$G_{PB2} \in \{AA, BB, NC\},$$
$$G_{oo2} \in \{AA, BB, NC\},$$

Conventions and limitations of the technology

From previous definitions and Figure 7.1, it is apparent that AB genotype means that the parental alleles are different. We, however, do not have information whether A belongs to the mother and B belongs to the father and vice versa. We only have information about them being different.

Algorithmic detection of crossovers

To summarize the previous biological context we define crossover as the position on the chromosome where the reciprocal exchange of genetic information initiates (recombination) and causes transition of parental haplotypes. This can be detected for both, Model I and Model II illustrated in Figure 7.1. As Model I and Model II contain two and three cells, we refer to them as to Duos and Trios, respectively. To the best of our knowledge, the algorithm for detecting crossovers in Duos is currently not publicly available and we therefore designed an original solution. For detecting crossovers in Trios, we took the concept of an algorithm published in Ottolini et al., 2016, optimized and fully automated for batch processing.

7.1.3 Model I – reconstructing crossovers from Duos

Figure 7.1C shows stage of the meiotic division with two cells that both posses two strands of DNA. It is apparent that the recombinant parts of the chromosome share the same combinations of alleles (green and yellow building heterozygous genotype), whereas the part of the chromosome that did not recombine contain only information from one of the parents (Figure 7.2). If we therefore compare the genotypes from these two cells and mark matching genotypes with 1 and varying genotypes with 0, we would ideally obtain a regular expression $[0]\{n-m\}[1]\{m\}$, where n is the size of the chromosome and m is the size of the recombinant part of the chromosome. The transition point between 1 and 0 would represent the crossover. However, following factors need to be taken into account:

- Genotypes will also match by chance
- Genotypes contain errors causes by MDA (Chapter 5).

The sequence at the output is therefore expected to be $[0|1]\{n\}$, where n is the size of the chromosome.

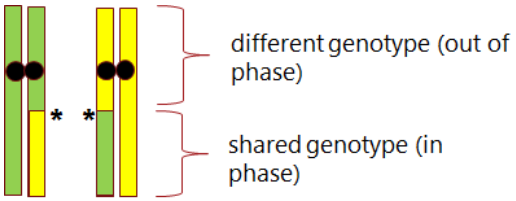

Figure 7.2. Shared heterozygous genotypes and non-shared homozygous genotypes with crossover between them marked with asterisk.

Design of a new algorithm

We designed an algorithm for the crossover detection in Duos (Algorithm 1) that uses simple arithmetics with B allele frequency and runs a function and a procedure in cascade. The B-allele frequency feature is interpolated from the three canonical clusters of the reference SNP array data implemented in GenCall (Section 4.6). Approximate values of B allele frequences intuitively correspond to the proportion of the minor B allele in the genotype and are in Table 7.1

The algorithm assumes that absolute differences in regions that did not recombine (and are therefore homozygous) are close to one. Therefore, it iteratively marks the absolute distances between B-allele frequences of the loci of the two cells.. It is however assumed that these regions can also contain markers with zeros when the corresponding homozygous SNPs from two cells match by chance as explained previously. Sufficiently high threshold (currently 0.95) filters these markers out. This is implemented in function *MarkAndSelect(.).* Subsequently, the candidate homozygous loci (with markers close to 1) undergo segmentation procedure (procedure *SegmentHomRegions*). Procedure *SegmentHomRegions* performs segmentation on the homozygous regions and separates them into potentially multiple homozygous segments. For the actual segmentation, a variational

Bayessian GMM is employed, as this allocates number of segments dynamically (Section 4.3.4). The boundaries of these segments then determine the positions of the crossovers.

Table 7.1 Theoretical values of B allele frequences.

Genotype	B allele freq
AA	0
BB	1
AB	0.5

Algorithm 1: Crossover detection in Duos

Input: *vectors Bfreq_1 and Bfreq_2 in corresponding to two cells from Model I*

Output: *list of crossovers*

1. *CandidateLoci←MarkAndSelect(Bref_1, Breq_2)*
2. *SegmentHomRegions(CandidateLoci)*

Function: MarkAndSelect(.)

For chromosome in Chromosomes

For locus in chromosome:

Mark[locus] ← abs(Bfreq_1[locus]-Bfreq_2[locus])

Select[locus] ← Mark[locus]>threshold

Return Select

Procedure: SegmentHomRegions(.)

For chromosome in Chromosomes:

Segment(chromosome)

ReportCrossovers()

7.1.4 Model II – reconstructing crossovers from Trios

The algorithm for finding crossovers in Trios is more complex due to number and types of cells involved – we operate with Model II-2, as this does not contain the paternal DNA (sperm) and therefore allows better resolution (Ottolini et al., 2015). The algorithm firstly only selects heterozygous SNPs from maternal genomic DNA (Figure 7.1A). These are termed as informative, as they contain both alleles that can be tracked in the PB1, PB2 and Egg (Figure 7.1E). Secondly, the algorithm compares one of the haploid cells (PB2 or Oocyte) from the same individual to all the other cells. This haploid cell is termed as hypothetical common ancestor or reference and its purpose is to determine the origin of the haplotype blocks (green and yellow strand in Figure 7.1). We call the regions that are shared with the reference as `in phase` and the procedure of determining the origin of

haplotype blocks as phasing. The reference, although initially assumed to be a homogenious haplotype block, also potentially carries crossovers. If this is the case, an artificial crossover would appear in all the other cells that underwent phasing. This needs to be examined (Figure 7.3) and phases corrected by adding a crossover to the reference (Ottolini et al. 2016).

The original implementation of the algorithm requires user interaction and manipulation throughout the analysis (Ottolini et al. 2016). I.e. the transitions between the haplotype blocks and common crossovers need to be visually inspected and marked and corrected manually. We reflected these drawbacks on following changes to the design of the algorithm:

- we automated the process by designing algorithm for resolving common crossovers and transitions between the haplotypes
- we optimized the algorithm by designing operations on matrices and vectors that can be efficiently implemented in one of the high level numerical libraries (Numpy for Python, Walt et al., 2011)

The pseudocode of the optimized algorithm is shown in Algorithm 2. Details on the functions are below Algorithm 2. Note that the requirements for minimal number of input trios ($n >= 3$) were previously validated in Ottolini et al, 2016.

Algorithm 2: Crossover detection in Trios

Input: *D={n vectors of genotypes- gDNA, PB1, PB2 and Egg} from the same individual}, where |n|=1+3x and x is number of trios (>=3)*

Output: *Cx={list of crossover positions per cell per chromosome}*

1. *Mask out loci where $gDNA \in \{AA, BB\}$ and $PB2 = AB$ and $Egg = AB$*
   //we only operate with informative SnPs and het PB2 and Egg are likely an error
2. *Convert genotypes to B-allele frequencies (AA←0, BB←1 AB←0.5)*
3. *Phase(D, Ref): perform | X-Ref |, where $X \in \{PB1, PB2, Egg\}$*
   a. *Return D_{phased}*
4. *Smooth_Haplotype(D_{phased}): apply 1D mode filter to phased data*
   a. *Return $D_{smoothed}$*
5. *ResolveCx($D_{smoothed}$) – resolve common crossovers in reference*
6. *Phase($D_{smoothed}$, $Ref_{resolved}$)*
7. *Report crossovers and haplotype blocks*

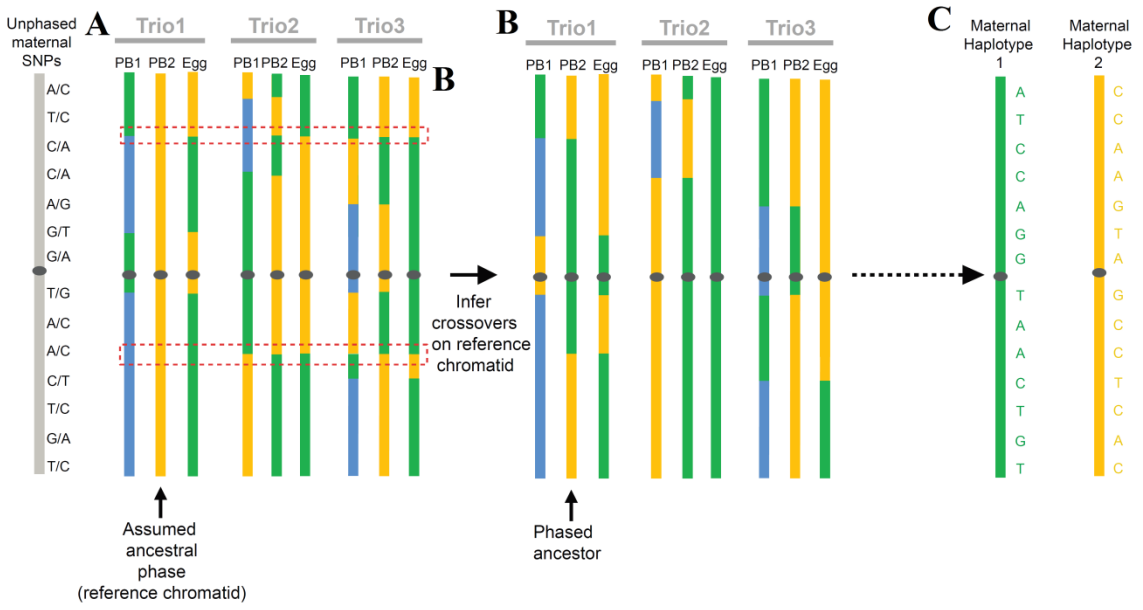

Figure 7.3. Demonstration of the phasing procedure and crossover detection using a single reference (PB2 from Trio1). (A) Firstly, the cells from all the other trios are compared to the reference chromatid and labeled in yellow or orange depending on the cell being in phase with the reference or not, respectively. Secondly, in order to resolve crossover in the reference chromatid, common crossovers in all the other haploid cells are detected (red dashed rectangles), removed and added to the reference chromatid (B). (C) Genotype of the phased maternal haplotype (Ottolini et al., 2015).

Function Phase

The function applies fast arithmetic operations to all chromosomes in all cells in the trio that compares all data to the reference cell. It returns mask to every chromosome present in the dataset with 0 for loci in phase with the reference, 1 for loci out of phase and 0.5 for heterozygous loci.

Function Smooth_Haplotype

Function processes the phased genotypes with 1D mode filter. 1D mode filter is analogous to median filter with mode as the core function. Median filter is a non-linear technique to remove noise (or generally outliers) from the data and prevent edges (Bovik et al., 1987). The edges are boundaries of the phases/haplotypes and there are two categories of outliers present in the data:

- True technical noise caused by whole genome amplification
- Heterogeneity of the heterozygous region as this can not only possess AB, but also AA and BB if the alleles are matching by chance (Section 7.1.3)

Table 7.2 demonstrates this problem and shows motivation for application of a smoothing filter. The task is to unify the phases by removing the erroneously phased genotypes caused by aforementioned reasons, but preserve the edges (crossovers) to obtain two haplotypes and a heterozygous region. Similar approach was previously used for haplotype smoothing in Zamani Esteki et al. (2015). Algorithm 3 shows the design of the smoothing filter, that uses 2D matrix to store (or reference) elements of the sliding window of size *k*. The size of the window for chromosome *chr*, W_{chr}, is calculated dynamically per chromosome using Eq 8.1 (Zamani Esteki et al., 2015). The calculations take only informative SNPs, that is loci that are heterozygous in the genomic DNA.

$$W_{chr} = round\left(\frac{nPM_{chr}}{nPM_1}\right) \times W_1, \quad (7.1)$$

where W_{chr} is chromosome k-specific window, W_1 is the size of the window for chromosome (currently set to 21 informative SNPs), nPM_{chr} is total amount of informative SNPs for chromosome *chr* and nPM_1 is the total amount of informative SNPs for chromosome 1.

Table 7.2 Demonstratrion on how errors in genotyping affect phasing procedure

SNP ID	Diploid cell	Reference	Phase	Status	Haplotype	Crossover
1	AA	AA	0	Correct	0	FP due ADI (SNP 1 and 2)
2	AB	AA	0.5	ADI		
3	BB	BB	0	Correct		No crossover
4	AA	AA	0	Correct		TP (SNP 4 and 5)
5	AA	BB	1	Correct	1	
6	AA	BB	1	Correct		No crossover
7	BB	AA	1	Correct		TP (SNP 7 and 8)
8	AB	AA	0.5	Correct	0.5	
9	AB	BB	0.5	Correct		FP (SNP 9 and 10)
10	AA	AA	1	ADO/Correct		
11	AB	BB	0.5	Correct		FP (SNP 10 and 11)

Algorithm 3: Function Smooth_Haplotype[8]

Input: *vector x and window size k, where k mod 2 =1*

Output: *Smoothed vector* x_{smooth}

1. *Calculate center of the window* $k_2 = (k-1) // 2$
2. *Init matrix* $y = | |x| \times k |$
3. *Copy x to* k_2*-th column of matrix y*
4. *For i =0 to* k_2*-1*
    a. $j = k_2 - i$
    b. *Copy elements x[0 to* $|x|-j] \rightarrow y[j,i]$
    c. *Copy* $x[0]$ *j-times* $\rightarrow y[0,i]$
    d. *Copy* $x[j \text{ to } |x|-1] \rightarrow y[0 \text{ to } |x| - j - 1, |x| - (i + 1)]$
    e. *Copy* $x[|x|-1]$ *j-times* $\rightarrow y[|x| - j \text{ to } |x| - 1, |x| - (i + 1)]$
5. *For every column in y:*
    a. *Calculate mode and add to* x_{smooth}

[8] The design of the algorithm was inspired by https://gist.github.com/bhawkins/3535131

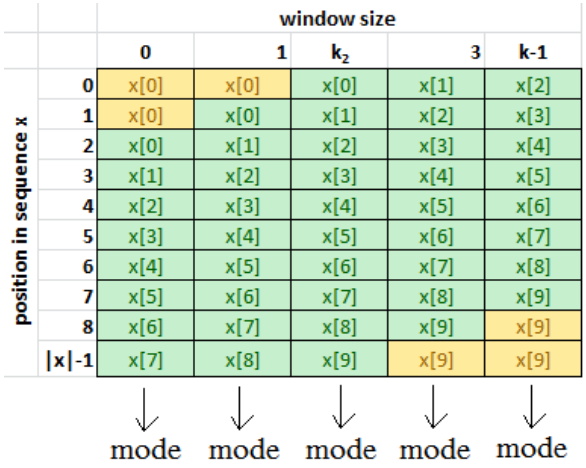

Figure 7.4. 2D translation of the mode filter sliding window for k=5 an on sequence x with length 10. Every column contains one-step walk of the sliding window. The boundaries are extended by k2=2

Note that 4c and 4e assures the endpoints are extended k_2 times to meet the condition |x| = $|x_{smooth}|$ and operation Copy is copy by reference (indices of *x*). Translating the problem into 2D array allows implementation in a fast numerical framework in high level language, i. e. Numpy in Python (Van der Walt et al., 2011). A principle of populating the 2D matrix Algorithm 3 is shown in Figure 7.4.

Function ResolveCx

Function creates a matrix of crossovers and subsequently searches for common crossovers that indicate crossover in the reference cell. As crossovers are defined by transition of haplotypes in a matrix (haplotype transition is indicated with 1 in matrix *cx*, Algorithm 4), the number of common crossovers per locus is sum over all columns (*colsum* in Algorithm 4) The crossover is then subsequently resolved and phases are added to the reference sequence.

Algorithm 4: Function ResolveCx

Input: *Collection of phased and smoothed trios* D_{phased} *from the same individual including reference Ref without crossovers*

Output: $Ref_{resolved}$ *with resolved crossovers and added phases*

1. *Init matrix of crossovers cx of the same dimensions as* D_{phased}
2. *For cell in* D_{phased}*::*
    a. *For pos in cell*
        i. *cx[pos,cellid] ← phase[i]!=phase[i+1]*
    b. *For row in cx:*
        i. *If colsum(row) = total number of haploid cells in* D_{phased}
            a. *Ref[row] ← 1*
3. *Init* $Ref_{resolved}$ *with dimensions identical to Ref*
4. *For pos in Ref:*
    a. $Ref_{resolved}$ *[pos] = cumsum(Ref[pos]!=Ref[pos+1]) mod 2*

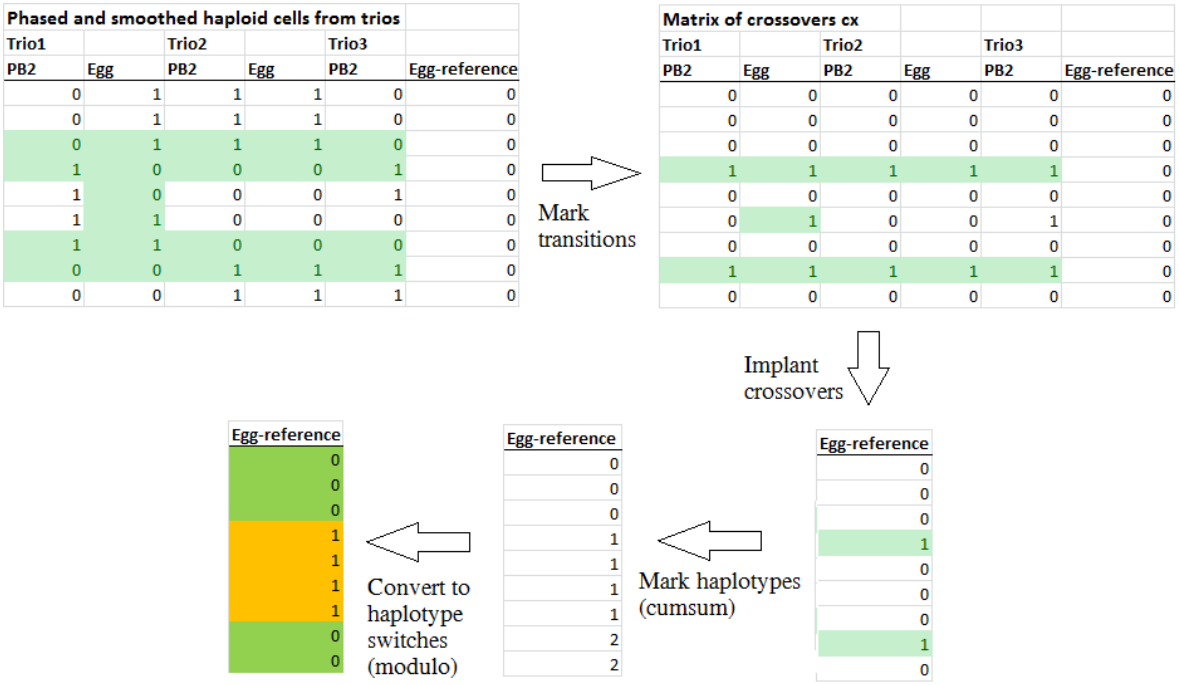

Figure 7.5. Visualisation of the algorithm for resolving common crossovers.

Note that step 2ai in Algorithm 4 marks all transitions between haplotypes and then 2bi checks for common crossovers. Once the crossovers are added to the reference, step 4a corrects for haplotype transitions between introduced crossovers using cumulative sum (*cumsum*). A visual example of the algorithm *ResolveCx* is in Figure 7.5. After the common crossovers are resolved, the cells undergo a second round of phasing – now with a phased reference. The procedure *Phase* in Algorithm 2 is universal in this matter and can be again used to adjust the phases of all cells in respect to crossover in the reference cell.

7.2 Gene conversions

7.2.1 Introduction

Crossover and recombination contributes to heterogeneity of the human population by shuffling the genetic information during meiosis. While crossover is a reciprocal transfer of genetic information that maintains the same amount of alleles for every locus, gene conversion is non-reciprocal. A required condition for both, crossover and gene conversion is presence of double strand breaks (DSBs). DSBs are subsequently resolved by either crossover, or gene conversion. In case of gene conversion, the missing gap in DNA is synthetised in favor of one of the alleles. This causes allelic imbalance (Figure 7.6). The detailed mechanism is explained thoroughly elsewhere (Chen et al., 2007).

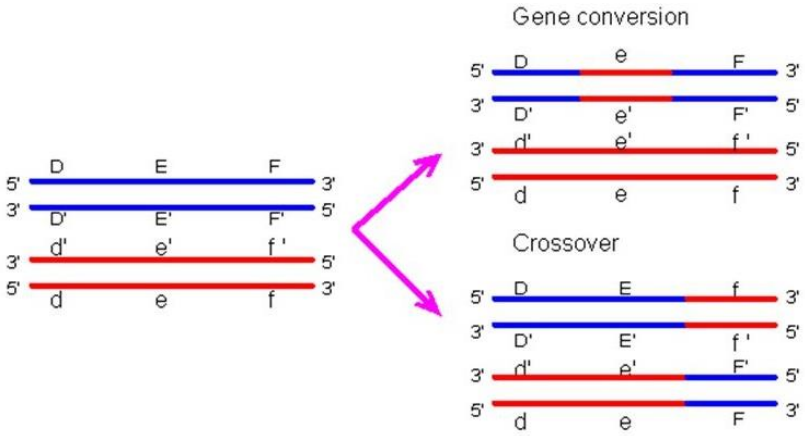

Figure 7.6. Gene conversion vs. crossover[9]

Thanks to availability of the data for all products of female meiosis (Section 7.1), we are able to formulate an algorithm to detect the gene conversions directly.

7.2.2 Detection of GCs in Trios

The detection of gene conversions is based on mendelian inheritance patterns (law of segregation). The design of the algorithm is summarized in Algorithm 5 and the principle is that every locus that shows allelic imbalance (different counts of allele A and B over all three cells per locus) is a potential gene conversion. The algorithm takes into account only heterozygous PB1 as confidence in heterozygous genotypes is generally higher (Table 6.7).

Algorithm 5: Detect GC

Input: *quadruplet of genotype vectors Q={gDNA, PB1, PB2, Egg} from a single individual*

Output: *list of GC*

1. *Convert genotypes to B-allele freq: AB ← 0.5, AA ← 0, BB ← 1.0*
2. *Select loci where gDNA=0.5 and PB1=0.5*
3. *For every locus l in Q:*
    a. *If* $4 \times gDNA[l] - (2 \times PB1[l] + PB2[l]) \; ! = 0$
        i. *Report GC at locus l*

Due to potential noise present in the data, Algorithm 5 in this form requires high confidence genotype data. This will be discussed in Chapter 8.

7.3 Conclusion

In this chapter we presented algorithms for knowledge extraction from single-cell data, namely data from female meiosis. We partly discussed how noise would theoretically affect performance of these

[9] Figure taken from http://www.web-books.com/MoBio/Free/Ch8D4.htm

algorithms. We will elaborate on this in the next chapter and show practical examples of biological inference.

8 Applications of SureTypeSC

8.1 Introduction

In this chapter, we demonstrate practical applications of the novel algorithm for filtering genotypes (Chapter 5) and algorithms for knowledge extraction from the single-cell data (Chapter 7). We will use both, data from female meiosis and reference dataset described in Section 5.4. The goal is to show performance of the novel filtering algorithm on these data in comparison with the state of the art algorithm, GenCall and how the quality of genotyping affects the knowledge extraction. We operate with various thresholds of the algorithms depending on the application.

8.2 Improved crossover detection

8.2.1 Duos

We demonstrate the functionality of the algorithm for crossover detection in Duos (Algorithm 1, Section 7.1.3) on chromosome 1 of a PB1-oocyte duo from Gruhn et al. (in resubmission). We were interested how the algorithm performs on original unfiltered data and with different stringencies of GenCall and SureTypeSC. The results are shown in Figure 8.1. The strategies for testing different thresholds were following (number of markers after filtration corresponds to number of points in Figure 8.1):

a. Raw data without any filtration (`Raw data`, 22,905 markers) and standard GenCall genotyping (QC015, 18,111 markers)
b. Standard RF-GDA from SureTypeSC (RF-GDA015, 16,442 markers)
c. GenCall with high stringency (QC087, 8,799 markers)
d. RF-GDA057 with threshold that accepts similar amount of markers as QC087 (8751 markers)
e. QC095 as an example of extreme filtering of GenCall with only few points left.

The results suggest that increased stringency contributes to clearer homozygous segments by removing poor quality signal – this is true for both, GenCall and RF-GDA. However, RF-GDA shows higher specificity towards the noise for similar number of markers –the segments are clearer with less noise (Figure 8.1B vs. Figure 8.1C and Figure 8.1D vs. Figure 8.1E). Lower specificity of GenCall towards noise is confirmed in Figure 8.1F, where highly stringent GenCall (QC095) still fails to reject few likely false positives (dispersed green points around green cluster in Figure 8.1F).

Concordant with our validation studies in Chapter 6, RF-GDA075 (threshold used for high precision with empirical error rates in Table 6.7) effectively cleans up the signal and enables the clearest separation of the homozygous clusters with the segmentation algorithm. Note that the red

singleton in Figure 8.1G could be either FP left out by SureTypeSC or gene conversion (Section 7.2). The algorithm currently takes this marker as a boundary for the crossover (Fig. 8.5H), which might be wrong.

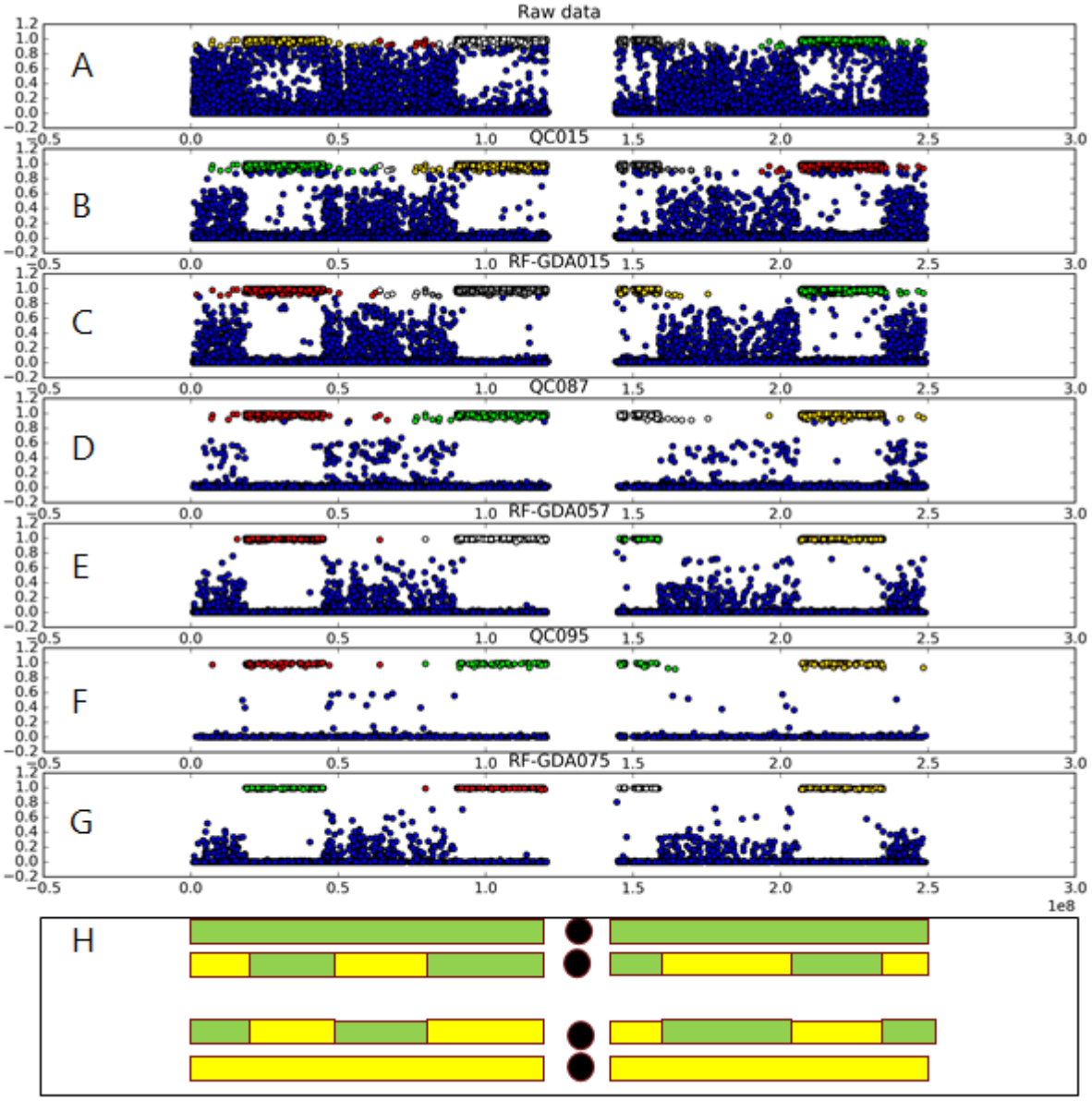

Figure 8.1. Visual representation of the results of the crossover detection algorithm in duos using different filtering strategies. The points other than blue represent clustered segments of homozygous data. (A) Raw data is the genotyping data from GenCall from GenomeStudio with no filtering. (B) is the data filtered with standard GenCall threshold 0.15. (C) is raw data filtered with algorithm RF-GDA from SureTypeSC using standard threshold 0.15. (D) is GenCall algorithm with stringend threshold 0.87. (E) RF-GDA with threshold that maintains the same amount of markers as QC087. (F) GenCall with highly stringent threshold close to 1. (G) RF-GDA previously validated for high precision. (H) Likely crossovers inferred from boundaries of the segments clustered from G.

8.2.2 Trios

We applied Algorithm 2 presented in Section 7.1.4 for detection of crossovers in Trios and were interested in performance of the algorithm with data from standard GenCall (QC015) and data filtered with RF-GDA. We demonstrate the differences between the results on chr17 from one trio from a

data collection from University of Copenhagen[10]. Comparison of the outputs of both genotyping strategies (Figure 8.2) indicates, that:

- The density of the heterozygous SNPs is higher with RF-GDA075 – this is expected and is concordant with the validation study that SureTypeSC can resolve more true positive heterozygous SNPs (Chapter 6)
- As a direct consequence of the previous point, we observe artificial crossovers and transitions between haplotypes in the heterozygous regions after smoothing with data from GenCall (QC015, Fig. 8.10B). The theory of artificial crossovers was illustrated in Table 7.2. An artificial haplotype transition is non-reciprocal (lacking support in PB2 and Egg). I.e. haplotype switch in PB1 from heterozygous to haplotype1 should trigger haplotype switch in one of the other cells, otherwise it is considered artificial. These events are marked with rectangles in Figure 8.2B.

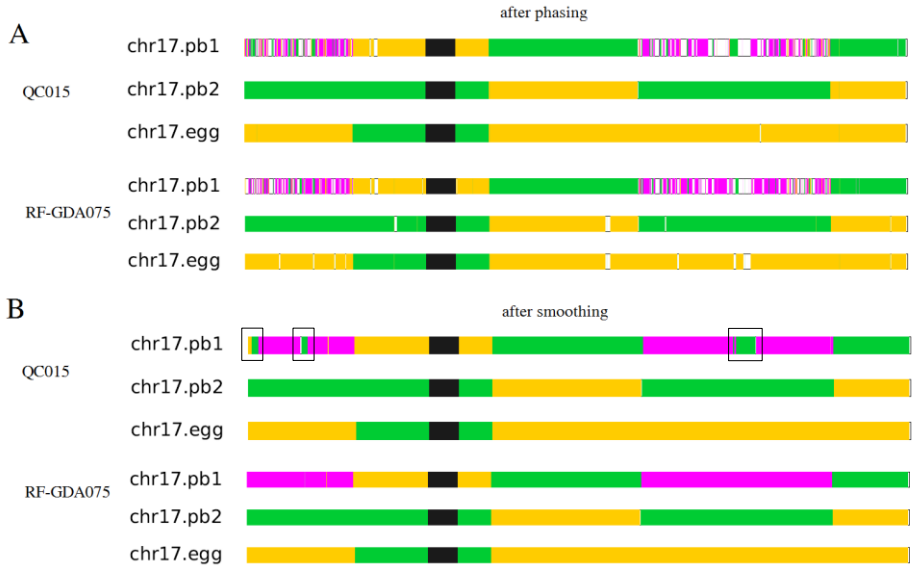

Figure 8.2. Results of the the phasing stage and the smoothing stage of the crossover detection algorithm fro Trios for data coming from GenCall with standard threshold (QC015) and RF-GDA075. Heterozygous regions (containing both green and yellow haplotypes) are in violett , and the parental phases in green and yellow. The solid rectangles indicate artifical haplotype blocks.

8.3 Direct detection of gene conversions

8.3.1 GenCall

Algorithm 5 for detection gene conversion (Section 7.2) does not distinguish between gene conversions and errors caused by MDA amplification. As mentioned in Section 7.2, Algorithm 5 takes into account only loci heterozygous in PB1. As incidence of ADI is much lower than ADO (Table 6.8), heterozygous genotypes are more reliable (Table 6.7) and the probability of transition between two homozygous genotypes (AA ↔ BB; Table 6.7) is close to zero. The possible scenarios

[10] https://icmm.ku.dk/english/research-groups/hoffmann-group/

are then listed in Table 8.1. Running the algorithm on the data with QC015 (empirical error rates in Table 6.8) reveals 2449 gene conversion events with heterozygous PB1 (Figure 8.3). As we are taking into account only heterozygous SNPs these span about 16% in a diploid cell (around 50,000 heterozygous SNPs in 300k microarray, Table 5.3). This fraction corresponds to $5.12 \times 10^6 bp$ in human genome. Assuming the gene conversion tract has length around 100 bp on average (Padhukasahasram and Rannala, 2013), 2449 detected gene conversions span roughly 250,000 bp. This corresponds to gene conversion rate of 4.9×10^{-2} per bp, per meiosis. Rate of gene conversion has been previously estimated as 5.9×10^{-6} per bp, per meiosis (Williams et al., 2015), which is roughly 10.000 times less than what we predicted. It is to conclude that most of the detected gene conversions from genotypes using standard genotyping using GenCall are false positives and product of noise.

Table 8.1. Possible combination of alleles where loci in PB1 are heterozygous

PB1	***PB2***	***Egg***	***Status***
AB	*AA*	*BB*	*Mendelian inheritance*
AB	*AA*	*AA*	*Gene conversion BB→AA or ADI in PB1*
AB	*BB*	*AA*	*Mendelian inheritance*
AB	*BB*	*BB*	*Gene conversion AA→BB or ADI in PB1*

8.3.2 SureTypeSC

We were futhermore interested in numbers of gene conversion after we filter the genotypes with SureTypeSC aimed for high precision (RF-GDA with error rates in Table 6.8). Figure 8.3 shows the results on top of the results of standard GenCall. The number of gene conversion events is reduced to only 10. This corresponds to gene conversion rate of 1.9×10^6 per bp, per meiosis. A lower number than reported in Williams et al., 2015 can be explained by the fact that we only taken into account the heterozygous part of the genome of PB1, but neverthless suggests that we were able to separate noise from the truth signal and detect gene conversions directly.

8.3.3 Validation with NGS data

To asses whether the 10 gene conversion that we detected on data with SureTypeSC are real, we performed whole genome next-generation sequencing (Section 2.2.4) at a target depth 50x. We used standard worklow (Figure 2.5) for preprocessing of the NGS data (Section 3.2) and computational pipeline GATK for genotyping (Van der Auwera et al., 2013). Table 8.2 shows that four conversions (in bold) were confirmed by NGS.

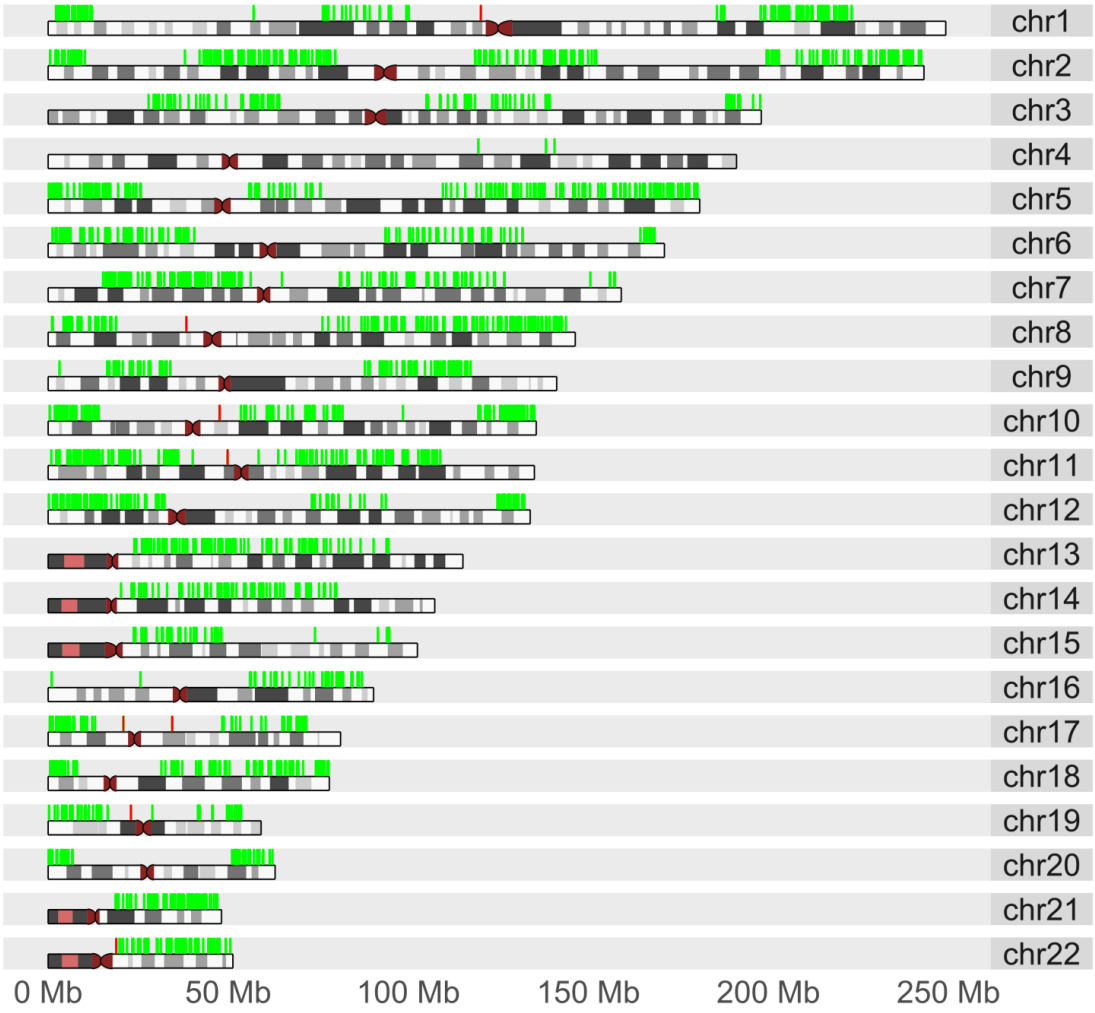

Figure 8.3. Results of gene conversion detection on a trio from a randomized oocyte collection. Green – gene conversions detected by GenCall (QC015); red – detected gene conversion after data filtering with SureTypeSC.

Table 8.2. Results of the validation of putative gene conversions from a randomized oocyte collection[1]

SNP ID	Chr	Position	SureTypeSC (SNP array)				NGS		
			gDNA	PB1	PB2	Egg	PB1	PB2	Egg
rs4659015	1	120145883	AC	AC	C	C	AA	C	C
rs12682402	8	38381884	TC	TC	C	C	TT	C	C
rs17835873	10	47605782	**AG**	**AG**	**A**	**A**	**GA**	**A**	**A**
rs7074244	10	47608158	**AC**	**AC**	**A**	**A**	**CA**	**A**	**A**
rs11259756	10	47634190	**TC**	**TC**	**T**	**T**	**TC**	**T**	**T**
rs11530438	11	49836893	TC	TC	T	T	CC	T	T
rs1045162	17	20903080	AC	AC	C	C	AA	C	C
rs9303700	17	34450463	**TG**	**TG**	**G**	**G**	**GT**	**G**	**G**
rs448372	19	22993128	TC	TC	G	G	AA	G	G
rs2005736	22	18912119	TC	TC	C	C	TT	C	C

[1] the alleles from SNP array were converted to nucleotides[11]

[11] Script for converting alleles to nucleotides available at https://github.com/Illumina/GTCtoVCF

8.4 Detection of copy number variants

8.4.1 Introduction

To this end, we assumed that the cells have correct numbers of chromosomes (Section 2.1.3). However, aneuploidies and copy number variants (CNVs) add an extra level of uncertainty to the data. Aneuploidy is presence of an abnormal number of chromosomes in a cell. In context of female meiosis, it is estimated that 5 % of pregnancies are affected by this chromosome abnormality, which can be fatal for the fetus or cause major genetic disorders (Hassold and Hunt, 2001). CNVs are conditions where only part of the chromosome is deleted or duplicated. Similarly to aneuploidies, they can have fatal consequences to the female oocyte (Martin et al., 2015).

8.4.2 Results

We were interested whether SureTypeSC would improve biological insight when used for high precision (RF-GDA with stringent threshold, Table 6.7 and Table 6.8). We assessed copy number variants (CNVs) in human oocyte data (Ottolini et al., 2015) and successfully resolved chromosomal deletion. The loss of a chromosome or chromosome segment results in one cell with only A and B calls (no heterozygous SNPs). The loss, however, is obscured by ADI when using the standard GenCall algorithm (Figure 8.4). SureTypeSC removes the ADIs (erroneous AB), increasing the certainty of the inference (Vogel et al., 2019).

8.5 Detecting subpopulations in the single cells

8.5.1 Introduction

In this section, we will return back to our reference single cell dataset GM12878 that we used for testing of SureTypeSC (Chapter 6). To this end, we assumed that all variants in the cell line that do not match the reference genome (`erroneous variants`) are allele drop outs or allele drop ins (Figure 8.5A). However, it has been previously reported, that the populations of cells of the same type are often heterogeneious (Altschuler and Wu, 2010). We were curious whether some of erroneous variants could be real and therefore used to detect heterogeneity within a cell population. We were interested in whether we could use SureTypeSC with high precision to reveal biological variability within the tested cell line GM12878.

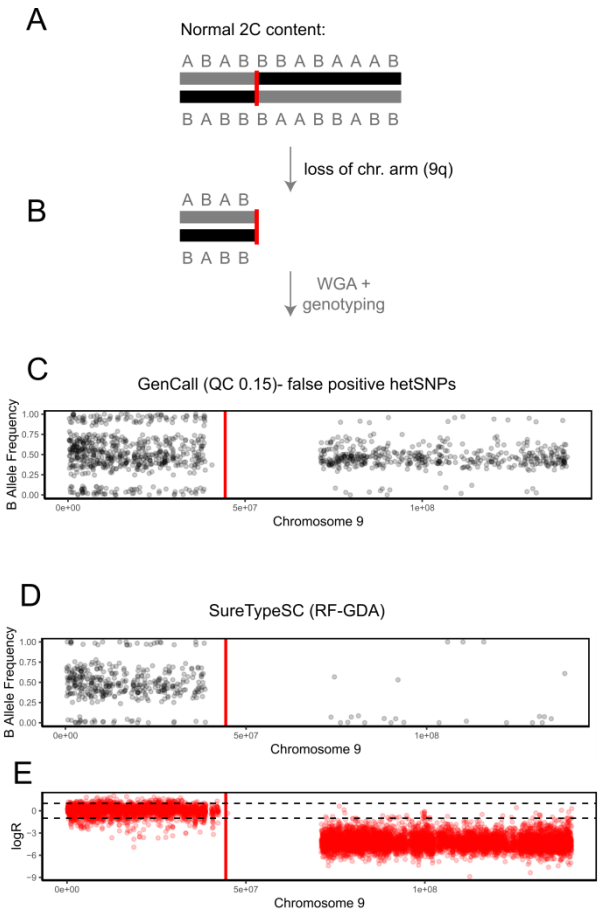

Figure 8.4. Segmental aneuploidy in a human oocytes. (A) The oocyte (Model I) is diploid, but contains a partial deletion (the entire q arm). (B) After WGA, the genotyping algorithm should reject the signal noise from SNPs on the q arm. However, the B allele frequency of genotyped SNPs passed the quality control of GenCall (C). In contrast, the B allele frequency plot of SureTypeSC (D) correctly rejects genotypes from the deleted chromosome arms calling only a few SNPs. Figure (E) shows the logarithmic ratio of observed vs. expected signal intensities along the chromosome. It confirms that the area of the chromosomal loss has lower signal (negative logR).

8.5.2 Methodology

We performed high precise genotype filtering using GenCall and RF-GDA (SureTypeSC) of 46 cells from GM12878 (Section 5.2). Both algorithms were adjusted for high recall (GenCall threshold 0.87, SureTypeSC threshold 0.75). To explore the heterogeneity in terms of concordance with the reference bulk DNA genome, we coded variant genotypes that corresponded to the reference with one and variants not matching the reference with zero. Similarly to Zafar et al. (2016), we imputed the missing genotypes with value 0.5 and used Hamming distance to calculate the pairwise dissimilarity between the 46 cells in GM12878. We then performed hierarchical clustering on all SNPs that contained at least one non-reference variant across the 46 cells. We included raw,non-filtered genotypes (QC001), genotypes from GenCall (0.87), and SureTypeSC (0.75) and used Ward distance for hierarchical clustering. We assessed stability of the clusters by performing bootstrap analysis implemented in the fpc package in R. We subsequently labelled the clusters with clusterwise Jaccard bootstrap mean (calculated from 1000 replicates) indicating the stability of the cluster (Figure 8.5).

8.5.3 Results

The hierarchical clustering reveals there are potentially four subpopulations of cells in GM12878 cell line that are invariant to the type of filtration used (Figure 8.5B, C, D). The bootstrap analysis, however, reveals that only the RF-GDA consistently gives four stable subpopulations (Jaccard mean bootstrap value for a cluster > 0.75, Hennig 2007).

The unstable clusters present in the trees from the minimal filter (QC 0.01) and 'high precision' genotyping using GenCall suggest non-reproducible noise being transferred to the bootstrapped replicates that is removed by SureTypeSC. Using a high precision mode, SureTypeSC, but not GenCall, was able to stably detect four subpopulations in the reference GM12878 cell line. Thus, SureTypeSC most likely revealed true heterogeneity within the single-cell population (Vogel et al., 2019).

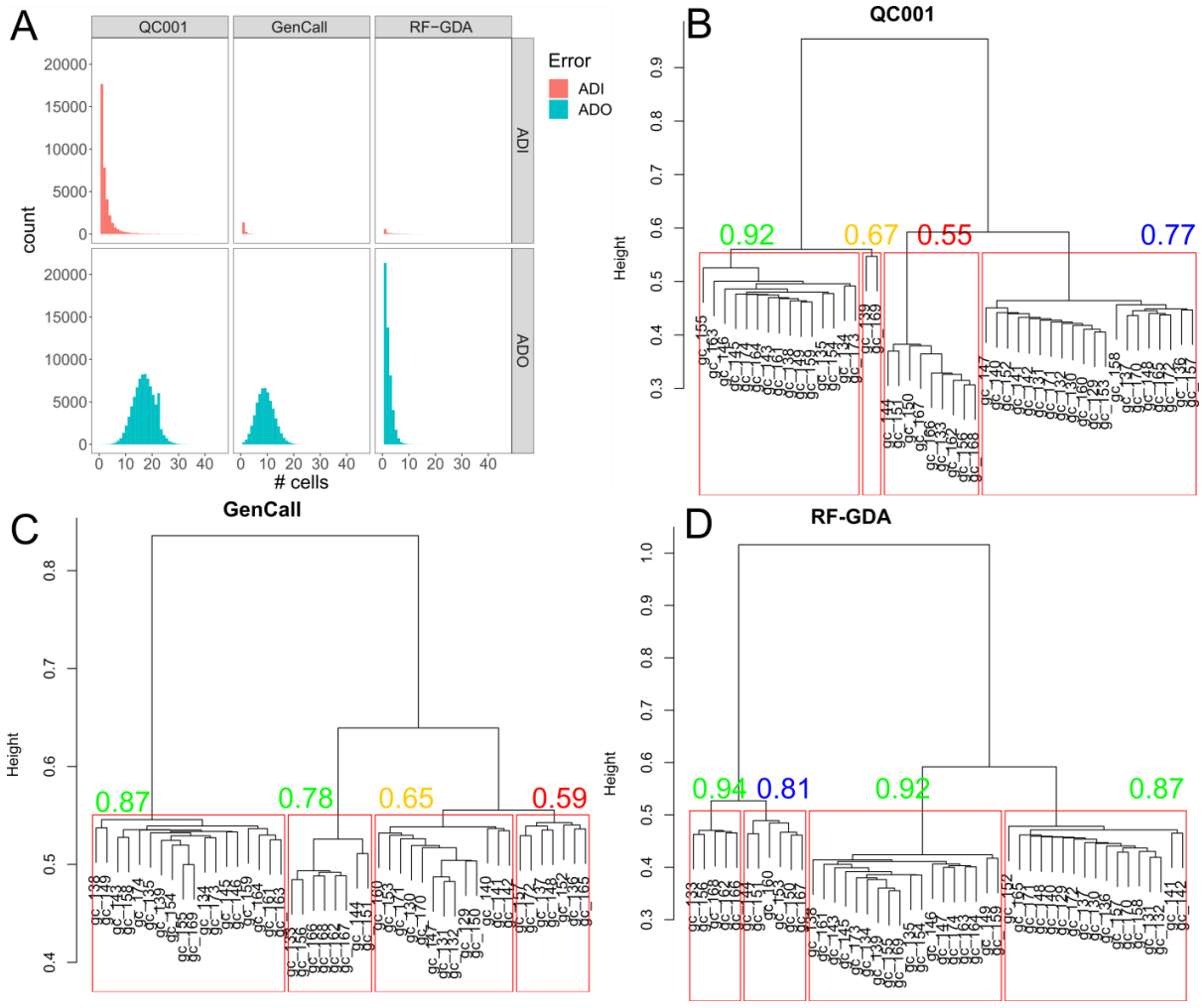

Figure 8.5. Detected errors and potential variants in the 46 cells of GM12878. (A) Histogram of detected ADI and ADO in raw data (left panel), data filtered with GenCall at high precision (GenCall score 0.87, middle panel) and data filtered with SureTypeSC (right panel). Hierarchical clustering on raw data (B), on data from GenCall (C) and from RF-GDA (SureTypeSC, D). The histograms were generated in R using hclust and evaluated using the clusterboot function from the R package *fpc*. The red rectangles show the potential subpopulations and the numbers indicate the Jaccard bootstrap mean. Labels are coloured according to the following rules (obtained from Henning 2007 and manual of the *fpc* package): Green labels indicate highly stable clusters (>0.85), blue labels indicate stable clusters (>0.75); orange labels might indicate a pattern, however the membership of the cells to a particular cluster is doubtful; red labels (<0.6) indicate unstable clusters. RF-GDA is the only algorithm that gives four stable clusters.

8.6 Conclusion

In this chapter we demonstrated the benefits of using SureTypeSC with single-cell data in connection to the knowledge algorithms explained previously and few other examples from Vogel et al. (2019). We show that SureTypeSC effectively filters out bias and thereby refines the detection of biological phenomenon (crossovers, Section 8.2) or allows its detection that was previously note possible due to low sensitivity of the state of the art algorithms (gene conversion and copy number variants; Section 8.3 and 8.4). We also demonstrated potential application of SureTypeSC for clustering of subpopulations in the single cell data. While some results presented in this chapter are supported by statistical testing (bootstrapping of subpopulations, Section 8.5) or inner controls (low logR values for the chromosomal deletion; Section 8.4), other require robust validation on bigger datasets (crossover detection; Section 8.2) .

9 Conclusion

The rapid evolution of single cell genomics is constantly reinforced by hundreds of exciting discoveries about how our body works on a fine level of single cell. A lot of knowledge from various fields of biology is required to explain how cancer cells develop, how cells in our brain communicate or which factors contribute to successful pregnancy. Precise genetic information about a cell is one of the pieces to the puzzle.

Accurate detection of genotype of a single cell is still challenging due to preciousness of the genetic material. While the laboratory techniques are trying to improve the methodology to deliver more accurate signal, it is a matter of fact that the whole genome amplification is the bottleneck of the whole process.

This work targets an important issue of single cell bioinformatics – how to reliably distinguish between a genotype artefact and real signal in the single-cell data. While this problem has been solved algorithmically for NGS data by various approaches, the SNP array technology has been left behind. This is unfortunate, because SNP array offers a cost-efficient alternative for analysing thousands of genomic loci with good coverage, which is confirmed by accuracy and call rate over 99% for standard, bulk DNA.

In this work, a reference population of single cell samples from SNP array was gathered and the noise that is coming through when using the state of the art genotyping workflow for SNP arrays was analysed. A cascade machine learning method that learns the pattern of noise in the single cell data was developed. Thorough validation of the method reveals that it is possible to recall more single-cell genotypes with better precision than with the current state of the art represented by the GenCall algorithm.

Furthermore, algorithms for knowledge extraction from products of female meiosis were designed and optimized, particularly for crossover detection. The improved genotype detection has direct impact on quality of the knowledge gathered from the data. It is likely that the presented algorithms can improve both, diagnostics and biological inference.

SureTypeSC was tested on a reference population of single cells. As these cells were clones, one would expect that the only source of heterogeneity is random noise. It however turns out, that after noise removal, the cells still contain differences and create subpopulations. This confirms what mentioned at the very beginning of this work – heterogeneity is everywhere within our body. Ability of deciphering of subpopulations in the single cell data would have a likely application in cancer biology.

While in the core of this work, I mainly focused on the SNP array technology, it is important to mention that NGS remains to have an important role for this thesis namely as a validation tool. It is common practice in life sciences to acquire information with various technologies and then create

consensus (or reject the hypothesis). This was also the situation when confirming the reference genotype for the training dataset, or, more importantly, for biological inference - in case of gene conversions – where I found feasible candidates using SureTypeSC and then partly confirmed the hypothesis with NGS data.

9.1 Summary of contribution

As mentioned at the beginning, this work was motivated by research interests of two scientific groups that I was member of during my doctoral studies. These groups supported my bioinformatics research with databases of biological data. The biological conclusions were inferred based on evidence found by methods presented in this work. Related to research questions asked in Section 1.2.1, I summarize the contribution of this thesis in the following points:

- I generated a reference population of single cells from SNP arrays with a reliable genotype confirmed by information from bulk DNA. This is a valuable resource for future research serving as a training dataset
- I identified pattern of noise in the data and based on this, I designed, implemented and validated an original two stage machine learning method named SureTypeSC. The method evaluates a single cell genotype from SNP array and calculates a confidence measure. The algorithm outperforms current state of the art in genotyping of single-cell SNP array data
- I designed and implemented algorithms for crossover detection in single-cell data from female meiosis. I demonstrated application and benefits of using SureTypeSC with these algorithms. I furthermore showed how SureTypeSC can help to solve other relevant questions of single-cell bioinformatics: aneuploidy detection, subpopulations clustering and gene conversion analysis.

9.2 Future perspectives

The functionality of the presented filtering algorithm for single-cell SNP array data could be likely shifted from detection to imputation. While currently it is not the goal to correct the erroneous genotypes and these are simply rejected, in future, they could be perhaps imputed by using the parameters from the second layer of the model. This would likely lead to a multiclass classification problem.

While Gaussian mixture model is used as a central structure of the second layer of the algorithm, it would be perhaps beneficial to experiment with probability distributions other than Gaussian. Multivariate truncated *t* distribution previously used in genotyping (Teo et al., 2008) is one of the candidates that could potentially deal with presence of outliers in the data (Vogel et al., 2019). The other possibility would be to employ variational Bayesian inference over GMM (Section 4.3.4).

While preliminary results do not show any benefits of this method on the presented single-cell data, more comprehensive experiments with the parameters could bring improvement over the current method. The motivation here is the central limit theorem and infinite number of Gaussian components that the variational inference allows.

The algorithms for crossover inference certainly require validation on more comprehensive datasets. More importantly, prior knowledge about positions of crossovers from population studies (Kong et al., 2010) could motivate development of Hidden Markov Model over the data. This, in combination with SureTypeSC, could likely improve the crossover inference by eliminating the false positive events detected in the data.

Bibliography

Alberts, B., Bray, D., Hopkin, K., Johnson, A.D., Lewis, J., Raff, M., Roberts, K., and Walter, P. (2015). Essential Cell Biology (Garland Science).

Altschul, S.F., Gish, W., Miller, W., Myers, E.W., and Lipman, D.J. (1990). Basic local alignment search tool. J. Mol. Biol. 215, 403–410.

Altschuler, S.J., and Wu, L.F. (2010). Cellular Heterogeneity: Do Differences Make a Difference? Cell 141, 559–563.

Bamberg, P. (1991). A Course in Mathematics for Students of Physics 1 (Cambridge - New York etc: Cambridge University Press).

Behjati, S., and Tarpey, P.S. (2013). What is next generation sequencing? Arch Dis Child Educ Pract Ed 98, 236–238.

Bishop, C.M. (2006). Pattern Recognition and Machine Learning (Information Science and Statistics) (Berlin, Heidelberg: Springer-Verlag).

Blagodatskikh, K.A., Kramarov, V.M., Barsova, E.V., Garkovenko, A.V., Shcherbo, D.S., Shelenkov, A.A., Ustinova, V.V., Tokarenko, M.R., Baker, S.C., Kramarova, T.V., et al. (2017). Improved DOP-PCR (iDOP-PCR): A robust and simple WGA method for efficient amplification of low copy number genomic DNA. PLOS ONE 12, e0184507.

Bland, J.M., and Altman, D.G. (1999). Measuring agreement in method comparison studies. Stat Methods Med Res 8, 135–160.

Blanshard, R.C., Chen, C., Xie, X.S., and Hoffmann, E.R. (2018). Chapter 20 - Single cell genomics to study DNA and chromosome changes in human gametes and embryos. In Methods in Cell Biology, H. Maiato, and M. Schuh, eds. (Academic Press), pp. 441–457.

Blei, D.M., and Jordan, M.I. (2006). Variational inference for Dirichlet process mixtures. Bayesian Analysis 1, 121–143.

Bolstad, B.M., Irizarry, R.A., Astrand, M., and Speed, T.P. (2003). A comparison of normalization methods for high density oligonucleotide array data based on variance and bias. Bioinformatics 19, 185–193.

Bourcy, C.F.A. de, Vlaminck, I.D., Kanbar, J.N., Wang, J., Gawad, C., and Quake, S.R. (2014). A Quantitative Comparison of Single-Cell Whole Genome Amplification Methods. PLOS ONE 9, e105585.

Bovik, A.C., Huang, T.S., and Munson, D.C. (1987). The Effect of Median Filtering on Edge Estimation and Detection. IEEE Transactions on Pattern Analysis and Machine Intelligence PAMI-9, 181–194.

Breiman, L. (2001). Random Forests. Mach. Learn. 45, 5–32.

Carvalho, B., Bengtsson, H., Speed, T.P., and Irizarry, R.A. (2007). Exploration, normalization, and genotype calls of high-density oligonucleotide SNP array data. Biostatistics 8, 485–499.

Chao, K.-M., and Zhang, L. (2009). Sequence Comparison: Theory and Methods (London: Springer-Verlag).

Chen, C., Xing, D., Tan, L., Li, H., Zhou, G., Huang, L., and Xie, X.S. (2017). Single-cell whole-genome analyses by Linear Amplification via Transposon Insertion (LIANTI). Science 356, 189–194.

Chen, J.-M., Cooper, D.N., Chuzhanova, N., Férec, C., and Patrinos, G.P. (2007). Gene conversion: mechanisms, evolution and human disease. Nature Reviews Genetics 8, 762–775.

Collins, F.S., Brooks, L.D., and Chakravarti, A. (1998). A DNA Polymorphism Discovery Resource for Research on Human Genetic Variation. Genome Res. 8, 1229–1231.

Dempster, A.P., Laird, N.M., and Rubin, D.B. (1977). Maximum likelihood from incomplete data via the EM algorithm. J. Roy. Statist. Soc. Ser. B 39, 1–38.

Dhaliwal, A. (2019). DNA Extraction and Purification. Materials and Methods.

Eberle, M.A., Fritzilas, E., Krusche, P., Källberg, M., Moore, B.L., Bekritsky, M.A., Iqbal, Z., Chuang, H.-Y., Humphray, S.J., Halpern, A.L., et al. (2017). A reference data set of 5.4 million phased human variants validated by genetic inheritance from sequencing a three-generation 17-member pedigree. Genome Res 27, 157–164.

Emery, B., and Barres, B.A. (2008). Unlocking CNS Cell Type Heterogeneity. Cell 135, 596–598.

Feuk, L., Carson, A.R., and Scherer, S.W. (2006). Structural variation in the human genome. Nature Reviews Genetics 7, 85–97.

Friedler, S. (2012). In Vitro Fertilization: Innovative Clinical and Laboratory Aspects (BoD – Books on Demand).

Garibyan, L., and Avashia, N. (2013). Research Techniques Made Simple: Polymerase Chain Reaction (PCR). J Invest Dermatol 133, e6.

Genton, M.G. (2002). Classes of Kernels for Machine Learning: A Statistics Perspective. J. Mach. Learn. Res. 2, 299–312.

Giannoulatou, E., Yau, C., Colella, S., Ragoussis, J., and Holmes, C.C. (2008). GenoSNP: a variational Bayes within-sample SNP genotyping algorithm that does not require a reference population. Bioinformatics 24, 2209–2214.

Han, J., Kamber, M., and Pei, J. (2011). Data Mining: Concepts and Techniques, Third Edition (Haryana, India; Burlington, MA: Morgan Kaufmann).

Handyside, A.H., Lesko, J.G., Tarín, J.J., Winston, R.M., and Hughes, M.R. (1992). Birth of a normal girl after in vitro fertilization and preimplantation diagnostic testing for cystic fibrosis. N. Engl. J. Med. 327, 905–909.

Handyside, A.H., Harton, G.L., Mariani, B., Thornhill, A.R., Affara, N., Shaw, M.-A., and Griffin, D.K. (2010). Karyomapping: a universal method for genome wide analysis of genetic disease based on mapping crossovers between parental haplotypes. J. Med. Genet. 47, 651–658.

Hartl, D.L. (2012). Essential Genetics: A Genomics Perspective (Burlington: Jones & Bartlett Learning).

Hassold, T., and Hunt, P. (2001). To err (meiotically) is human: the genesis of human aneuploidy. Nature Reviews Genetics 2, 280.

Hastie, T., Tibshirani, R., and Friedman, J. (2016). The Elements of Statistical Learning: Data Mining, Inference, and Prediction, Second Edition (New York, NY: Springer).

Hennig, C. (2007). Cluster-wise assessment of cluster stability. COMPUT STAT DATA AN 52, 258–271.

Hornik, K., Stinchcombe, M., and White, H. (1989). Multilayer feedforward networks are universal approximators. Neural Networks 2, 359–366.

Hu, P., Zhang, W., Xin, H., and Deng, G. (2016). Single Cell Isolation and Analysis. Front Cell Dev Biol 4.

Illumina, Inc. (2010). The Power of Intelligent SNP Selection. Technical Note: DNA Analysis.

Illumina, Inc. (2014). Infinium Genotyping Data Analysis. Technical Note: Genotyping.

International HapMap Consortium (2003). The International HapMap Project. Nature 426, 789–796.

International HapMap Consortium (2010). Integrating common and rare genetic variation in diverse human populations. Nature 467, 52–58.

Junker, J.P., and van Oudenaarden, A. (2014). Every Cell Is Special: Genome-wide Studies Add a New Dimension to Single-Cell Biology. Cell 157, 8–11.

Kearns, M., Labs, T., and Mansour, Y. A Fast, Bottom-Up Decision Tree Pruning Algorithm with Near-Optimal Generalization. 16.

Kent, W.J. (2002). BLAT--the BLAST-like alignment tool. Genome Res. 12, 656–664.

Kermani, B.G. (2008). Artificial intelligence and global normalization methods for genotyping.

Kingsford, C., and Salzberg, S.L. (2008). What are decision trees? Nat. Biotechnol. 26, 1011–1013.

Kong, A., Thorleifsson, G., Gudbjartsson, D.F., Masson, G., Sigurdsson, A., Jonasdottir, A., Walters, G.B., Jonasdottir, A., Gylfason, A., Kristinsson, K.T., et al. (2010). Fine-scale recombination rate differences between sexes, populations and individuals. Nature 467, 1099–1103.

Krane, D.E. (2002). Fundamental Concepts of Bioinformatics (Pearson Education).

Krishnamoorthy, K. (2006). Handbook of Statistical Distributions with Applications (Boca Raton: Chapman and Hall/CRC).

Kubat, Z., Zluvova, J., Vogel, I., Kovacova, V., Cermak, T., Cegan, R., Hobza, R., Vyskot, B., and Kejnovsky, E. (2014). Possible mechanisms responsible for absence of a retrotransposon family on a plant Y chromosome. New Phytol. 202, 662–678.

LaFramboise, T. (2009). Single nucleotide polymorphism arrays: a decade of biological, computational and technological advances. Nucleic Acids Res 37, 4181–4193.

Lamy, P., Grove, J., and Wiuf, C. (2011). A review of software for microarray genotyping. Hum Genomics 5, 304–309.

Langmead, B., Trapnell, C., Pop, M., and Salzberg, S.L. (2009). Ultrafast and memory-efficient alignment of short DNA sequences to the human genome. Genome Biology 10, R25.

Li, H., and Durbin, R. (2009). Fast and accurate short read alignment with Burrows–Wheeler transform. Bioinformatics 25, 1754.

Mago, V.K., and Bhatia, N. (2012). Cross-disciplinary Applications of Artificial Intelligence and Pattern Recognition: Advancing Technologies (Information Science Reference).

Martin, C.L., Kirkpatrick, B.E., and Ledbetter, D.H. (2015). CNVs, Aneuploidies and Human Disease. Clin Perinatol 42, 227–242.

McKinney, W. (2010). Data structures for statistical computing in python. Proceedings of the 9th Python in Science Conference 51–56.

McLeod, C.M., and Mauck, R.L. (2017). On the origin and impact of mesenchymal stem cell heterogeneity: new insights and emerging tools for single cell analysis. Eur Cell Mater 34, 217–231.

Montpetit, A., Nelis, M., Laflamme, P., Magi, R., Ke, X., Remm, M., Cardon, L., Hudson, T.J., and Metspalu, A. (2006). An Evaluation of the Performance of Tag SNPs Derived from HapMap in a Caucasian Population. PLOS Genetics 2, e27.

Murphy, K.P. (2012). Machine Learning: A Probabilistic Perspective (Cambridge, MA: The MIT Press).

Needleman, S.B., and Wunsch, C.D. (1970). A general method applicable to the search for similarities in the amino acid sequence of two proteins. Journal of Molecular Biology 48, 443–453.

Ng, A. (2019). CS229 Lecture Notes, PartIV: Generative Learning algorithms.

Novák, P., Neumann, P., and Macas, J. (2010). Graph-based clustering and characterization of repetitive sequences in next-generation sequencing data. BMC Bioinformatics 11, 378.

Orlova, T. (2010). Matematical models, algorithms and statistics of sequence alignment. University of South Carolina.

Oshiro, T.M., Perez, P.S., and Baranauskas, J.A. (2012). How Many Trees in a Random Forest? In Machine Learning and Data Mining in Pattern Recognition, (Springer, Berlin, Heidelberg), pp. 154–168.

Ottolini, C.S., Newnham, L., Capalbo, A., Natesan, S.A., Joshi, H.A., Cimadomo, D., Griffin, D.K., Sage, K., Summers, M.C., Thornhill, A.R., et al. (2015). Genome-wide maps of recombination and chromosome segregation in human oocytes and embryos show selection for maternal recombination rates. Nat Genet 47, 727–735.

Ottolini, C.S., Capalbo, A., Newnham, L., Cimadomo, D., Natesan, S.A., Hoffmann, E.R., Ubaldi, F.M., Rienzi, L., and Handyside, A.H. (2016). Generation of meiomaps of genome-wide recombination and chromosome segregation in human oocytes. Nat Protoc 11, 1229–1243.

Padhukasahasram, B., and Rannala, B. (2013). Meiotic gene-conversion rate and tract length variation in the human genome. European Journal of Human Genetics.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., et al. (2011). Scikit-learn: Machine Learning in Python. J. Mach. Learn. Res. 12, 2825–2830.

Peel, D., and McLachlan, G.J. (2000). Robust mixture modelling using the t distribution. Statistics and Computing 10, 339–348.

Prompramote, S., Chen, Y., and Chen, Y.-P.P. (2005). Machine Learning in Bioinformatics. In Bioinformatics Technologies, Y.-P.P. Chen, ed. (Berlin, Heidelberg: Springer Berlin Heidelberg), pp. 117–153.

Psifidi, A., Dovas, C.I., Bramis, G., Lazou, T., Russel, C.L., Arsenos, G., and Banos, G. (2015). Comparison of Eleven Methods for Genomic DNA Extraction Suitable for Large-Scale Whole-Genome Genotyping and Long-Term DNA Banking Using Blood Samples. PLOS ONE 10, e0115960.

Rieger, R., Michaelis, A., and Green, M.M. (1968). A Glossary of Genetics and Cytogenetics: Classical and Molecular (Berlin Heidelberg: Springer-Verlag).

Ritchie, M.E., Liu, R., Carvalho, B.S., Irizarry, R.A., and The Australia and New Zealand Multiple Sclerosis Genetics Consortium (ANZgene) (2011). Comparing genotyping algorithms for Illumina's Infinium whole-genome SNP BeadChips. BMC Bioinformatics 12, 68.

Rizzo, J.M., and Buck, M.J. (2012). Key Principles and Clinical Applications of "Next-Generation" DNA Sequencing. Cancer Prev Res 5, 887–900.

Saito, T., Rehmsmeier, M., and Wren, J. (2017). Precrec: fast and accurate precision–recall and ROC curve calculations in R. Bioinformatics 33, 145–147.

Serratì, S., Summa, S.D., Pilato, B., Petriella, D., Lacalamita, R., Tommasi, S., and Pinto, R. (2016). Next-generation sequencing: advances and applications in cancer diagnosis.

Shapiro, E., Biezuner, T., and Linnarsson, S. (2013). Single-cell sequencing-based technologies will revolutionize whole-organism science. Nature Reviews Genetics 14, 618–630.

Sims, D., Sudbery, I., Ilott, N.E., Heger, A., and Ponting, C.P. (2014). Sequencing depth and coverage: key considerations in genomic analyses. Nature Reviews Genetics 15, 121–132.

Slatko, B.E., Gardner, A.F., and Ausubel, F.M. (2018). Overview of Next-Generation Sequencing Technologies. Current Protocols in Molecular Biology 122, e59.

Smith, T.F., and Waterman, M.S. (1981). Identification of common molecular subsequences. J. Mol. Biol. 147, 195–197.

Spits, C., Le Caignec, C., De Rycke, M., Van Haute, L., Van Steirteghem, A., Liebaers, I., and Sermon, K. (2006). Optimization and evaluation of single-cell whole-genome multiple displacement amplification. Hum. Mutat. 27, 496–503.

Staaf, J., Vallon-Christersson, J., Lindgren, D., Juliusson, G., Rosenquist, R., Höglund, M., Borg, Å., and Ringnér, M. (2008). Normalization of Illumina Infinium whole-genome SNP data improves copy number estimates and allelic intensity ratios. BMC Bioinformatics 9, 409.

Steflova, P., Tokan, V., Vogel, I., Lexa, M., Macas, J., Novak, P., Hobza, R., Vyskot, B., and Kejnovsky, E. (2013). Contrasting Patterns of Transposable Element and Satellite Distribution on Sex Chromosomes (XY1Y2) in the Dioecious Plant Rumex acetosa. Genome Biol Evol 5, 769–782.

Teh, Y.W. (2010). Dirichlet Process. In Encyclopedia of Machine Learning, C. Sammut, and G.I. Webb, eds. (Boston, MA: Springer US), pp. 280–287.

Teo, Y.Y., Inouye, M., Small, K.S., Gwilliam, R., Deloukas, P., Kwiatkowski, D.P., and Clark, T.G. (2007). A genotype calling algorithm for the Illumina BeadArray platform. Bioinformatics 23, 2741–2746.

Thorisson, G.A., Smith, A.V., Krishnan, L., and Stein, L.D. (2005). The International HapMap Project Web site. Genome Res 15, 1592–1593.

Van der Auwera, G.A., Carneiro, M.O., Hartl, C., Poplin, R., Del Angel, G., Levy-Moonshine, A., Jordan, T., Shakir, K., Roazen, D., Thibault, J., et al. (2013). From FastQ data to high confidence variant calls: the Genome Analysis Toolkit best practices pipeline. Curr Protoc Bioinformatics 43, 11.10.1-33.

Vanneste, E., Bittman, L., Van der Aa, N., Voet, T., and Vermeesch, J.R. (2012). New Array Approaches to Explore Single Cells Genomes. Front Genet 3.

Vogel, I., Blanshard, R.C., and Hoffmann, E.R. SureTypeSC—a Random Forest and Gaussian mixture predictor of high confidence genotypes in single-cell data. Bioinformatics.

Walt, S. van der, Colbert, S.C., and Varoquaux, G. (2011). The NumPy Array: A Structure for Efficient Numerical Computation. Computing in Science Engineering 13, 22–30.

Williams, A.L., Genovese, G., Dyer, T., Altemose, N., Truax, K., Jun, G., Patterson, N., Myers, S.R., Curran, J.E., Duggirala, R., et al. (2015). Non-crossover gene conversions show strong GC bias and unexpected clustering in humans. ELife 4, e04637.

Wright, M.N., Gola, D., and Ziegler, A. (2017). Preprocessing and Quality Control for Whole-Genome Sequences from the Illumina HiSeq X Platform. In Statistical Human Genetics: Methods and Protocols, R.C. Elston, ed. (New York, NY: Springer New York), pp. 629–647.

Wu, J., Xiao, J., Zhang, Z., Wang, X., Hu, S., and Yu, J. (2014). Ribogenomics: the Science and Knowledge of RNA. Genomics Proteomics Bioinformatics 12, 57–63.

Yu, K., Dang, X., Bart, H., and Chen, Y. (2015). Robust Model-Based Learning via Spatial-EM Algorithm. IEEE Transactions on Knowledge and Data Engineering 27, 1670–1682.

Zafar, H., Wang, Y., Nakhleh, L., Navin, N., and Chen, K. (2016). Monovar: single-nucleotide variant detection in single cells. Nat. Methods 13, 505–507.

Zamani Esteki, M., Dimitriadou, E., Mateiu, L., Melotte, C., Van der Aa, N., Kumar, P., Das, R., Theunis, K., Cheng, J., Legius, E., et al. (2015). Concurrent Whole-Genome Haplotyping and Copy-Number Profiling of Single Cells. Am J Hum Genet 96, 894–912.

Publications

Publications important for the PhD thesis

Published:

Vogel, I., Blanshard, R.C., and Hoffmann, E.R. SureTypeSC—a Random Forest and Gaussian mixture predictor of high confidence genotypes in single-cell data. Bioinformatics.

- I designed and implemented the method (SureTypeSC) and performed validation on reference and experiments on unseen data. My contribution was 90%, the co-authors provided the raw data (performed the laboratory part of the workflow) and drew the biological conclusions on the results.

Kubat, Z., Zluvova, J., Vogel, I., Kovacova, V., Cermak, T., Cegan, R., Hobza, R., Vyskot, B., and Kejnovsky, E. (2014). Possible mechanisms responsible for absence of a retrotransposon family on a plant Y chromosome. New Phytol. *202*, 662–678.

- I designed workflow for processing the NGS data, particularly small RNAs and mRNAs in relation to transposons. The results explain their regulation, transcription and proliferation. My contribution to this study was 30%.

Steflova, P., Tokan, V., Vogel, I., Lexa, M., Macas, J., Novak, P., Hobza, R., Vyskot, B., and Kejnovsky, E. (2013). Contrasting Patterns of Transposable Element and Satellite Distribution on Sex Chromosomes (XY1Y2) in the Dioecious Plant Rumex acetosa. Genome Biol Evol *5*, 769–782.

- I designed workflow for processing genomic clusters of transposons created from NGS and created a pipeline to analyse their expression and to refine their abundance. My contribution to this study was 30%.

In revision:

Gruhn, J., Zielinska, A., Shukla, V., Blanshard, R., Capalbo, A., Cimadomo, D., Nikiforov, D., Chan, A., Newnham, L., Vogel, I., et al. Chromosome errors in human eggs shape natural fertility. (submitted to Science).

- SureTypeSC was used to analyse the SNP array data from this study to clean up the genotypes and draw biological conclusions. My contribution to this study was 10%.

Other publications:

Larsen, N.B., Liberti, S.E., Vogel, I., Jørgensen, S.W., Hickson, I.D., and Mankouri, H.W. (2017). Stalled replication forks generate a distinct mutational signature in yeast. Proc. Natl. Acad. Sci. U.S.A. *114*, 9665–9670.

Kralova, T., Cegan, R., Kubat, Z., Vrana, J., Vyskot, B., Vogel, I., Kejnovsky, E., and Hobza, R. (2014). Identification of a novel retrotransposon with sex chromosome-specific distribution in Silene latifolia. Cytogenet. Genome Res. *143*, 87–95.

Ren, L., Chen, L., Wu, W., Garribba, L., Tian, H., Liu, Z., Vogel, I., Li, C., Hickson, I.D., and Liu, Y. (2017). Potential biomarkers of DNA replication stress in cancer. Oncotarget *8*, 36996–37008.

Soukupova, M., Nevrtalova, E., Cížková, J., Vogel, I., Cegan, R., Hobza, R., and Vyskot, B. (2014). The X chromosome is necessary for somatic development in the dioecious Silene latifolia: cytogenetic and molecular evidence and sequencing of a haploid genome. Cytogenet. Genome Res. *143*, 96–103.