Addressing NGS Data Challenges: Efficient High Throughput Processing and Sequencing Error Detection

Inaugural-Dissertation

zur

Erlangung des Doktorgrades

der Mathematisch-Naturwissenschaftlichen Fakultät

der Universität zu Köln

vorgelegt von

Amit Kawalia

aus Para Bass Suila, Alwar, Rajasthan (India)

Köln, 2016

Berichterstatter: Prof. Dr. Peter Nürnberg

(Gutachter) Prof. Dr. Michael Nothnagel

Prüfungsvorsitzender: Prof. Dr. Hartmut Arndt

Beisitzer: Dr. med. Holger Thiele

Tag der mündlichen Prüfung: 18/01/2016

Abstract

Next generation sequencing (NGS) technologies have facilitated the identification of disease causing mutations, which has significantly improved patient's diagnosis and treatment. Since its emergence, NGS has been used in many applications like genome sequencing, DNA resequencing, transcriptome sequencing and epigenomics, to unfold the various layers of genome biology. Because of this broad spectrum of applications and recent decrement in cost, usage of NGS has become a routine approach to address many research as well as medical questions. It is producing huge amounts of data, which necessitate highly efficient and accurate computational analysis as well as data management.

This thesis addresses some of the challenges of NGS data analysis, mainly for targeted DNA sequencing data. It describes the various steps required for data analysis including their significance and potential negative effects on consecutive downstream analysis and so on the final variant lists. In order to make the analysis more accurate and efficient, an extensive testing of different bioinformatics tools and algorithms was preformed and a fully automated data analysis workflow was developed. This workflow is implemented and optimized on high performance computing (HPC) systems. I describe different design principles and parallelization strategies that enable proper exploitation of HPC resources to achieve high throughput of data analysis. Besides correcting for known sequencing errors by using existing tools, this work is also aimed at the detection of a new class of systematic sequencing errors called recurrent systematic sequencing errors. I present an approach for the exploration of this class of errors and describe the probable causes and patterns behind them. This includes some known and novel patterns observed during this work. Furthermore, I provide a tool to filter the false variants due to these errors from any variant list. Overall, the work performed during this thesis has been already used (and will be used in future as well), to provide accurate and efficient data analysis, which enables exploration of the genetic background of various diseases.

Zusammenfassung

Die Next-Generation-Sequencing-(NGS)-Technologien haben die Identifizierung krankheitsverursachender Mutationen erleichtert, wodurch die Diagnose und Behandlung von Patienten deutlich verbessert wurde. Seit seiner Einführung wird NGS in vielen Anwendungsbereichen, wie Genom-Sequenzierung, DNA-Resequenzierung, Transkriptom-Sequenzierung und Epigenomik, eingesetzt, um die verschiedenen Ebenen der Biologie des Genoms zu entschlüsseln. Aufgrund dieses breiten Anwendungsspektrums und der aktuellen Kostensenkung ist die Verwendung von NGS zu einem Routineverfahren zur Bearbeitung vieler forschungsbezogener und medizinischer Fragestellungen geworden. Dadurch werden große Datenmengen erzeugt, die hoch effiziente und exakte computergestützte Analysen sowie ein entsprechendes Datenmanagement notwendig machen.

Diese Dissertation widmet sich einigen der mit der NGS-Datenanalyse verbundenen Herausforderungen, vor allem in Bezug auf die gezielte DNA-Sequenzierung ausgewählter genomischer Bereiche („targeted sequencing“ genannt). Sie beschreibt die verschiedenen für die Datenanalyse erforderlichen Schritte, ihre Bedeutung und potentiellen negativen Effekte auf anschließende Folgeanalysen und damit auf die finalen Variantenlisten. Um die Analyse exakter und effizienter zu machen, wurden umfassende Tests verschiedener bioinformatischer Tools und Algorithmen durchgeführt und ein vollautomatischer Analyse-Workflow entwickelt. Dieser Workflow ist auf Hochleistungsrechensystemen (HPC Systemen) implementiert und für diese optimiert worden. Ich beschreibe verschiedene Entwurfsprinzipien und Parallelisierungsstrategien, um eine gute Nutzung der Ressourcen eines HPC-Systems und hohen Durchsatz in der Datenanalyse zu erreichen. Neben der Korrektur bekannter Sequenzierungsfehler durch vorhandene Tools, widmet sich diese Arbeit auch der Detektion einer neuen Klasse systematischer Sequenzierungsfehler, „wiederkehrende systematische Fehler“ genannt. Ich präsentiere ein neues Verfahren, um diese Fehlerklasse zu untersuchen und beschreibe die ihr wahrscheinlich zugrundeliegenden Ursachen und Muster. Dabei beobachtete ich einige bekannte und neue Muster. Weiterhin stelle ich ein Tool zur Verfügung, um von diesen Fehlern verursachte falsche Varianten aus beliebigen Variantenlisten zu filtern. Die während dieser Doktorarbeit durchgeführten und hier präsentierten Arbeiten wurden bereits (und werden weiterhin) verwendet, um exakte und effiziente Datenanalyse durchzuführen, die die Erforschung des genetischen Hintergrundes verschiedenster Krankheiten ermöglicht.

Acknowledgement

I am very grateful to my supervisors Peter Nuernberg and Susanne Motameny who have given me an opportunity to work in the very exciting and challenging field of Next Generation Sequencing during this thesis. Special thanks to Susanne Motameny for extensive proofreading, moral support and guidance throughout my thesis. I also want to thank Holger Thiele who involved me in some other interesting projects and also supported me during the whole period. I would also like to thank Michael Nothnagel for his guidance, suggestions and for reviewing this work. Furthermore, I would like to thank Kamel Jabbari and Wilfried Gunia for their guidance and many fruitful conversations, both scientific and non-scientific. It has been a great pleasure to work with all of them. Without their support and guidance throughout my work, this thesis would not have been accomplished.

I am also thankful to all other colleagues and my friends for their assistance and support. I am also grateful to my parents and brother who believed in me and supported me throughout my career. I cannot thank my wife enough who patiently supported and helped me during my "low days" and tolerated my craziness.

Table of Contents

List of Figures

List of Tables

List of abbreviations

Abbreviation	Term
AT	Adenine Thymine
BAM	Binary version of SAM
BP	Base pair
BQSR	Base Quality Score Recalibration
BWA	Burrows-Wheeler Alignment
CHR	Chromosome
CNV	Copy Number Variation
CPU	Central processing unit
DNA	Deoxyribonucleic acid
DP	Read depth
FDR	False Discovery Rate
FN	False negative
FP	False positive
FPR	False positive rate
GATK	Genome Analysis Tool Kit
GB	Gigabyte
GC	Guanine Cytosine
HC	Haplotype Caller
HDR	High divergent region
HPC	High Performance Computing
Indel	Insertion or deletion
LCR	Low complexity region
LIMS	Laboratory Information Management System
MAF	Minor allele frequency
MB	Megabyte
MQ	Mapping Quality
MSA	Multiple Sequence Alignment

NGS	Next Generation Sequencing
PCR	Polymerase chain reaction
PM	Prefix Model
PPV	Positive prediction value
QC	Quality Control
RAM	Random-access memory
RSE	Recurrent Systematic Error
SAM	Sequence alignment format
SM	Suffix Model
SNP	Single Nucleotide Polymorphism
SSE	Sequence Specific Error
STR	Short tandem repeats
SV	Structural Variation
TB	Terabyte
TP	True positive
TPR	True positive rate
UG	Unified Genotyper
VCF	Variant Call Format
VQSR	Variant Quality Score Recalibration
WES	Whole Exome Sequencing
WGS	Whole Genome Sequencing
XML	Extensible Markup Language

Chapter 1
Introduction

Exploration of the causal gene variants underlying human diseases is one of the major interests of medical sciences. Recent advances in DNA sequencing technologies have enabled characterization of genomic landscapes of many diseases at significantly lower cost and in less time. The early identification of disease causing mutations has significantly improved patient's diagnosis and treatment/therapy. Nowadays, DNA sequencing has become a routine work to address many research as well as medical questions in order to improve disease management.

The field of DNA sequencing has a very rich and diverse history (Rabbani, Mahdieh, Hosomichi, Nakaoka, & Inoue, 2012). The story of sequencing began, when Sanger's studies of insulin first demonstrated that proteins are composed of linear polypeptides formed by joining amino acid residues (Hutchison, 2007; Sanger & Tuppy, 1951; Sanger, 1949). Shortly afterwards, the double-helical structure of DNA was proposed (Crick & Watson, 1953), which raised the very significant question about DNA decoding (DNA to protein). However, due to the complexity of DNA, it took approximately 12 years to sequence the first gene (Holley et al., 1965).

In 1977, the first method for DNA sequencing through chain termination was developed (Sanger, Nicklen, & Coulson, 1977). This method was quite slow and laborious which prompted it's automation (L. M. Smith et al., 1986). Automated Sanger sequencing became the core technology of the Human Genome Project (HGP), funded in 1990, and produced the first human genome draft sequences (Lander et al., 2001; Venter et al., 2001). The HGP took 13 years to finish and was quite expensive which led to the development of cheaper and faster next generation sequencing (NGS) technologies (Mardis, 2008).

NGS technologies, also known as high-throughput sequencing technologies, parallelize the sequencing process and produce millions of sequences simultaneously at relatively low cost. Taking advantage of these technologies, the 1000 Genomes Project could describe the genomes of 1,092 individuals from 14 populations. Aim of the project was to provide a catalogue of human genomic variation that was achieved by using a combination of low-coverage whole-genome and exome sequencing only in four years (Abecasis et al., 2012; The 1000 Genomes Project Consortium, 2010). After its emergence, NGS has been used in many applications to unfold the various layers of genetics and genome biology. In a research setting, it has been used for de novo genome sequencing, DNA resequencing, transcriptome sequencing and epigenomics. Now, it has also become an asset of clinical diagnostic laboratories for patient care (Coonrod, Margraf, & Voelkerding, 2012).

Since the advent of NGS, there have been lots of improvements in the sequencing technologies like accuracy, read length and throughput. Moreover, the sequencing costs are also rapidly decreasing. Today NGS techniques are easily accessible and have become a preferred method in most of the research and diagnostics centres resulting in huge amount of sequencing data. The actual challenges are for the bioinformatics community to manage and analyse these data accurately and efficiently. In this work, I will address some of these issues that are relevant for targeted DNA sequencing. In the following sections, first I will provide an overview of NGS technologies. Then, I will provide an overview of targeted DNA sequencing methods and their applications in research and diagnostics. At last, I will briefly mention some data analysis terminologies followed by a description of the challenges of NGS data analysis and an overview of the thesis structure.

1.1 NGS technologies

There are many different platforms for massively parallel DNA sequencing: Roche/454[1], Illumina[2], Ion Torrent[3], PacBio RS[4] and Oxford Nanopore[5]. The first three: Illumina, 454

[1] http://454.com/applications/index.asp This and other subsequent URLs are accessed on 26 August 2015.

and Ion Torrent belong to the second generation sequencing technologies and are the most commonly used platforms in research and clinical labs today. The basic sequencing method of all of these platforms is sequencing by synthesis (SBS) (Hyman, 1988; Turcatti, Romieu, Fedurco, & Tairi, 2008). In SBS methods, sequencing starts with a primer attached to the single stranded template DNA and proceeds with incorporation of a nucleotide base using a DNA polymerase. Every incorporated nucleotide is detected and the further extension, by means of the DNA polymerase, finally results in a complementary sequence to the template DNA (Berglund, Kiialainen, & Syvänen, 2011). Although using the same SBS technology, these three platforms differ in the clonal amplification and nucleotide base detection process. Both 454 and Ion torrent use emulsion PCR amplification, whereas Illumina uses bridge amplification for the clonal amplification process (Margulies et al., 2005; Quail et al., 2012; Williams et al., 2006). During the nucleotide base detection process, Illumina detects fluorescent signals emitted from nucleotide incorporation, whereas changes in pH and emission of light are detected by Ion torrent and 454 respectively (Mardis, 2008, 2013). Details of these technologies can be found in (Mardis, 2008, 2013; Metzker, 2010).

PacBio and Nanopore use recent sequencing technology known as third generation sequencing (Schadt, Turner, & Kasarskis, 2010). Both sequencing platforms perform single molecule sequencing in which sequencing is performed on single DNA molecule using single DNA polymerase without prior cloning or amplification step used in SBS methods. These techniques allow for more accurate sequencing in repetitive or low complexity regions (cf. Chapter 2) and are able to generate very long reads than the SBS technologies. However, currently these technologies are as matured as the second generation (e.g. Illumina) technologies, having high raw read error rates and low throughput. Moreover, there is a need to develop new sophisticated data analysis

[2] http://www.illumina.com/systems/sequencing-platform-comparison.html

[3] http://www.lifetechnologies.com/de/de/home/life-science/sequencing/next-generation-sequencing.html

[4] http://www.pacb.com/products-and-services/pacbio-systems/

[5] https://www.nanoporetech.com/

algorithms, as these technologies posses different error profiles and different aspects of information in the generated sequencing data (Schadt et al., 2010). In our institute, we mainly use the Illumina sequencing technology and this work is also based on Illumina generated sequencing data. Thus, in the following section we provide a brief description of Illumina sequencing technology.

1.1.1 Illumina

Illumina is a widely used platform for DNA Sequencing, which uses clonal array formation and reversible terminator technology[6] (Bentley et al., 2008). Every DNA sequencing starts with library preparation (cf. Figure 1.1a) that includes DNA fragmentation, enzymatic trimming, fragment adenylation and adapter ligation (Mardis, 2013). After library construction, Illumina sequencing starts with cluster generation (the clonal amplification step) on the flow cell surface that includes binding of adapter ligated DNA templates to the oligos on the flowcell followed by repeated bridge amplifications (initiated by polymerases). This procedure results in clusters of co-localized clonal copies of each fragment (cf. Figure 1.1b). After that, each cluster is supplied with a polymerase and four fluorescently labelled nucleotide bases. Reversible terminator sequencing (cf. Figure 1.1c) starts with incorporation of all four bases in each cycle, but only adds one base per cycle at each cluster due to the blocking group attached at the 3'-OH position of the ribose sugar of the nucleotide base, which prevents incorporation of an additional base (Mardis, 2013). The incorporated nucleotide emits a fluorescent signal, which is detected and reported by image sensors. Repetition of these cycles generates a sequence read from each cluster and thus yields millions of DNA sequences at the end. Illumina SBS technology supports both single-read and paired-end libraries. As the names suggest, single-read sequencing allows sequencing in one direction, whereas the paired-end approach performs sequencing from both ends of the DNA fragment. Nowadays, paired-end sequencing is mainly used as it provides better alignment across repetitive regions and also detects rearrangements such as insertions, deletions, and inversions.

[6] http://www.illumina.com/documents/products/techspotlights/techspotlight_sequencing.pdf

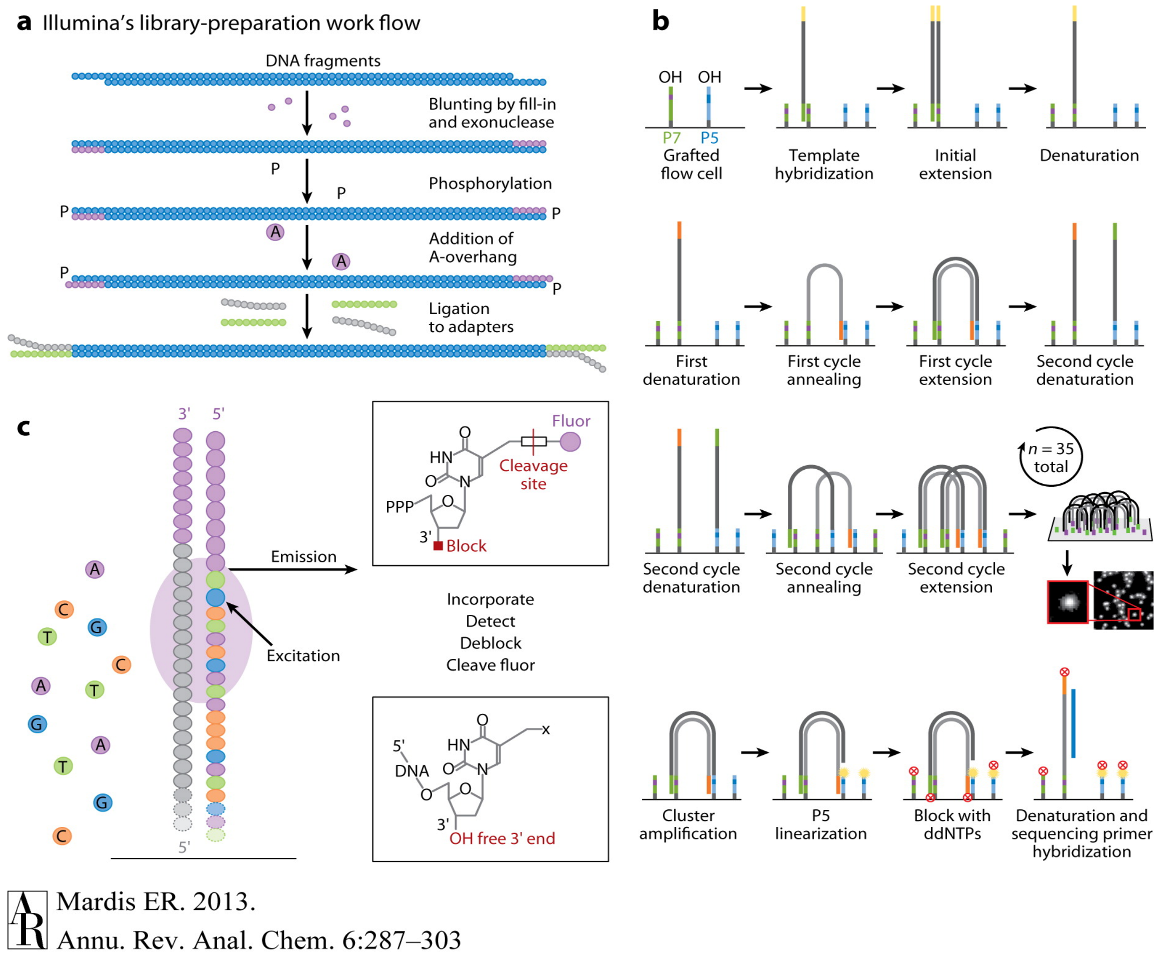


Figure 1.1 Overview of Illumina sequencing (figure from (Mardis, 2013)).

The Illumina platforms can be used for many different sequencing applications, such as whole-genome sequencing, de novo sequencing, candidate region targeted resequencing, DNA sequencing, RNA sequencing, and ChIP-Seq[7]. In order to support all of the mentioned applications adequately, Illumina provides four series of sequencers: MiSeq, NextSeq, HiSeq, and HiSeqX, that vary in throughput, runtime, and cost. The selection of sequencer usually determined on the basis of the project's requirements. At our institute, we use mainly HiSeq for exome sequencing and MiSeq for gene panel sequencing (or other target enrichment sequencing).

[7] http://www.illumina.com/technology/next-generation-sequencing/sequencing-technology.html

1.2 NGS applications

Next-generation sequencing has already many applications and it is still expanding. So far, it has been used in whole genome sequencing (WGS), targeted DNA sequencing (whole exome sequencing, gene panel sequencing, amplicon sequencing), transcriptome profiling (RNA-Seq), DNA-protein interactions (ChIP-Seq), etc. This thesis is focused on targeted DNA sequencing, thus only these applications will be described here.

1.2.1 Target enrichment methods

Though the sequencing cost has decreased significantly over the years and the goal of sequencing a whole genome for 1000 dollar is also achieved (at least partially[8]), still WGS is not routine work for most of the medium sized sequencing centres. Besides of high costs for obtaining approximately 30-fold coverage (i.e. sequencing each genomic position at least 30 times) of a human genome (generates approx. 90 Gb data in total), it needs a lot of computational resources to process and store this huge amount of data (Mamanova et al., 2010). Moreover, the whole genome contains both coding and non-coding part and due to the lack of annotations for the non-coding part, it is hard to interpret WGS data. Therefore, targeted enrichment methods are cost-effective alternates, allowing sequencing of selected genomic regions. There are mainly three different approaches to capture target regions: hybridization based, polymerase chain reaction (PCR) based and Molecular Inversion Probe (MIP). However, due to high cost and some coverage issues of MIP (Mamanova et al., 2010), hybridization or PCR based approaches are the commonly used methods for targeted DNA sequencing. I will describe these methods briefly here, a detailed comparison between them can be found in (Bodi et al., 2013; Kiialainen et al., 2011; Mamanova et al., 2010; Mertes et al., 2011).

PCR based capture

Polymerase chain reaction (PCR) based capturing of a genomic region of interest is a very traditional approach, which has already been used for Sanger sequencing. It has

[8] It needs a specific setup of sequencers (HiSeq X Ten: http://www.illumina.com/systems/hiseq-x-sequencing-system.html), which is very costly and out of the reach of medium sized sequencing centers.

also become ideal for the NGS applications that require to capture smaller targets approximately 10–100 kb in size (e.g. Gene panel sequencing). In brief, this approach starts with a target-specific primer design followed by a PCR reaction. This reaction can be a simple multiplex PCR (e.g. Ion AmpliSeq™ panels from Life Technologies, GeneRead DNAseq Gene Panels from Qiagen etc.), a micro-droplet PCR (RainDance), or array-based PCR (Access Array™ from Fluidigm) (Altmüller, Budde, & Nürnberg, 2014).

Hybridization based capture

These methods are faster and less costly than PCR based methods and used for the larger target size around 500 KB up to 65 MB (whole exome sequencing). They follow the basic principle of hybridization where baits (probes designed to match the target regions for sequencing) hybridize with a DNA library and pull down only fragments from the regions of interest. There are two different ways to execute this procedure: Micro-array based capture and solution based capture. In the first method, hybridization occurs on a micro-array chip where probes fixed to the chip surface hybridize to fragmented genomic DNA and immobilize complementary target sequences. On the contrary, solution based methods (Gnirke et al., 2009) use an excess of biotinylated DNA or RNA complementary probes (mobile probes in solution) to hybridize fragmented DNA. Then, streptavidin labelled magnetic beads are used to purify the target regions (Bodi et al., 2013). The solution based methods require less amount of DNA, and produce more uniform and specific sequences than the array-based method (Mamanova et al., 2010). Moreover, they do not need additional hardware (hybridization station) like the array-based methods. There are lots of commercial kits available based on this approach like SureSelect (Agilent), TruSeq (Illumina), SeqCap (NimbleGen), etc. (Altmüller et al., 2014).

1.2.2 Whole exome sequencing (WES)

The ultimate goal of medical research is the identification of disease causing mutations to find therapeutic treatments or cures. It is known that the majority of the disease-causing mutations are located in coding and functional regions of the genome (Botstein & Risch, 2003; Ng et al., 2009; Majewski, Schwartzentruber, Lalonde, Montpetit, &

Jabado, 2011). Hence, sequencing of the complete coding regions (known as exome) has the potential to uncover causal mutations of genetic disorders (mostly monogenic), as well as predisposing variants in common diseases and cancers (Rabbani, Tekin, & Mahdieh, 2014). Additionally, the exome constitutes only about 1% of the human genome. Therefore, it suffices to sequence approximately 64 MB, making exome sequencing a cost and time effective alternative for WGS (Ng et al., 2010).

Overview of exome sequencing

In general, the sequencing process can be grouped into 3 stages: library preparation, sequencing and imaging, and data analysis (Metzker, 2010). Library preparation is the construction of the DNA templates required for DNA sequencing (Roe, 2004). It starts with DNA isolation and fragmentation (cf. Figure 1.2-A and 1-B), followed by end- repair and adapter ligation (cf. Figure 1.2-C) (Salomon & Ordoukhanian, 2015; Van Dijk, Jaszczyszyn, & Thermes, 2014). Then, the library enrichment for exons (discard the noncoding part) is performed by using target enrichment methods (mostly hybridization based approaches (cf. Section “Hybridization based capture”)). Thereafter, PCR amplification is usually performed to produce a sufficient amount of DNA template. As the final product, library preparation generates the template molecules for amplification on the flow cell surface (cf. Figure 1.2-D). After clonal amplification of the DNA templates into clusters of identical molecules, sequencing starts on the flow cell with a series of cycles. In each cycle, a complementary nucleotide labelled with one of four coloured fluorescent dyes is added to each cluster of identical molecules (cf. Figure 1.2-E). After identification of the fluorescent indicator of each cluster (by using a laser and camera coupled to a microscope), the fluorescent indicator is removed, and the cycle is repeated. Repetition of these cycles generates a sequence read of desired length (usually 75 to 150 bp). During data analysis, sequence reads are aligned to a reference DNA sequence followed by a genotype call for each position by using different bioinformatics tools (cf. Figure 1.2-F) (Biesecker & Green, 2014). The nucleotide detection procedure described above is for Illumina sequencing. Other technologies like Ion Torrent or 454 can perform exome sequencing. Their sequencing procedures differ mainly in clonal amplification and nucleotide detection (cf. Section 1.1).

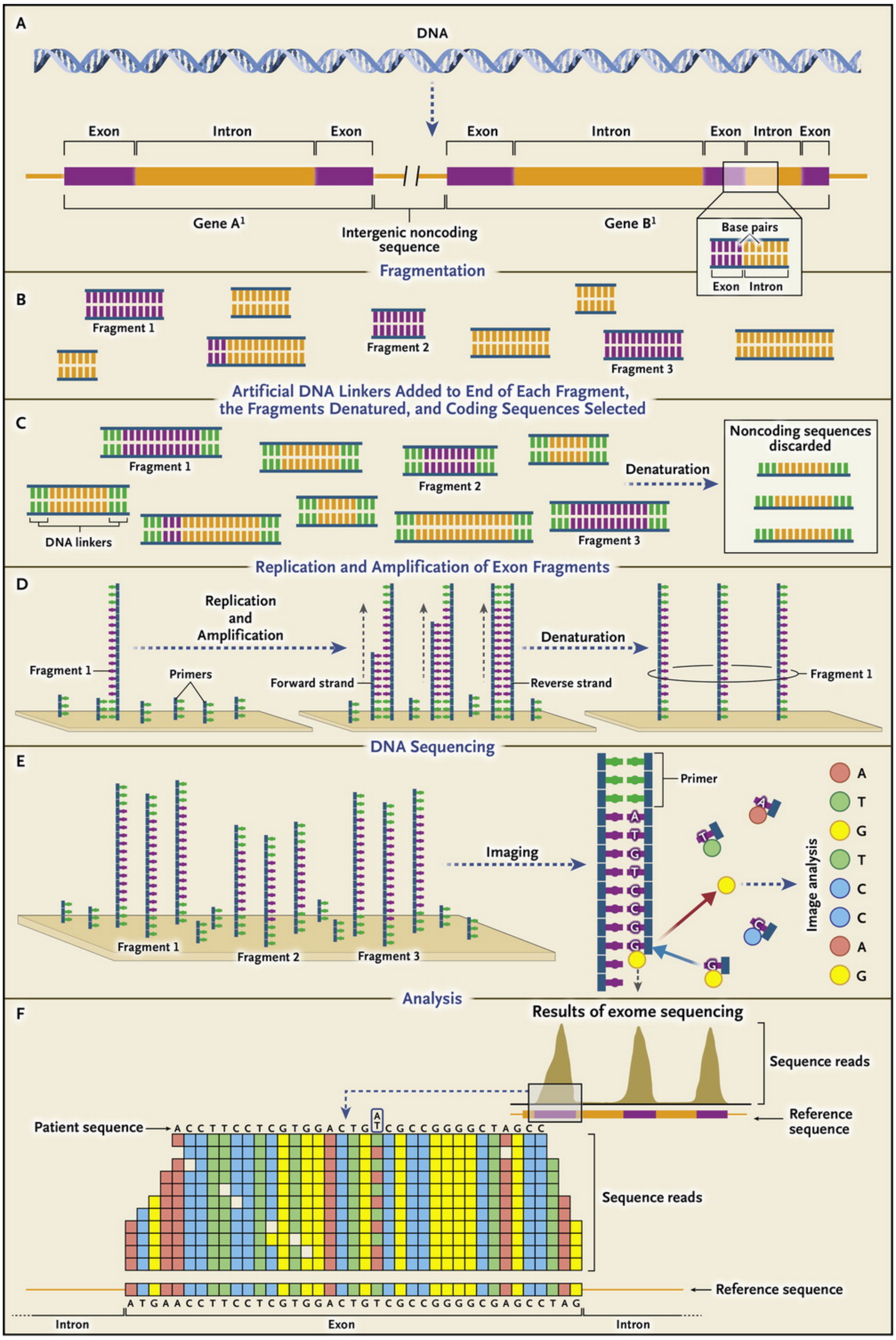


Figure 1.2 Overview of Exome Sequencing (on Illumina sequencer). This figure is taken from (Biesecker & Green, 2014).

Applications of WES

It is already reported in many studies (Botstein & Risch, 2003; Ng et al., 2009) that the exome covers most of the functional variants including nonsense/missense mutations, small insertions/deletions, mutations that affect splicing and regulatory mutations (Rabbani et al., 2012). Hence, WES is a promising method to discover a significant number of disease causing variants and has been used in different areas of research as well as diagnostics (cf. Figure 1.3). In research settings, WES has the following main applications (Majewski, Schwartzentruber, Lalonde, Montpetit, & Jabado, 2011):

1. Characterization of monogenic disorders: WES is widely used to detect mutations causing monogenic inherited disorders. (Rabbani et al., 2012) presented a list of mendelian disease[9] genes identified (between 2010-2012) by exome sequencing and the numbers (102 studies) are significant enough to make WES a success story.
2. Identification of de novo mutations: De novo mutations are the extreme and more deleterious form of rare genetic variations (Veltman & Brunner, 2012). The case–parent trios sequencing is the most frequently used approach to detect this type of mutations (Chesi et al., 2013; Fromer et al., 2014; Vissers et al., 2010).
3. Uncovering the layers of complex disorders: After the success of WES for monogenic disorders, it also became an approach to identify causative variants in heterogeneous, complex diseases (Coonrod et al., 2012; Kiezun et al., 2013). (Jiang, Tan, Tan, & Yu, 2013) reviewed the application of NGS in neurology and presented recent usage (studies from 2011-2013) of WES to identify rare variants in epilepsy, alzheimer, myotrophic lateral sclerosis, parkinson, spino cerebellar ataxias, and multiple sclerosis. Besides numerous applications in neurological disorders, WES is a helpful technique in some common complex disorders like[10]asthma (DeWan et al., 2012), diabetes (Shim et al., 2015; Synofzik et al., 2014), obesity (Paz-Filho et al., 2014; Thaker et al., 2015), etc. Moreover, it is also

[9] Good explanation of mendelian disorders: http://www.nature.com/scitable/topicpage/mendelian-genetics-patterns-of-inheritance-and-single-966

useful in cancer to find germline or somatic mutations (Ku et al., 2013; Ning et al., 2014; Pleasance et al., 2010).

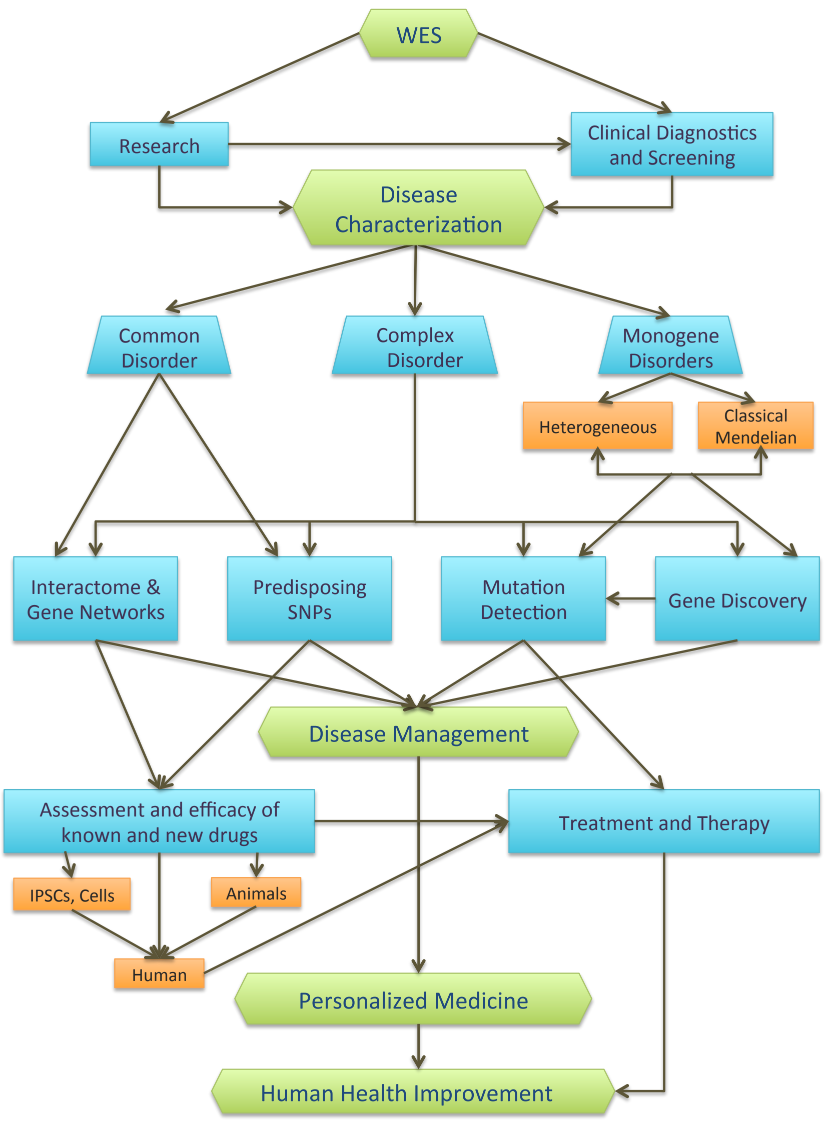


Figure 1.3 Applications of WES and its impact on human health improvement. This figure provides an overview of exome sequencing usage in research and diagnostic settings. The overall aim of both researchers and clinicians is to explore diseases for better disease management, which can lead to personalized medicine for better cure/treatment. This figure is a modified version of the figure in (Rabbani et al., 2014).

Recently, clinicians also started using WES to establish diagnoses for rare, clinically unrecognizable disorders which might have a genetic background (Biesecker & Green, 2014; Coonrod et al., 2012; Need et al., 2012). As mentioned above, WES can find causative variants, which facilitates understanding of the genetic mechanisms of these diseases. This can lead to gene-specific treatments or therapies like gene inclusion or replacement (Aiuti et al., 2009; Cavazzana-Calvo et al., 2010; Ott et al., 2006). WES can be also useful in prenatal diagnosis (PND), pre-implementation genetic diagnosis (PGD), prognosis of preclinical individuals, new-born screening procedures and treatment (Rabbani et al., 2014). It can also help in cancer by identifying driver mutations and genetic events leading to metastasis. This knowledge can be developed into treatments that prevent tumour recurrence or avoid therapeutic resistance (Majewski et al., 2011). Altogether, WES can significantly contribute to personalized medicine and could improve human health by providing better disease management.

Limitations of WES

Besides, the enormous benefits of exome sequencing, there are a few drawbacks (Biesecker, Shianna, & Mullikin, 2011; Directors, 2012; Rabbani et al., 2014). Due to the target design and enrichment procedures, WES

1. is weak in structural variation detection.
2. does not cover certain sets of exons or provides less coverage for some exons, so that causal variants may be missed in some diseases.
3. can miss mutations in repetitive or GC rich regions and in genes with corresponding pseudogenes.

1.2.3 Gene panel sequencing

Gene panel sequencing is another type of targeted sequencing where genomic regions from a set of genes associated with a certain disease are targeted rather than the complete exome[11]. There is prior knowledge about many diseases like their patho-mechanism or responsible genes, hence, sequencing of only those genes is sufficient and more cost and time effective than WES. Gene panel sequencing can use both hybridization based and PCR based target enrichment methods. It provides higher coverage (almost 100% of the target regions are usually covered sufficiently to call

[11] Sequencing overview is almost the same as in WES except for a possibly different target capture.

variants), specificity, and uniformity than WES, which are essential for confident variant detection, especially for low-frequency variants (e.g., mutations only present in a small subset of cells in a tumour sample). Therefore, gene panel sequencing is gaining popularity both in research and diagnostic settings (Altmüller et al., 2014; Glöckle et al., 2013; Lynch et al., 2012; Rehm, 2013; Sie et al., 2014; Sikkema-Raddatz et al., 2013; Wooderchak-Donahue et al., 2012).

Due to its high accuracy, medical centres are making different gene panels for diagnostic purpose. In this context, a brief overview of applications of gene panels and a list of clinically available disease-targeted tests has been reported by (Rehm, 2013). Moreover, (Sikkema-Raddatz et al., 2013) claims that targeted sequencing can replace Sanger sequencing in clinical diagnostics. They compared results from a panel of 48 genes associated with hereditary cardiomyopathies with Sanger sequencing and achieved approximately 100% reproducibility. This study concludes that this panel can be used as a stand-alone diagnostic test. In another study (Consugar et al., 2014), gene panel sequencing was used for genetic diagnostic testing of patients with inherited eye disorders. They designed a genetic eye disease (GEDi) test in this study with high sensitivity and specificity (97.9 and 100% respectively).

In cancer research and diagnosis, gene panel sequencing (or amplicon sequencing) has also been used for many studies to uncover somatic and germline mutations (Dahl et al., 2007; De Leeneer et al., 2011; Harismendy et al., 2011; Meldrum, Doyle, & Tothill, 2011). (Laduca et al., 2014) used four hereditary cancer panels (breast panel, ovarian panel, colon panel, and cancer panel) for diagnosis of hereditary cancer predisposition (Laduca et al., 2014). There are also many commercially available standard cancer panels or other gene panel like Illumina TruSight[12], Illumina TruSeq, Amplicon Cancer Panel[13] or Ion AmpliSeq Cancer Hotspot Panel v2[14] which can be used routinely in both research and clinical settings (Tsongalis et al., 2014). A brief list of standard and customized gene

[12] http://www.illumina.com/products/trusight-panels.html

[13] http://www.illumina.com/products/truseq_amplicon_cancer_panel.html

[14] https://www.lifetechnologies.com/order/catalog/product/4475346

panels can be found in (Altmüller et al., 2014). Moreover, the most recent development in this direction is the "Mendeliome", which is a set of approximately 3000 Mendelian genes known to cause human diseases (Alkuraya, 2014).

1.3 Basic terminologies in data analysis

This section contains definitions of the few terms that are required to understand the content of following sections. Some other terminologies will be defined during their usage throughout the other chapters.

Fastq: It is a text-based file format[15], containing sequence reads and the quality score of every base of the read.

Read: It is a raw sequence (string of the letters A,C,G,T,N) generated from a sequencing machine.

Insert size: In paired-end sequencing, the insert size is the length of the sequenced DNA fragment. It can be computed from the alignment as the length of read1 plus the length of read2 plus the distance between them.

Read alignment: It is alignment (or mapping) of reads from a sequenced DNA on the reference genome sequence, to identify the differences or regions of similarity between both sequences. In general, there are two basic types of alignment:

- **Global alignment:** This is the alignment between entire sequences of approximately equal size. The Needleman–Wunsch algorithm (Needleman & Wunsch, 1970) is the general algorithm for this type of alignment. By using dynamic programming[16], it breaks the sequences into different parts and tries to align them. Then the algorithm searches for an optimal alignment of the complete sequence based on all sub-alignments.

[15] http://en.wikipedia.org/wiki/FASTQ_format

[16] http://en.wikipedia.org/wiki/Dynamic_programming

- **Local alignment:** As the name suggests, local alignment tries to find the most similar part/section between two sequences instead of mapping the entire sequences. The basic algorithm for this purpose is the Smith-Waterman algorithm (T. F. Smith & Waterman, 1981), which is another significant application of dynamic programming.

SAM/BAM: SAM stands for Sequence Alignment Format[17] and BAM is its compressed (encoded) format. SAM is a tab-delimited format (generated after alignment) containing all necessary information related to the alignment of sequencing reads on the reference sequence, e.g. mapping position, mapping quality, sequence reads and their base quality scores etc.

VCF (Variant Call Format): It is a tab-delimited text file format to store sequencing variants with a certain set of information (Danecek et al., 2011) (e.g. position in genome, type of variant, variant supporting evidences etc.).

Coverage: It can be described in two different ways: read coverage and sample coverage. A read coverage (also called read depth) is the number of reads aligning at a specific genomic location. Sample coverage is the average read coverage across all genomic locations targeted during a sequencing experiment. Sample coverage can be represented by values like 10x, 20x, 30x, which means the average read depth is 10, 20, 30, respectively.

Base quality score (Q): It is also known as a phred score (Q) as it is calculated by the Phred algorithm (Ewing & Green, 1998; Ewing, Hillier, Wendl, & Green, 1998). It represents the probability that a base is miscalled. If *P* is the estimated error probability for a base-call, then the phred score is calculated as follow:

$$Q = -10 \log_{10} P$$

[17] https://samtools.github.io/hts-specs/SAMv1.pdf

Mapping quality score (MQ): It is a phred score which represents the probability that a read is misaligned on the reference (H. Li, Ruan, & Durbin, 2008). If *P* is the estimated error probability for a misalignment, then the mapping quality can be calculated as follow:

$$MQ = -10 \log_{10} P$$

Variant quality score (Qual): It is also a phred score like MQ and Q and describes the probability of a called base being an alternate allele. If *P* is the posterior probability P(g|D) of the called genotype g given the observation D, then Qual can be calculated as (H. Li et al., 2008):

$$Qual = -10 \log_{10}(1 - P)$$

For all the quality scores (Q, MQ, Qual), a high phred score means that the examined site has low error probabilities and vice-versa.

Specificity and Sensitivity: In general, specificity estimates the fraction of correctly identified negatives (e.g. wrong outcome), whereas sensitivity estimates the fraction of correctly identified positives (e.g. true outcome)[18]. In this thesis, these terms will be used in two different contexts: variant calling and alignment. Definition of these terms in the context of alignment can be found in Chapter 2 (where these terms have been used for the first time). In the variant calling context, specificity and sensitivity, also known as true negative rate and true positive rate, are measures of the accuracy and the ability to detect a variant of a variant caller, respectively, and can be calculated as follows:

Sensitivity or true positive rate (**TPR**)= $\mathrm{TPs}/(\mathrm{TPs} + \mathrm{FNs})$

Specificity or true negative rate (**TNR**)= $\mathrm{TNs}/(\mathrm{TNs} + \mathrm{FPs})$

False positive rate (**FPR**) = 1- Specificity

*True positives (**TPs**):* These are the real variants, also called by the variant caller.

*False positives (**FPs**):* These are variants that are called by the variant caller but are not the real variants.

*False negatives (**FNs**):* These are real variants that are not called by the variant caller.

[18] https://en.wikipedia.org/wiki/Sensitivity_and_specificity

*True negatives (**TNs**):* These are variants that are not real, and also not called by the variant caller.

1.4 Challenges in NGS data analysis

The rapid fall in sequencing cost expanded the usage of NGS techniques, which resulted in huge amounts of data. Now it's turn for the Bioinformatics community to make sense of these data. There are many challenges, which need to be addressed carefully. They can be categorized mainly in two parts: Efficient data processing and accuracy of results. The challenges belonging to each category are briefly described below.

1.4.1 Efficient data processing

NGS technologies are high-throughput in nature that means they produce huge amounts of sequencing data in a relatively short time. The amount of data can be categorized in two different sections: first data produced by a single sequencing run for a sample and secondly data produced by a study containing certain set of samples. In both of these categories, the amount of data generated is increasing rapidly. Thus, the data processing and management is a major challenge and should be able to address all requirements for different applications.

In order to obtain relevant findings from the raw data, it needs to go through various stages like data cleaning, sequence alignment, variant calling, etc. There are plenty of algorithms for these individual tasks but in order to run them fast, they need to be stitched together in the correct order. This workflow can be very complex and needs enough resources or high performance computing (HPC) for smooth operation. Moreover, it should be automated to reduce the manual efforts and chances of manual errors during data analysis operations.

Nowadays, almost every automated NGS workflow uses HPC clusters. Here, the power of multiple CPUs can be used in a massively parallel way to finish tasks that would take a couple of days on a single CPU in a couple of hours. For example, the MegaSeq

(Puckelwartz et al., 2014) and HugeSeq (Lam et al., 2012) workflows use HPC for whole genome sequencing data analysis. These workflows utilize the MapReduce[19] approach to speed up the data analysis. In the MapReduce approach, first data is split into chunks and then processed in parallel followed by merging of the results. We also use the MapReduce approach to speed up our exome workflow. Existing automated workflows can be Linux-based frameworks implemented on HPC systems via bash[20] scripts like NGSANE (Buske, French, Smith, Clark, & Bauer, 2014) or our exome workflow. Additionally, they can be available in form of a web-interface like WEP (Antonio et al., 2013) or a VirtualBox and Cloud Service like SIMPLEX (Fischer et al., 2012), enabling users with little bioinformatics knowledge to analyse their data on remote HPC systems. All of these above-mentioned examples show the necessity of workflow implementation on HPC systems to speed up NGS data analysis and also provide some details of the HPC implementation and parallelization strategies. However, besides the speed of a workflow, its stability, robustness and maintainability are also important concerns. An automated workflow should be fast, stable and robust as well as easy to maintain. The automation of the workflow and its implementation (with consideration of the above mentioned principles) on HPC comes with many challenges, which will be described in Chapter 3.

1.4.2 Accuracy of results

Downstream analysis of sequencing data can be significantly affected by the accuracy of results. Error propagation during data processing may lead to false positives (FPs) and misleading findings. On the other hand, badly selected strategies/tools can lead to false negatives, which means loss of true variants. These undetected variants might be the causative ones or can lead to uncover additional layers of some diseases, thus, this problem is more severe than FPs. Therefore, to find a proper balance between accuracy and sensitivity of results is another significant challenge for data analysis, which will be described in Chapter 2.

[19] https://www.usenix.org/legacy/event/osdi04/tech/dean.html

[20] http://tiswww.case.edu/php/chet/bash/bashtop.html

Although, there are different alignment algorithms accompanied with post alignment improvements and different variant callers with reasonable accuracy. However, getting a variant list containing only true variants is still a challenge and false positive calls (FPs) remain a problem. Some of the FPs are due to the limitations of current tools, however, random sequencing errors and sequencing biases are other major contributors. Sequencing biases can be categorized in three main classes coverage bias, batch effects and systematic errors (Taub, Corrada Bravo, & Irizarry, 2010; Yang, Chockalingam, & Aluru, 2013).

Coverage bias

Coverage bias means non-uniform coverage throughout the sequenced genomic locations. Genomic locations containing low-complexity regions[21] like GC-rich sequences are the major source of coverage bias, that is non-uniform coverage or even no coverage in these regions (Sims, Sudbery, Ilott, Heger, & Ponting, 2014). (Dohm, Lottaz, Borodina, & Himmelbauer, 2008) first studied the effect of the unbalanced GC content of a genome (known as GC bias) on read coverage and found lower read coverage in GC or AT rich regions. Soon after, Kozarewa and colleagues showed that amplification artefacts introduced during the library preparation are the major source of the heterogeneous distribution of the read coverage (Kozarewa et al., 2009). After that, the GC bias effect has been studied many times in different contexts with similar observations (Aird et al., 2011; Chen, Liu, Yu, Chiang, & Hwang, 2013; Chilamakuri et al., 2014; Clark et al., 2011; Lan et al., 2015). During the library preparation, the PCR step yields lower amplification of regions with high GC or high AT content which results in lower sequencing coverage (Clark et al., 2011). Moreover, the GC bias reduces the efficiency of capture probe hybridization that also leads to lower read coverage (Chilamakuri et al., 2014). This reduced read coverage might result in many false positive (FPs) or false negative (FNs) variant calls (cf. Chapter 2). For example, low coverage at a certain region can be detected as a copy number variant (e.g. deletion) although it is just

[21] DNA regions having biased nucleotide composition and enriched with simple sequence repeats.

because of the GC effect. Similarly, some true variant might be missed or filtered out due to the minimum coverage threshold filter used during variant calling.

Because of the prevalent knowledge about the effect of GC bias, there are many ways to avoid or correct this effect. It can be avoided either during the library preparation (Aird et al., 2011; Kozarewa et al., 2009; Oyola et al., 2012) or compensated during the downstream analysis of sequencing data. (Benjamini & Speed, 2012; Cheung, Down, Latorre, & Ahringer, 2011) provided the "GCcorrect" tool and BEAD algorithm, respectively, that perform different normalization techniques for GC bias correction. Moreover, most of the copy number calling tools based on the depth of coverage approach (Medvedev, Stanciu, & Brudno, 2009) perform read depth normalization to correct for the GC bias (Alkan et al., 2009; Fromer et al., 2012; Krumm et al., 2012; Plagnol et al., 2012).

Batch effect

"Batch effects are sub-groups of measurements that have qualitatively different behaviour across conditions and are unrelated to the biological or scientific variables in a study" (Leek et al., 2010). Thus, the different batches of a sequencing experiment can have errors that are due to the technical or manual variability. For example, usage of different lots of reagents, chips or instruments can introduce variability between different batches of sequencing experiments. Moreover, the individual experience of different technicians handling the samples during the library preparation and sequencing process can also result in batch effects. Batch effects can easily lead to some false discoveries or conclusions. Leek and colleagues has summarized the consequences of batch effects and also demonstrated the presence of batch effects in sequencing data from the 1000 genomes project (Leek et al., 2010). Moreover, they also provided solutions to avoid batch effects. Careful study design and some statistical solutions (exploratory statistical analysis) can easily eliminate the consequences of batch effects (Leek et al., 2010).

Systematic error

Systematic error is the error introduced by the sequencing platform that follows some systematic patterns instead of a random distribution. Systematic errors can be classified as position specific and sequence specific (Meacham et al., 2011). Position specific error is the class of errors where the error (i.e. mismatch) occurs mostly at certain genomic positions or in certain regions of the reads (e.g. base calling errors towards the 3' end). In contrast, the sequence specific error is surrounded (upstream or downstream of the error position) by a certain sequence motif (consecutive nucleotide bases of a certain length).

As Illumina sequencing is the most widely used technology, lots of studies have been performed to understand errors generated from these sequencers. It has been observed that Illumina generates more substitution errors than Indel errors (Dohm et al., 2008). Dohm et al. observed that most of the subsititution errors are either persent at the ends of reads (esp. at the 3' end) or in the GC-rich regions. They found that the nulceotide base A is substituted by C and C is substituted by G more often then the other types of base susbtituion. (Kircher, Stenzel, & Kelso, 2009) studied these base calling errors and found that these errors are more frequent during the first and the last cycles. They observed that the base calling errors are more frequent in the later cycles due to the effects of crosstalk, declining intensities, pre-phasing and phasing, and accumulation of the "T" nucleotide. (Metzker, 2010) reviewed different NGS techonlogies and also reported that the lagging-strand dephasing is the main cause of this type of sustitution errors in Illumina data. Dephasing means loss of synchronicity of growing primers for any given cycle during clonal amplification process (Metzker, 2010). In another word, when the blocking effect of the 3'-OH group (which allows the next nucleotide base incorporation) (cf. Chapter 1) does not work (or still works in the next cycle), some copies of the DNA template lose synchronization with the other copies of DNA template belonging to the same cluster (e.g. incorporation lags one base behind the rest of cluster). It can be referred in two ways: lagging strand dephasing and leading strand dephasing. The lagging strand dephasing refers to the incomple extension of a template ensemble becuase of lagging behind compared to the rest of the cluster as mentioned above. In contrast, leading-strand dephasing is the incorporation of more than one nucleotide in a given cycle (Metzker, 2010). Based on the available knoweldege, lots of improvement has been performed like improvements in the sequencing chemistry and the base calling algorithms. Moreover, recent alignment and variant calling algorithms are also aware of this type of error. Moreover, some standard filtering strategies are also able to filter FPs generated by these errors. However, there is still lots to explore in this class of error as many systematic errors are hard to detect or not detected so far. Thus, Chapter 4 is dedicated to the exploration of systematic errors.

1.5 Thesis organization

As mentioned above, NGS data analysis comes with certain challenges. This thesis is aimed to address some of these challenges to enhance the efficiency and accuracy of DNA sequencing data analysis. The work presented in thesis is my own work except Chapter 3, which is joint work of Dr. Susanne Motameny and myself.

In **Chapter 1**, I have provided an overview of NGS technologies, targeted DNA sequencing methods and their applications. I have briefly mentioned some data analysis terminologies followed by a description of the main challenges of NGS data analysis.

Chapter 2 provides an overview of the data analysis steps of targeted sequencing as implemented in our data analysis workflow. It contains details of essential steps (including their significance) required to generate a variant list from raw sequencing data. It also describes the possible adverse effects of each step (if used inappropriately) and highlights the limitations of the applied algorithms. Moreover, it demonstrates the importance of the selection of appropriate tools and associated parameter settings in terms of both accuracy and efficiency. Furthermore, it provides an overview of filtering and benchmarking strategies applied in our data analysis workflow to achieve a good balance between accuracy and sensitivity of the results.

In **Chapter 3**, details of our in-house developed automated data analysis workflow are provided. The aim was to avoid manual repetition of all of the data analysis steps as mentioned in Chapter 2 and thus to save time for downstream analyses of data and its validation. Fast and efficient automated data processing requires a large amount of computational resources that necessitates its implementation on high performance computing (HPC) clusters. However, HPC comes with some challenges and requires certain workflow design principals as parallelization strategies and some other tricks. This chapter provides details of these design principals and strategies, which gave our workflow speed, stability and robustness while keeping it easy to maintain.

Chapter 4 is dedicated to an important but hard to detect class of sequencing errors: Systematic Sequencing Errors (SSEs), which can easily damage the accuracy of the results. The chapter presents a novel approach to explore SSEs. During the error exploration, reproducible error behaviour in different datasets has been observed which led to a newly coined class of errors: "Recurrent Systematic Errors (RSEs)". The characteristics of RSEs and some of the known and novel patterns behind this type of error are described in this chapter. Moreover, an RSE filtering tool "FilterRSEs" has been developed, which can filter FPs (due to RSE) from a given variant list (VCF file). It uses the detected RSE location and associated annotations to provide different filtering options.

Chapter 5 concludes the thesis with a discussion of the findings and an outlook on current and future developments.

1.5.1 Relevant publications

The following publications related to this thesis have been published:

- *Leveraging the Power of High Performance Computing for Next Generation Sequencing Data Analysis*: Tricks and Twists from a High Throughput Exome Workflow. **Amit Kawalia**, Susanne Motameny, Stephan Wonczak, Holger Thiele, … Ulrich Lang, Viktor Achter, Peter Nürnberg. *PLOS ONE (2015)*
- *Rare variants in γ-aminobutyric acid type A receptor genes in rolandic epilepsy and related syndromes*. Reinthaler EM, Dejanovic B, Lal D, Semtner M,

…**Kawalia A**…EuroEPINOMICS Consortium, Nürnberg P, Lerche H, ….Neubauer BA, Zimprich F. *Annals of Neurology (2015)*

- *Homozygous and compound-heterozygous mutations in TGDS cause Catel-Manzke syndrome*. Ehmke N, Caliebe A, Koenig R, Kant SG, …, **Kawalia A**…Nürnberg P, Siebert R, Manzke H, Mundlos S. *American Journal of Human Genetics (2014)*
- *Mutations in STX1B, encoding a presynaptic protein, cause fever-associated epilepsy syndromes*. Schubert J, Siekierska A, Langlois M, May P, …**Kawalia A**…Nürnberg P, Crawford AD, Esguerra CV, Weber YG, Lerche H. *Nature Genetics (2014)*
- *DEPDC5 mutations in genetic focal epilepsies of childhood*. Lal D, Reinthaler EM, Schubert J, …**Kawalia A**…Nürnberg P, Sander T, Weber Y, Zimprich F, Neubauer BA. *Annals of Neurology (2014)*

Chapter 2
Accurate DNA sequencing data analysis: hurdles and solutions

Almost every day, sequencers produce huge amounts of DNA sequencing data containing millions of sequencing reads. Raw data coming out from these machines contain only sequencing reads and their quality scores. Each sequencing read is just a series of four letters (A, T, G, C) representing the four nucleotides (adenine, thymine, guanine, cytosine, respectively) and does not provide any significant information in its original state. To make sense of these raw sequencing data, they need to go through various bioinformatics analyses. Conversion of a raw sequencing file (fastq file) into a variant list (VCF file) requires certain data processing steps (Van der Auwera et al., 2013) that can be mainly categorized into 4 different sections: pre-processing, alignment, variant calling, and functional annotation.

All of these analysis steps are interconnected, build on each other, and usually involve running some tool or script. Thus, every analysis step can affect its consecutive analysis in an adverse manner if not used appropriately. For example, discarding bad quality data during the quality control step can improve the specificity of results. However, this can lead to loss of data and hence lower sensitivity of results. Similarly, selection of wrong tools/algorithms with their default (or inappropriate) parameter settings can also lead to bad specificity or sensitivity. Both of these situations are harmful for the final results and can mislead variant discovery. Less specific results contain lots of false positives, which make detection of true variants complicated. On the contrary, less sensitive results can lead to the absence of some true variants in the final variant list. Thus, finding a proper balance between specificity and sensitivity of results is very important.

This chapter aims to address these issues in order to achieve a good balance between sensitivity and specificity of the results (cf. Section 1.3). It provides a description of the main steps of DNA sequencing data analysis, their significance and the possible adverse effects. It contains details of pre-processing of raw data, alignment, variant calling, filtering, and evaluations of variant calls. The variant calling step can be categorized in four different parts: Single nucleotide polymorphism (SNP), insertion & deletions (Indels), copy number variants (CNV) and structural variants (SV) calling. I will provide details of only SNP and Indels calling, as I tested and implemented this part of the variant calling. In CNV calling, SV calling, and functional annotation of variants, I have only contributed in the HPC (High Performance Computing) implementation; thus, this part of our workflow will not be described here. In the following, each of these steps are described in separate sections containing the significance of the particular analysis, followed by a brief introduction (or comparison) of relevant tools executing this analysis and their limitations (if exists). Thereafter, details of the selected tool or my own solution (e.g. self-developed scripts) and the reasons behind the selected solution are provided. At last, I provide some insights (both known and learned during testing) into the adverse effects of analysis steps (if used inadequately).

I have tested different tools belonging to all of the above-mentioned data analysis categories in terms of their efficiency and accuracy. A good tool for the data analysis workflow should be fast and resource-efficient (e.g. use a small number of cores and a small amount of memory) (cf. Chapter 3). Moreover, it should generate a balanced result in terms of specificity and sensitivity (cf. Section 1.3). Thus, I have performed extensive testing of different tools and their parameter settings and selected the relatively best tool with its optimum performance. At last, we evaluated the results generated from our data analysis workflow in order to confirm that our selection of tools, parameters and filtering strategies are satisfactory. All of the testing is performed on human sequencing data and all observations or suggestions in this entire chapter are for human sequencing data analysis (especially for exome sequencing experiments with Illumina sequencing technology).

2.1 Pre-processing of raw data

This is the first and a very significant step for any kind of data analysis. Raw sequencing data from sequencers is not always having good quality and can have some fallacious characteristics. Thus, before any analysis steps, a quality check of the data should be performed to distinguish between a bad and a good quality sample[22]. For example, a good quality sample should have base quality scores greater than 20 for all bases at each position in the read, should have less GC bias and no adapter contamination etc. The detailed information about these quality checks and their significance can be found in Appendix. The quality control (QC) can be performed by the FastQC tool[23], PRINSEQ[24] (Schmieder & Edwards, 2011) or any other similar tool. I tested both FastQC and PRINSEQ and found that they are quite similar and efficient enough to perform some basic quality checks (cf. Appendix). However, I selected the FastQC tool for QC, as it is a well tested and adapted tool by the bioinformatics community (Kircher, 2011; Mutarelli et al., 2014; Pabinger et al., 2013). A widely used tool is more likely to be bug free than the less popular tool.

If a sample reported as a bad quality sample by QC reports, then it should be either discarded or should undergo certain processing steps to improve the quality. For example, if a sample is showing bad quality scores towards the end of the reads then trimming of these bases can improve the overall base quality scores. Thus, I perform following QC steps either by default or based on the QC report generated from the quality checks. The decision of performing these steps can vary from one data to another.

[22] Good quality sample: http://www.bioinformatics.babraham.ac.uk/projects/fastqc/good_sequence_short_fastqc.html (This and subsequent URLs are accessed on 28 June 2015)

[23] http://www.bioinformatics.babraham.ac.uk/projects/fastqc/

[24] http://prinseq.sourceforge.net/index.html

2.1.1 Adaptor trimming

Adaptors are short oligonucleotide (oligo) sequences required to link the end of a DNA fragment to the surface of the sequencing platform (cf. Section 1.1.1). If the fragment size is smaller than the read length, the sequencing reaction will continue into the adaptor resulting in the presence of a partial or complete adaptor sequence inside a read (Kircher, Heyn, & Kelso, 2011). These adaptor sequences are not a part of the reference genome sequence and might damage alignment accuracy by inserting lots of mismatches during their mapping on the reference sequence. Therefore, the detection and removal of these sequences can also enhance the accuracy of results.

I tested FASTX-Toolkit[25] and Cutadapt tool (Martin, 2011) for adaptor trimming. Both tools are widely used tools, but I preferred Cutadapt due to its flexibility and additional functionality. Cutadapt performs gapped alignment of single or multiple adaptors on only 3' or 5' end or both end of the read sequence. It allows selecting the mapping error rate and length of overlap and will remove the best matching adaptor. It can either trim few bases or can discard the entire read when the read length becomes too short after trimming. Overall, it is a very flexible tool, which performs good adaptor matching and trimming with its default parameter setting on our exome data. In contrast, the FASTA/Q Clipper method of the FASTX-Toolkit for adapter trimming performs only basic adapter clipping and does not have all of the functionality (or flexibility) provided by Cutadapt.

A faulty adapter trimming can be a reason of loss of information or false positives, if the appropriate parameter selection is not performed. As mentioned above, the default parameter setting works well on our exome data, but an optimum combination of parameters can vary from one data to another. There are two critical parameters in Cutadapt: minimum overlap and error rate. If a low value of minimum overlap and a high error rate are used during adaptor matching, then the tool can perform partial mapping of the adaptor sequence on the reads. These partial matches are considered as adaptor matches that trigger adapter trimming. Thus, the tool will trim a portion of a

[25] http://hannonlab.cshl.edu/fastx_toolkit/

read that actually not having adaptor contamination and this way lead to a loss of sequencing reads.

2.1.2 Quality based trimming

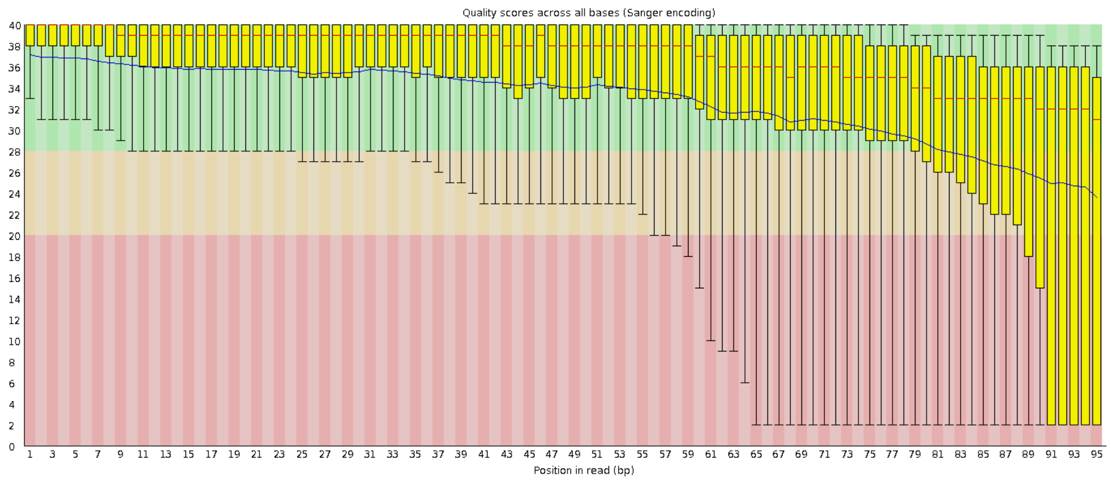
Raw sequencing reads can have low quality bases at one or both ends (Dohm et al., 2008). As explained in Chapter 1, nucleotide bases are identified one by one from the cluster of identical molecules through various sequencing cycles. During these repetitions (cycles), error propagation or accumulation can happened due to various causes, such as, air bubbles, spot-specific signal noise, malfunctioning of laser or lens (Del Fabbro, Scalabrin, Morgante, & Giorgi, 2013). This can lead to quality deterioration towards the ends of a read. Thus, low base quality scores can be found at the ends that indicate less confident base calling meaning that the called base can be wrong. Figure 2.1 shows the effect of quality trimming on one of the in-house sequenced exome sequencing datasets. It contains large amounts of bad quality bases (< 20 base quality score) at the end of reads (cf. Figure 2.1 (A)). However, quality trimming removes all bad quality bases, which improved the overall quality of the data (all bases in reads now have a base quality score > 30) (cf. Figure 2.1 (B)). If the quality trimming is not performed on this kind of data, then mapping of these wrong bases on the reference sequence can easily disrupt the alignment accuracy (resulting in bad mapping quality) and contribute to false positives in variant calling (Del Fabbro et al., 2013). Therefore, trimming or soft clipping (masking) of these bases from both ends (5’ and 3’ end) of a read (or at-least 3’ end) is required. If a read is having high number of bad quality bases, then the entire read should be discarded.

For the quality trimming, I first tested some standard quality trimming tools like FASTX-Toolkit, seqtk[26]. The FASTX-Toolkit quality trimmer works well on the data but performs trimming only on the 3’ end of reads ignoring the 5’ end. Another tool, seqtk is able to perform trimming on both ends. However, both tools do not provide a direct parameter to discard reads that have bad quality bases beyond the given or desired proportion.

[26] https://github.com/lh3/seqtk

Furthermore, they are not able to keep pairing information if one read of a pair is discarded completely. Thus, to overcome these issues, I wrote a perl script to trim bad quality bases from both ends of a read. The trimming approach is partially based on BWA's soft-clipping method[27] as implemented in the "TrimBWAstyle"[28] script. It screens the base quality scores from the end of a read and trims the low quality part (bases below the cutoff value). The script checks for a low quality base (score less than a given quality threshold) among the last or first five bases at the 3' and 5' end, respectively. This base position is the starting position for the trimming action (start base).

A



B

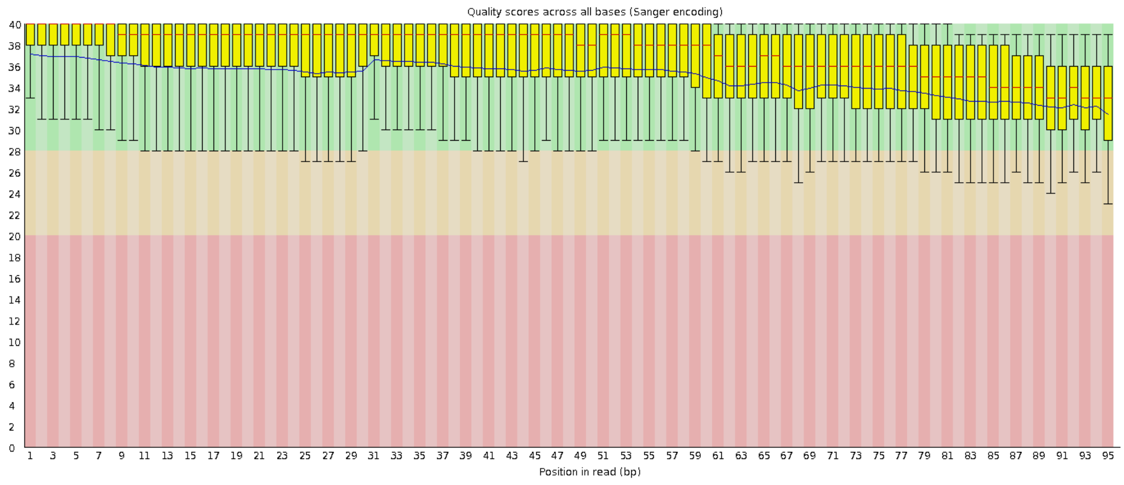


Figure 2.1 Effect of quality trimming on our in-house data. A) Base quality score across all bases before trimming. B) Base quality score across all bases after trimming (Y-axis: base quality score, X-axis: position of base along read).

[27] https://github.com/lh3/bwa/blob/master/bwaseqio.c

[28] http://wiki.bioinformatics.ucdavis.edu/index.php/TrimBWAstyle.pl

The script computes the difference between the quality score of the start base and the given quality threshold. Then it moves to the next base in downstream or upstream direction from the start base (for 5’ and 3’ end, respectively); computes the difference to the given quality threshold (as mentioned above) and adds this new difference value to the prior value. It continues this procedure till the final value of the sum of the differences drops to zero (<= zero). Bases up to the position where the score drops to zero are trimmed. As it considers the quality score of neighbouring bases, it can allow for a few good quality bases during the trimming action (depending on the score’s range) and continue to hunt for preceding/following bad quality bases. Thus, this strategy is better than the basic hard trimming, which lacks the above-mentioned feature and only trims a continuous stretch of bad quality bases. The quality trimming script also discards the entire read if the read length (after trimming) becomes shorter than a given threshold or the percentage of bad quality bases in a read is more than a given threshold. Moreover, it keeps the pairing information when a read is completely discarded during trimming. The pseudo code of trimming script can be found in the Appendix.

In general, quality trimming is beneficial for bad quality data, but it can have some adverse effects. Discarding a significant amount of bases or the entire read could decrease the read coverage, which might affect the sensitivity or specificity of results. There are a few examples to illustrate the above-mentioned effect:

1. If the majority of reads from a sample have intermediate (10-20) or low base quality scores (< 10) and a high quality score (usually 10 or 15) threshold is used for trimming, then most of the data will be discarded that leads to less sensitive results.
2. If the data is having low read coverage and trimming decreases it further, then it can be dangerous for variant calling and could introduce many false positives or false negatives due to lack of supporting evidence.

Overall, the decision to do quality trimming and adapter trimming (including selection of appropriate thresholds or parameters) should be based on the data that needs to be analyse or should be driven from its QC reports. Generalization (or default application) of this approach can hurt downstream analysis steps and produce less reliable results (MacManes, 2013).

2.2 Sequence alignment

Sequence alignment/mapping is the comparison between sequenced DNA and the reference genome sequence to identify the differences or regions of similarity. Thus, the alignment of reads to the reference sequence is the backbone of the entire data analysis. An alignment algorithm tries to find an optimal alignment for an individual read on the reference. For this purpose, it searches the reference genome for locations where the read can be mapped (possibly with gaps or mismatches), assigns an alignment score to each of these mappings and then selects the one with the highest score as the optimal alignment. Whether or not a read can be mapped depends on the number of allowed mismatches and gaps that are usually supplied by parameters and tune the alignment algorithm’s sensitivity and specificity. The sensitivity and specificity of alignment algorithm can directly affect the number of false positive or false negative variant calls, respectively. If the alignment is too specific, i.e. allowing few or no mismatches/gaps during sequence mapping, then it is possible that many reads won’t map to the reference that decreases the read coverage on many sites. Due to the lower coverage, a variant from such a site will not be called, even though it might be a real one (cf. Figure 2.4).

On the other hand, if the alignment is too sensitive, i.e. allowing more mismatches/gaps during the mapping, then it will result in lots of randomly mapped reads and can produce many false positives (cf. Figure 2.4). Besides the allowed error rate, other data characteristics like read length, sequencing platform, type of experiment etc., are also important factors to consider. All of these factors are interconnected to each other, for example, read length and error rate depend on the sequencing platform whose selection depends on the aim of study. Long reads with a low error rate are the best-case scenario having high probability for unambiguous mapping to the reference genome sequence. Therefore, we need to consider the above-mentioned factors carefully, in order to get an adequate balance between sensitivity and specificity. There is a trade-off between the two effects that means if we go for high sensitivity then specificity will decrease and vice-versa.

Recent alignment algorithms (H. Li & Homer, 2010) (based on the above mentioned concepts) are much more complex than the basic algorithms (cf. Section 1.3) and align millions of sequence reads to the reference sequence in more efficient manners. The Burrows-Wheeler Transform (BWT) (Burrows & D. J. Wheeler, 1994) is a widely used algorithm for this purpose and implemented by almost all popular alignment tools, e.g. BWA (H. Li & Durbin, 2009), Bowtie (Langmead, Trapnell, Pop, & Salzberg, 2009), SOAP2 (R. Li et al., 2009). BWT is a fast and memory efficient algorithm that performs index-based sub-string searching by using a reversible permutation of characters.

I used BWA for the alignment of reads to the human reference genome (GRCh37[29]). This reference sequence is taken from 1000 genome project[30], but I replaced the original mitochondrial DNA sequence with NCBI's mitochondrion reference sequence (NC_012920[31]). BWA is one of the accurate and widely used aligner for DNA sequencing reads. It has been used in many studies and has also been compared with other popular alignment tools like Bowtie and Novoalign[32] (Giannoulatou, Park, Humphreys, & Ho, 2014; Hatem, Bozdağ, & Çatalyürek, 2011). (Highnam et al., 2015) developed a benchmarking platform (GCAT[33]), where different data analysis algorithms can be compared on real biological data. GCAT benchmarking also suggests that the BWA-MEM algorithm is one of the best alignment algorithms similar in terms of correctly aligned reads to Novoalign (the best alignment algorithm so far, but only commercially available

[29] http://www.ncbi.nlm.nih.gov/projects/genome/assembly/grc/human/

[30] http://www.1000genomes.org/category/reference

[31] http://www.ncbi.nlm.nih.gov/nuccore/NC_012920.1

[32] http://www.novocraft.com/products/novoalign/

[33] http://www.bioplanet.com/gcat

with full functionality) (cf. Figure 2.2). BWA can perform both paired-end (PE) and single-end (SE) alignment by 3 different algorithms[34]:

1. BWA-backtrack (H. Li & Durbin, 2009): performs gapped global alignment and is mainly used for Illumina PE and SE reads (< 100 bp).
2. BWA-SW (H. Li & Durbin, 2010): performs Smith-Waterman alignment and is typically used for 454/Sanger single-end reads (up to 1 MB). It can also perform sensitive alignment of paired-end reads (> 100 bp).
3. BWA-MEM (H. Li, 2013): This is the latest algorithm (similar to BWA-SW) that searches for maximal exact matches (MEMs). It is more accurate and faster than the other two algorithms and can replace them for alignment of reads between 70bp to 1Mbp.

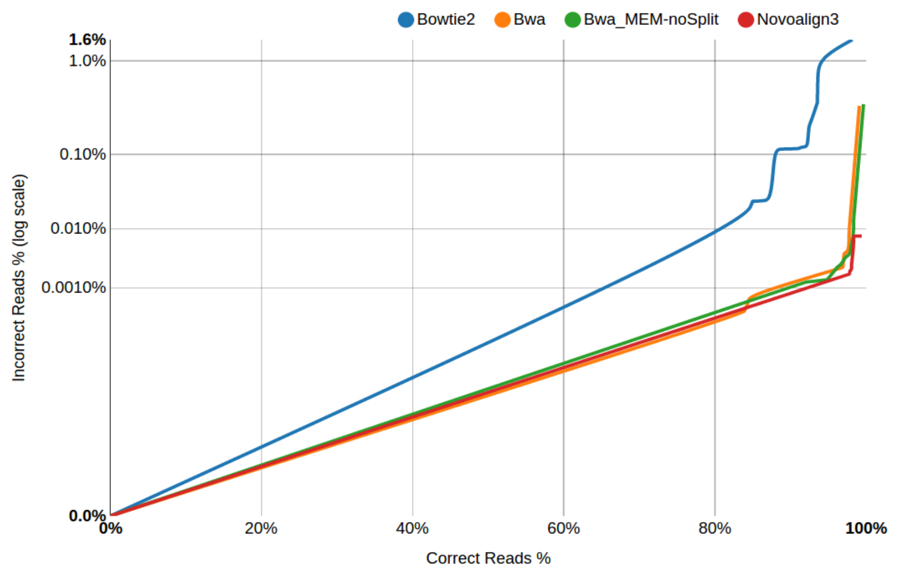


Figure 2.2 GCAT comparison of 4 different alignment algorithms. It shows the percentage of correctly and incorrectly mapped reads (on X and Y axis respectively) on simulated paired-end 100-bp Illumina reads. This figure is taken from (Highnam et al., 2015).

I performed extensive testing of BWA parameters to get the best possible alignment. Initially, BWA offered only two different algorithms: backtrack and SW (see above). As most of our data (esp. from Illumina exome sequencing) contain reads less than 100bp, I

[34] Before using these algorithms, one first needs to construct the FM-index for the reference genome by using BWA's index function.

selected BWA-backtrack for alignment. I also used BWA-SW for alignment of sequencing data generated from Ion torrent and 454 sequencing technologies (as they generate longer reads > 100bp). However, as mentioned above, I mainly used Illumina data and performed major testing and analysis with these data only. I tested some of the relevant parameters of the BWA-backtrack to see their effects on alignment like maximum edit distance (n), maximum edit distance in the seed (k), seed length (l), etc. (cf. Appendix). I found that the default parameter settings work well for most of these parameters. However, I observed significant differences in the sensitivity and specificity of the alignment when changing the maximum edit distance (n) parameter[35]. This parameter regulates the number of allowed mismatches during alignment of reads. Thus, higher value means more sensitive alignment and vice-versa. Recently, BWA started to perform better alignment by using the new MEM algorithm compared to the alignments generated by backtrack algorithm. BWA-MEM allows moderate error rates during alignment and provides a good trade-off between sensitivity and specificity of alignment. It is also recommended by the BWA developer to replace the old backtrack algorithm by MEM on the reads having read length greater than 70bp. I compared the generated alignments on our control sample NA12878 (cf. Section 2.4.2) from both algorithms in the following contexts: alignment statistics and evaluation of variants called from the generated alignments.

Alignment statistics

There are a few basic alignment statistics like total number of aligned reads, high quality aligned reads or bases etc. that can provide an overview of an alignment algorithm's sensitivity or specificity. In general, a higher percentage of aligned reads means that the alignment is more sensitive. Table 2.1 shows alignment statistics computed from alignments generated by BWA-MEM (with default parameters) and BWA's backtrack algorithm (BWA-aln) with default parameters as well as with n=7 (remaining parameters are at default values). I used the Picard tool to compute these alignment statistics (cf. Section 2.2.2). As shown in the table, BWA-MEM performs better in every aspect of the

[35] http://bio-bwa.sourceforge.net/bwa.shtml

read alignment. It aligned more reads with a higher percentage of high quality (HQ) aligned reads and bases than the BWA-aln algorithm (at both default and n=7). This means BWA-MEM performs read alignment with higher sensitivity and specificity (with less noise reads containing only of A bases and/or N bases[36] entirely) than the BWA-aln algorithm. Moreover, BWA-MEM is faster than BWA-aln and takes approximately 4 hours less compared to BWA-aln at n=7. As mentioned above, BWA-MEM works better with reads having a read length greater than 70bp. Thus, BWA-aln algorithm is required for the alignment of short reads (< 70bp read length). This algorithm is more sensitive at n=7, as it aligns more reads with a bit higher percentage of high quality (HQ) aligned reads and bases, than at default parameters.

BWA's algorithms	Total Reads	% Aligned Reads	% Reads aligned in pair	% HQ Aligned Reads	Noise Reads	% HQ Aligned Bases	Approx. Time taken[37]
aln (default n)	92078710	0.968	0.995	0.937	487	0.938	3,0
aln (n=7)	92078710	0.974	0.995	0.938	469	0.938	5,30
MEM	92078710	0.988	0.995	0.940	0	0.942	1,10

Table 2.1 Comparison between BWA's alignment algorithms in context to reads alignment.

BWA's algo-rithms	Raw variants	Variants After Hard Filtering	Recal SNPs	Ti/Tv ratio Recal SNPs	Ti/Tv ratio After Hard Filtering	TPR Recal[38] SNPs	FPR Recal SNPs	TPR After Hard Filtering	FPR After Hard Filtering
aln (n=7)	588102	517099	38156	2.74	2.85	0.995	0.029	0.986	0.007
MEM	730044	670800	37592	2.81	2.87	0.995	0.017	0.986	0.004

Table 2.2 Comparison between BWA's alignment algorithms in context to called variants.

[36] https://broadinstitute.github.io/picard/picard-metric-definitions.html#AlignmentSummaryMetrics

[37] The format of time is in hours, minutes. All algorithms ran with 4 cores and 8 gb RAM.

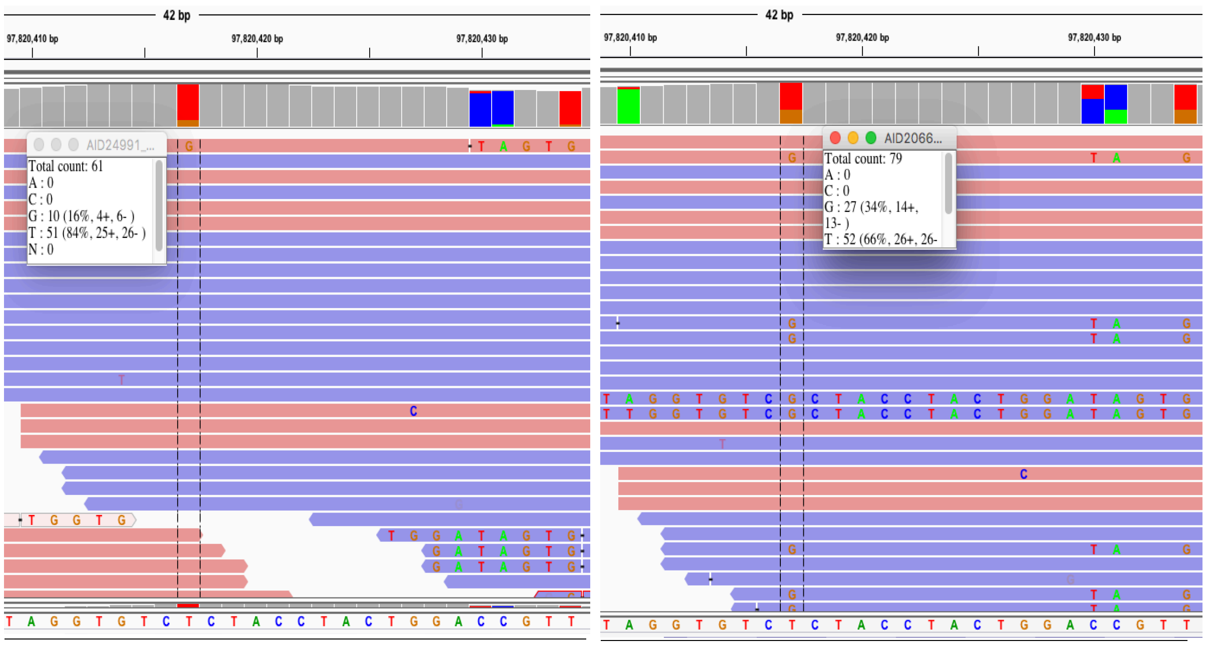
[38] Filtered SNPs by VQSR

Variant list evaluation

The performance of alignment algorithms can also be evaluated by the sensitivity and specificity of variant lists (cf. Section 2.4). To do this, I generated variant lists with GATK's Unified Genotyper variant caller (cf. Section 2.3), using reads (100 bp or longer) aligned by both BWA-MEM and BWA-aln algorithms followed by the same post alignment improvements (cf. Section 2.2.1). Table 2.2 shows the number of called variants with their evaluation. The variant list generated from the aligned reads by BWA-MEM shows the higher Ti/Tv ratio, lower FPR with the same TPR after both types of filtering (Hard and VQSR) (cf. Section 2.3.1). This clearly indicates that BWA-MEM is a better algorithm than the BWA-aln for reads greater than 70 bp in length.

Effect of alignment sensitivity and specificity at variant sites

Besides the evaluation of sensitivity and specificity of called variants, I observed some significant effect on a few variants at different levels of alignment sensitivity. Figure 2.3 shows the difference between the alignment of reads at position chr 2: 97820417 (highlighted with vertical dotted bars) by BWA-aln (shown in the right part of the figure) and BWA-MEM (shown in the left part of the figure). BWA-aln (at n=7) is having high sensitivity thus it aligned a few more reads (18 reads more than BWA-MEM), which provided strong support for the alternate allele "G" at this position with 34% frequency compared to 16% frequency in aligned reads by BWA-MEM (frequencies of alternate allele and reference allele are shown in white boxes). Due to the high alternate allele frequency this variation can easily be called as a SNP, although it is a FP (validated by GIAB benchmarking (cf. Section 2.4.2)). Moreover, the sensitive alignment by BWA-aln also increased the frequency of nearby alternate alleles depicted in the coloured bar in the coverage track (shown in grey colour at top of the both figures), which results in more FPs from these sites (also reported as FPs by GIAB benchmarking). However, due to the moderate sensitivity of BWA-MEM, fewer mismatches are allowed during the mapping of reads that resulted in a less noisy alignment, and thus, less FPs at this site.



AID24991_...
Total count: 61
A : 0
C : 0
G : 10 (16%, 4+, 6-)
T : 51 (84%, 25+, 26-)
N : 0

AID2066...
Total count: 79
A : 0
C : 0
G : 27 (34%, 14+, 13-)
T : 52 (66%, 26+, 26-

Figure 2.3 Difference between the alignments produced by BWA-MEM and BWA-aln at n=7.

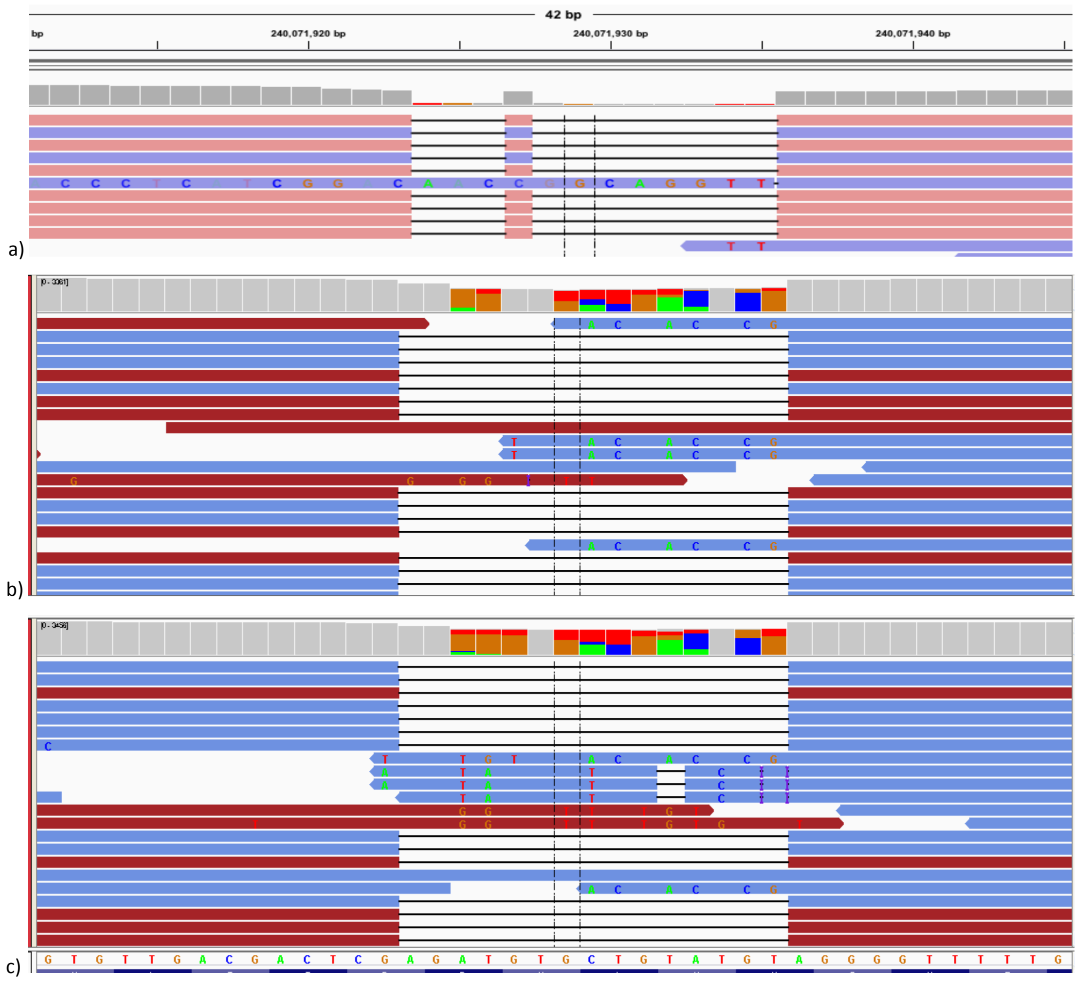


Figure 2.4 Difference between the alignments produced by BWA-aln at n=0.04, n=7 and n=10.

Similarly, Figure 2.4 shows the adverse effect of sensitive alignment on the real deletion of 11bp in one of the old test exome datasets (sequenced in-house). Part (a) of the figure shows an alignment performed by BWA-aln at default value of n (n=0.04), part (b) at n=7 and part (c) shows a more sensitive alignment at n=10. At default n, the BWA-aln performs more specific alignment, thus we can observe a lower number of aligned reads in the coverage track (shorter grey coloured bars) compared to the other diagrams. Due to the lower number of the supporting reads (only 10 reads), this variant can be skipped or filtered by variant callers. In the other alignments n=7 (part b) and n=10 (part c), more mismatches are allowed during the alignment that result in a few more supporting reads for this deletion. However, due to the increased sensitivity at n=10, a few extra reads with a one base deletion and lots of mismatches can be observed.

After observing the effect of sensitivity and specificity of the alignment algorithm, I decided to rather perform sensitive alignment instead of a too specific one, as false positives can be filtered at any stage after the alignment. However, lost data due to specific alignment cannot be retrieved later on. As most of the recent Illumina sequencing reads are longer than 70bp, I have not tested the effects of BWA-MEM on sequencing reads < 70 bp and simply follow the BWA developer's recommendation to use BWA-MEM for reads longer than 70 bp. I use BWA-backtrack in sensitive mode (n=7) only for Illumina sequencing reads < 70 bp, whereas BWA-MEM is used for reads having read length => 70 bp. I also use the same combination for the alignment of sequencing reads generated from Ion-torrent sequencers. These reads are usually having unequal read length, thus, I split the data into two parts: reads having length < 70 bp and reads having length > 70 bp. After data splitting, I perform alignment similar to Illumina reads (as mentioned above). Hence, the selection of algorithm and parameter tuning should also be based on characteristics of the targeted data like read length, sequencing platform etc.

Even with the proper selection of the parameters, alignment algorithms are not able to perform accurate mapping in certain genomic regions, especially in the following genomic regions:

- Low complexity regions (LCRs): LCRs are the regions that are highly enriched with one or combination of a few nucleotide bases (Radó-Trilla & Albà, 2012), for example, simple repeats (micro-satellites), poly-purine/poly-pyrimidine stretches, or regions of extremely high AT or GC content[39]. In these regions, alignment algorithms produce mappings with many mismatches or fail to map a read on the reference sequence, using sensitive or strict alignment, respectively. Moreover, due to the sequence similarity in repetitive regions, they can map a read at multiple positions or at the wrong position (Treangen & Salzberg, 2012). For example, reads belonging to gene A can map on gene B having high sequence similarity to gene A or can map on both genes. This might lead to either loss of a true variant or many FPs (SNPs, Indels). Indel errors are very prominent in these LCR regions and also cannot be ignored by modern variant callers which perform realignment before variant calling (H. Li, 2014) (cf. Section 2.3).
- Highly divergent regions (HDRs): HDRs are regions where sequences are having less than 99.5% identity in the human genome (e.g. GC rich regions, highly recombining subtelomeric regions) (Kuruppumullage Don, Ananda, Chiaromonte, & Makova, 2013). These regions are hard to sequence by short read NGS techniques and also cannot be mapped accurately by alignment algorithms. These regions sometimes resemble to some structural variants[40] (SVs) associated with segmental duplications and are also problematic for SV detection.
- Regions belonging to paralogous genes: Paralogous genes are duplicated genes within a species (usually genes belonging to a gene family). They can have slightly different functional roles or can be pseudogenes (functionally inactive genes). In this case the aligner can map reads to multiple positions (to a gene as well as to its paralogous gene(s)) or can produce a wrong mapping (mapping on the paralogous gene(s) only). Therefore, these genes contain many FPs and appear

[39] http://www.repeatmasker.org/webrepeatmaskerhelp.html#lowcomp

[40] http://grantome.com/grant/NIH/F32-GM097807-01

frequently during variant discovery experiments (Arthur, Cheung, & Reichardt, 2015; Meldrum et al., 2011).

2.2.1 Post processing of aligned reads

Quality control and good alignment (with optimum settings) are not sufficient to get an accurate alignment of the reads. There are still some errors or mapping artefacts present in the data that can be addressed or filtered only after alignment (Liu et al., 2012; Nielsen, Paul, Albrechtsen, & Song, 2011). Thus, to avoid these errors additional post processing of aligned reads is required. This post processing is suggested by the GATK best practice guideline[41] (Van der Auwera et al., 2013) and has become a standard procedure or default step in the majority of the data analysis workflows (D’Antonio et al., 2013; Lam et al., 2012). I also tested it and found it useful for exome sequencing data. The following sections contain three steps (cf. Figure 3.1) of the post processing of the aligned reads and their improvements.

Duplicate marking (removal)

PCR amplification during library construction can result in duplicate reads, which might lead to FPs during variant calling. A variant at some site can be called by a variant calling algorithm due to the high read depth (number of supporting reads) that is actually only generated by an accumulation of duplicates. Additionally, many duplicates of a read with wrong Indels (aligned due to the in-accurate read mapping) can mask the correct Indels during Indel realignment (cf. Section “Local Indel realignment”).

I used the Picard tool[42] to remove duplicates. Picard is widely used tool and has many functionalities like duplicate removal/marking, BAM to fastq conversion, BAM/SAM file merging, computation of alignment and enrichment statistics etc. Its MarkDuplicates algorithm compares 5’ coordinates and mapping orientations of each read pair and marks all pairs as duplicates for which these parameters are identical. It keeps one read

[41] https://www.broadinstitute.org/gatk/guide/best-practices?bpm=DNAseq

[42] http://broadinstitute.github.io/picard/command-line-overview.html#MarkDuplicates

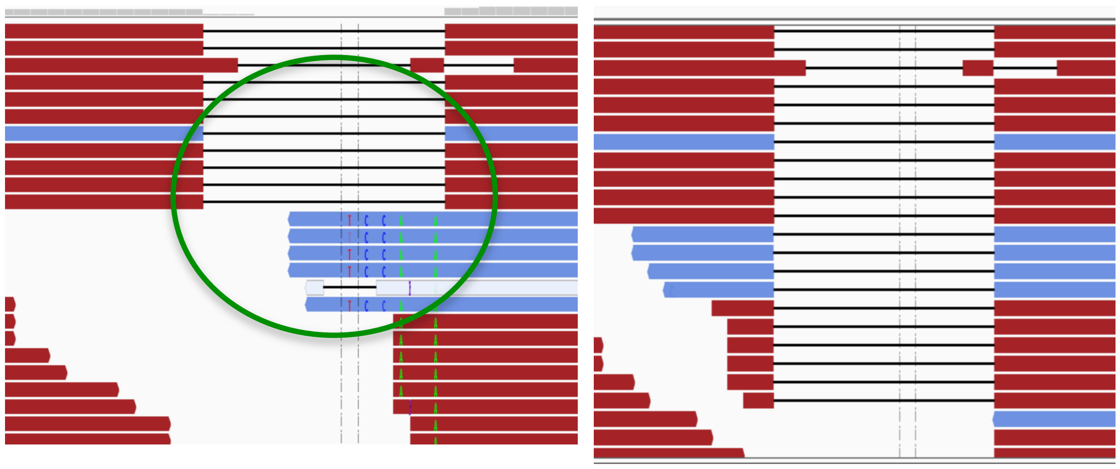
pair[43] among all duplicates based on the highest sum of base qualities, where the pair should have all bases with quality (Q) >= 15. Duplicate removal should be performed on the entire BAM file (not on BAM files split e.g. by chromosome), to find inter-chromosomal duplicates. In some experiments, due to their chemistry (e.g. in the PCR based target selection approach), most of the reads can be detected as PCR duplicates by the MarkDuplicates tool. However, these reads are not duplicates and discarding them might lead to significant loss of data. Therefore, an on/off switch for triggering duplicate removal based on the type of data is used in our data analysis workflow.

Local Indel realignment

Alignment algorithms can fail to map some Indels correctly at some position and can produce different alignments for different reads. Some reads can have one or a few mismatches at that position instead of one complete Indel, and some reads can map without Indels. These misalignments can result in FPs or loss of some Indels during variant calling. Thus, Indel realignment is an essential step to correct misalignments by performing multiple sequence alignment (MSA) on suspicious positions. Figure 2.5 shows some alignment inaccuracies and their correction by local Indel realignment at a certain position of our in-house data. Part (a) of the figure shows the raw alignment (before realignment). It is showing a deletion (and some mismatches) with very few supporting reads (highlighted in the green coloured circle shape) that might lead a variant caller to skip this deletion. In part (b), we can see the improved alignment after Indel realignment and a sufficient increment in the number of supporting reads to detect the deletion (without mismatches) present at this site. The Indel realignment method is a part of the Genome Analysis Tool Kit[44] (GATK) (DePristo et al., 2011; McKenna et al., 2010). It is a two-step process: at first, it checks for suspicious intervals (esp. near Indels) in the alignment and then performs local realignment (MSA) over those intervals. The detailed information of this two-step process can be found in Appendix.

[43] http://broadinstitute.github.io/picard/faq.html

[44] https://www.broadinstitute.org/gatk/guide/topic?name=methods



a.) Before Indel realignment b.) After Indel realignment

Figure 2.5 Improvement in the alignments after Indel realignment process.

Base quality score recalibration (BQSR)

BQSR is also a part of GATK, which adjusts under or over estimated base quality scores, resulting from systematic biases during sequencing or due to the inaccurate estimation by the sequencer's basecalling software. It assigns new quality scores by calculating the probability of a mismatch on the reference sequence. For example, if the original quality score of a base is 25 (good enough for variant calling), but if this base is actually observed to be a mismatch on the reference at a 1 in 100 rate, then it should have 20 as a quality score value. Thus, an overestimated quality score provides false confidence in the base call and can lead to FPs. Moreover, it is known that the base calling errors are higher towards the end of the reads (means in lower cycles) than at the beginning of the reads. It has also been observed that the mismatches are associated with sequence context, the dinucleotide AC is usually having lower quality than TG[45] (Dohm et al., 2008; Nakamura et al., 2011). BQSR is addressing these errors by correcting bases quality score based on the analysis of four different covariates: read group, quality score, machine cycle and di-nucleotide, from the given data. The details of BQSR method can be found in Appendix.

[45] http://gatkforums.broadinstitute.org/discussion/44/base-quality-score-recalibration-bqsr

2.2.2 Alignment & enrichment statistics

After post alignment improvement, it is good to assess the overall sample quality in terms of coverage and alignment quality. Both, specificity and sensitivity of variant detection can be lowered by bad quality data. Thus, alignment and enrichment statistics can help to decide whether a certain sample is good enough for variant calling or not. For example, a sample should have at least 20X read coverage with good mapping quality (> 20 or > 30) throughout the sample. Moreover, a high percentage of aligned reads, uniform quality distribution by cycle, small deviation in the insert sizes are other useful parameters to judge a sample and its alignment quality. I used the Picard tool to compute alignment and enrichment statistics. It generates a summary of alignment[46] parameters, e.g. total number of aligned and unaligned reads, read length and insert size distribution, mapping quality distribution, quality by machine cycle, etc. It also summarizes the sample coverage[47] by giving coverage values at 2X, 10X, 20X, 30X and the proportion of on/off target reads.

2.3 SNP/Indel calling

Apart from the alignment accuracy, accurate SNP/Indel calling is also necessary to find some true causal variants in the sequencing data. There are lots of different variant calling algorithms that claim to find variants from sequencing data (Bao et al., 2014; Pabinger et al., 2013). However, it has been reported that there is low concordance between the results from different variant callers (O'Rawe et al., 2013; Yi et al., 2014). Therefore, I also tested 4 different variant callers: Platypus (PP) (Rimmer et al., 2014), Samtools (H. Li et al., 2009) mpileup (MP), GATK's Unified Genotyper (UG) and Haplotype Caller (HC) (DePristo et al., 2011; McKenna et al., 2010).

Variant lists were generated from BWA-MEM alignments of the in-house sequenced control sample NA12878 (cf. Section 2.4.2) followed by post-processing. Default or suggested parameters have been used for all variant callers (with base quality score>10).

[46] https://broadinstitute.github.io/picard/picard-metric-definitions.html#AlignmentSummaryMetrics

[47] https://broadinstitute.github.io/picard/picard-metric-definitions.html#HsMetrics

Figure 2.6 shows the number of variants in sample NA12878 called by Samtools mpileup, GATK's HC and Platypus. As can be seen in the figure, HC is more sensitive and calls a larger number of variants than the other tools. On the other hand, mpileup focuses on accuracy and calls much less variants than the other callers. Among our four variant callers, GATK's UG and Samtools' mpileup use a Bayesian approach (H. Li, 2011) to call SNPs and Indels and treat each position independently. These alignment-based approaches provide sensitive variant calling but also produce many FPs. The algorithms rely on alignment accuracy, which is not good in low complexity regions (LCRs) and in highly divergent regions (cf. Section 2.2). Although misalignment near Indels can be corrected by local realignment (cf. Section 2.2.1), still these algorithms can produce errors around Indels and larger (complex) variants (Rimmer et al., 2014). On the contrary, PP and HC are the most recent and sophisticated haplotype-based callers. Both tools perform local denovo assembly (by building a De Bruijn-like graph) in order to find the correct haplotype, which is used for SNP/Indels identification, and try to compensate for the above-mentioned drawbacks. However, these methods can perform badly in repetitive regions due to the loss of contiguity information during the segmentation of reads into consecutive k-mers, which is required for graph construction (Rimmer et al., 2014; Zerbino et al., 2008). Thus, the integration of callers from these two different algorithms is better than the usage of a single algorithm to achieve highly accurate and sensitive variant calls.

Table 2.3 shows the performance of different variant callers and the integrated approach (the combined results from different callers) in the context of sensitivity and specificity of variant lists generated for control sample NA12878 (cf. Section 2.4). Mpileup's VCF file is excluded from this comparison due to some compatibility issues. In this comparison, a significant difference between numbers of raw variant calls can be observed which result in little overlap between these callers (cf. Figure 2.6). Unified Genotyper (UG) calls approximately two times more variants than Haplotype Caller (HC) and Platypus (PP). These three callers showed good sensitivity (TPR > 99%) with reasonable specificity (FDR < 0.02). However, the combined results from these three callers, where a variant should be called by at least 2 callers (minN2), showed better results in terms of specificity (lower FDR and better PPV) with almost identical sensitivity. The Ti/TV ratio of the combined variant calls is also higher than the calls from an individual variant caller.

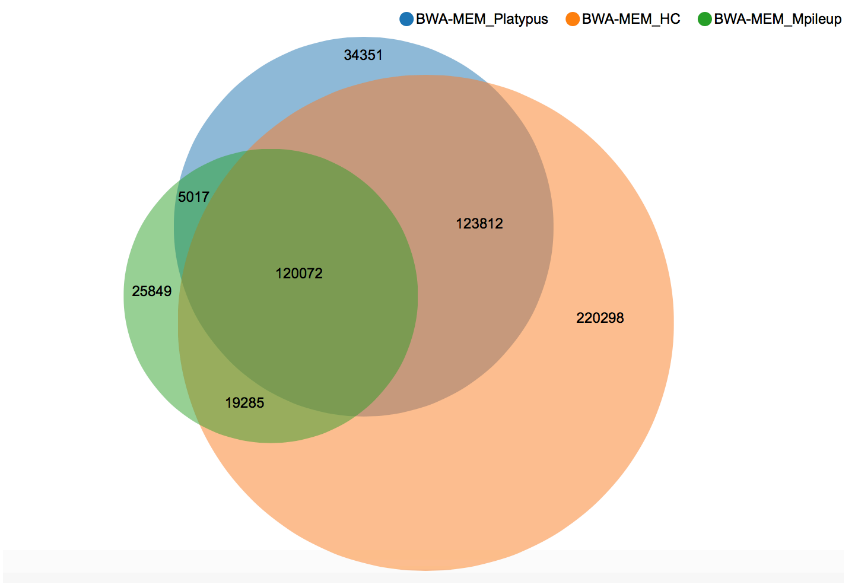


Figure 2.6 Overlap between raw variant lists of control sample NA12878 generated from 3 different variant callers. Number of variant calls generated by Samtools mpileup, GATK haplotype caller (HC) and Platypus are shown in green, orange, and blue colour, respectively.

Variant Callers	Raw Variants	Shared Raw Variants	Match	Non-Match	FDR	TPR	FPR	PPV	Ti/Tv
UG	718794	23721	23327	394	0.0166	0.995	0.0168	0.983	2.812
HC	443366	23524	23286	238	0.0101	0.994	0.0102	0.989	2.851
PP	283755	23602	23275	327	0.0138	0.993	0.0140	0.986	2.818
Combined (minN2)	303722	23412	23275	137	0.0058	0.993	0.0058	0.994	2.872
Combined (minN1)	735351	23885	23341	544	0.0227	0.996	0.0232	0.977	2.78

Table 2.3 Comparison of variants callers' performance to the integrated approach.

Similar observation can be made in Figure 2.7, which shows that the results from integration of variant callers are more accurate. The figure shows the Qual score threshold used in the comparison of variant list with NIST list. The variants generated from the combined method are having high confidence variants with Qual score greater than 30, compared to the individual callers where many variants are having Qual score between 7-30 range. Therefore, I decided to use a set of variant callers (Platypus, Samtools, GATK's UG and HC) to perform SNP/Indel calling and combine their results. A recent comparison also reports the same finding that the integration of multiple callers is better than any single variant caller to get variants of high confidence and sensitivity (Bao et al., 2014).

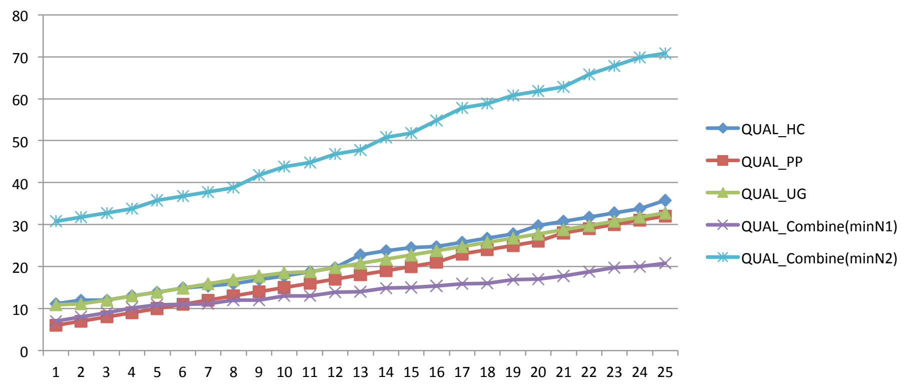


Figure 2.7 Variant Qual score thresholds used during the comparison of individual callers and the integration approach. X-axis shows number of selected thresholds and Y-axis shows the Qual score.

2.3.1 Filtering strategies

Continuous development of algorithms has already improved variant calling up to a significant level and further improvements are still going on. Although the current algorithms are rather accurate they still produce a lot of FPs, due to their moderate filter settings (to balance between specificity and sensitivity) or due to sequencing artefacts (e.g. systematic bias, alignment bias or some random errors etc.). In order to filter these FPs, additional efforts are required. Filtering strategies can vary from filtering based on some fixed values (hard filtering) to filtering based on models trained from variants of each data set individually (VQSR, Variant Quality Score Recalibration). Moreover, it can be based on prior knowledge about the disease or the design of the study (e.g. based on pedigree, trios etc.) or based on functional effects of the called variants on the phenotype. In general, hard filtering and/or VQSR can be applied on most of the exome data with some care (cf. next paragraph).

VQSR is a Gaussian mixture model[48], which categorizes a variant list into reliable and unreliable variant calls. It takes the overlap between a variant loci list from the HapMap3/Omni 2.5M SNP chip array (the truth/training resource sets) and the given variant list. Then, it learns the distribution of some SNP annotation values present in this list (e.g. QD, SB, HaplotypeScore, MQ, MQRankSum, ReadPosRankSum etc.) (cf. next paragraph). Based on the learned values, it splits the variant list into reliable and unreliable variant clusters (cf. Figure 2.8 (a)). These clusters are further used to compute the log odds ratio (known as VQSLOD) of being a true variant versus being a false call according to their distribution[49]. Higher or positive VQSLOD score (VQSLOD > 3) means that the variant call is more reliable and so this score can be used to filter FPs. However, the precise threshold should be selected according to data characteristics. In one of our data sets, we found that at higher thresholds there is a chance of missing true variants. Thus, we use a lower threshold to avoid such false negatives (VQSLOD > -8).

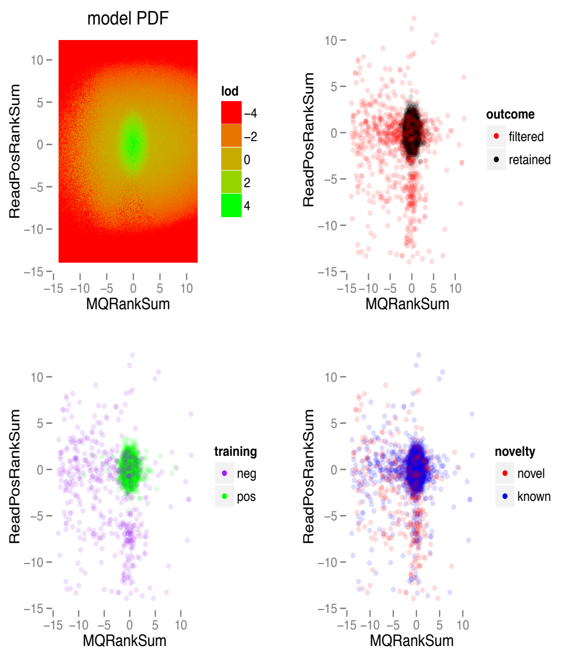
VQSR allows partition of the variant calls into quality tranches (cf. Figure 2.8 (b)). These quality tranches provide different thresholds that can be used to select a desired sensitivity or specificity relative to the truth set. There are four default values of tranches (90, 99, 99.9, 100), but these thresholds can be customized according to the filtering requirement. From 90 till 100 tranche, each tranche incorporates true positive calls as well as some false positives depending on the used threshold for tranche categorization. For example, tranche 100 means 100% sensitivity (i.e. the filtered list will contain 100% of known variable sites found in the truth set) but it can have many FPs. On the contrary, the 90% tranche is more accurate (less FPs than 100% tranche) but can miss some of the true variants. Thus, if the aim is to get a most comprehensive list then the highest tranche should be selected. The selection of a tranche should be based on the objective of the study or the filtering requirements. Moreover, tranche categorization also comes with Ti/Tv ratios (cf. Section 2.4.1) on novel variants in the list (cf. Figure 2.8 (b)), which can also be used for tranche selection. For example, lowering

[48] http://en.wikipedia.org/wiki/Mixture_model#Gaussian_mixture_model

[49] http://gatkforums.broadinstitute.org/discussion/39/variant-quality-score-recalibration-vqsr

the tranche value to a certain extent is safe as long as the Ti/Tv value is in a proper range (for exomes, the Ti/Tv should be around 2.8) (cf. Section 2.4.1).

a)



b)

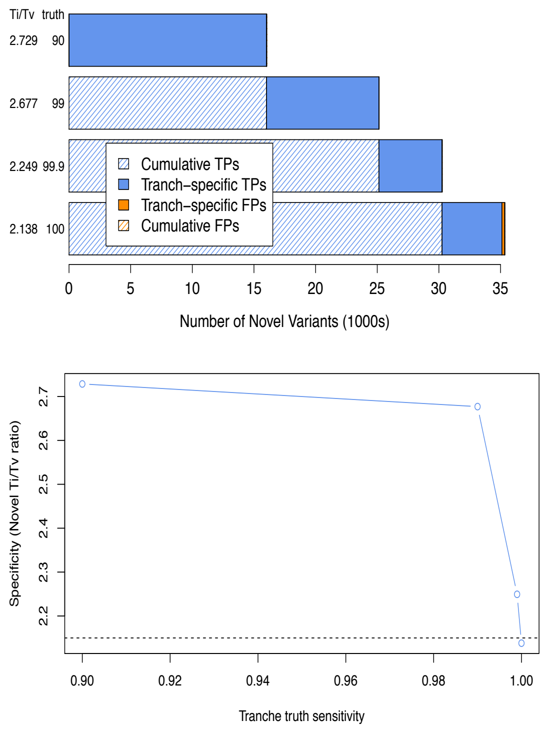


Figure 2.8 a) VQSR clustering based on 2 Annotations: ReadPosRankSum and MQRanksum. The upper left part shows the distribution of variants based on the scores of annotations varying from bad quality to good quality of variant (i.e. lod score range -4 to 4) (red colour to green colour respectively). Thus, the variants belonging to the green cluster have higher confidence than the variants having annotation values in the red part of figure (the higher the score the more reliable is the variant). Similarly, the other parts of the figure (upper right, bottom left) are also showing two different clusters and demonstrate which portion of the positive (reliable) and negative (unreliable) variants and portion of the should be filtered or kept after training, respectively. The figure at the bottom of right side is depicting the proportion of novel and known variants in the variant list. This figure can be used to judge VQSR, for example, if we observe distinct clusters of reliable and unreliable variants then that mean the variants are categorized properly (or up to certain extent). b) VQSR's four-default tranches and their correlation with sensitivity and Ti/Tv ratio. The upper part shows VQSR's four-default tranches and the lower part shows the correlation between VQSR's tranche sensitivity and the Ti/Tv ratio. Both figures (a & b) show that the gain in sensitivity of the variant list causes a significant drop in specificity (as well as Ti/Tv ratio) of the variant list and vice-versa. These diagrams are generated to for our control sample NA12878 and able to shows the performance of implemented VQSR filtering in our workflow.

VQSLOD is a reliable score for FPs filtering, but it requires a well trained model. Weak models can result in bad classification of the variants in a given call set, which could affect variant filtering and lead to many FPs or FNs. Thus, I tested VQSR and observed that the default parameter settings are not sufficient for training a good model (at least for our exome sequencing data). After several trial and error processes during testing (cf. Appendix) (including some recommendations from GATK), I found that the following facts should be considered during/before VQSR:

1. VQSR should be performed only on variants inside the target regions: Variants beyond the target regions may have bad quality (e.g. poor coverage, low mapping quality/accuracy etc.) and can damage or badly influence the model training. Thus, I performed VQSR only on variants belonging to the targeted regions of the sequenced experiment.
2. VQSR should have enough calls in the given variant list: As this is the learning procedure, it needs a sufficient amount of data (thousands of variant sites). Less data can result in a bad model, thus, in the case of less variant sites, more samples should be coupled with the sample of interest. I used 32 samples with multi-sample calling to generate a variant list with lots of variant calls. I used this list as a reference variant list with every sample of interest to have sufficient data for VQSR.
3. Model training parameters provided by the tool should be selected carefully: The default parameter setting might not be good for every data set, so the selection of parameters should be based on data characteristics (e.g. type of study, sequencing platform etc.) and the amount of data available for training. For example, with few variant sites, lowering the number of Gaussians ("-mG" parameter) can be helpful for training the model. I used –mG=6 for the VQSR training to achieve good performance on our data (default value is 8).

We also use a combination of some hard thresholds[50] to filter both SNPs and Indels as suggested by the GATK's best practice guideline[51], when the VQSLOD score is not available (e.g. Off-target variants). A combination (or individual application) of the following filters can discard a significant amount of the FPs caused the coverage bias, strand bias and alignment artefacts (or poor alignment):

1. Quality by Depth (QD): It is a normalized value of the variant quality score (Qual) by with respect to read depth. As Qual is calculated from the read depth, thus, the highly covered variants can have an over estimated Qual score. Therefore, normalization of Qual with respect to coverage provides a more realistic representation of the variant confidence. Low QD scores are indicative of false positive SNPs (or artefacts)
2. Fisher Strand (FS) score: Fisher strand score is a probability score from Fischer's Exact test to detect the strand bias. Strand bias means that a variant is only present in reads from one strand. Paired-end (or Single end) exome sequencing produces reads from both strands (forward or reverse), so a true variant should be present on reads from both strands. A higher FS score suggests that the SNP is mainly (most probably) supported by only reads from one strand, a typical signature of FPs.
3. Mapping quality (MQ): It is a Root Mean Square of the mapping quality of the reads and should be high for confident SNPs. Low value MQ signifies that the alignment accuracy at this position is not good and the aligned reads might be result of wrong alignment with some mismatches. Thus, the called variant can be just a mismatch due to bad alignment and should be discarded.
4. Mapping quality rank-sum test (MQRankSum): This score can be used to filter only heterozygous calls as this test performs the Mann-Whitney Rank Sum Test[52] for mapping qualities of the reads with reference bases vs. reads with the alternate allele. If the MQ of reads supporting the alternative allele is less than the MQ of reads supporting the reference allele, then these reads can be

[50] http://gatkforums.broadinstitute.org/discussion/2806/howto-apply-hard-filters-to-a-call-set

[51] https://www.broadinstitute.org/gatk/guide/best-practices?bpm=DNAseq

[52] https://en.wikipedia.org/wiki/Mann–Whitney_U_test

misaligned, which might introduce mismatches. These mismatches can easily be called as variants, which most probably are FPs.

5. ReadPosRankSum: Bases at the ends of the reads have a high probability of error, thus, a variant called from the end parts of reads can be a FP (cf. Section 2.1.2). The ReadPosRankSum test (i.e. Mann-Whitney Rank Sum Test) evidence of the positional bias, which reports the distance of alternate allele from the ends of the reads and should be used to avoid FPs at the read ends.

We use the following thresholds to filter false positives (for both SNPs and Indels):

- For SNPs: QD < 2.0, FS > 60, MQRankSum < -12.5, ReadPosRankSum < -8.0.
- For Indels: QD < 2.0, FS > 200.0, ReadPosRankSum < -20.0

The above-mentioned filters work best on Illumina data and should be used carefully for other technologies (cf. Section 2.4.2). The effect of these filters on variant list can be observed by evaluation of the sensitivity and specificity of the variant list (after filtering) (cf. Section 2.4). On good quality data these filters should not be harmful, but generalization of these filters (application on any data) might lead to some FNs. All of these filters and some other sample specific filtering strategies (as mentioned above) are implemented in our web-browser Varbank[53] (https://varbank.ccg.uni-koeln.de/). In Varbank, the user can apply different combinations of filters and his/her disease related knowledge to obtain a shorter list of interesting candidates.

[53] Developed by Dr. Holger Thiele

2.4 Evaluation of the variant list

After filtering or along the filtering process, we evaluated our results (variant list) to check the effect of the filters on the variant list as well as the sensitivity and specificity of our data analysis workflow by the following criteria:

2.4.1 Ti/Tv ratio

The Ti/Tv is the ratio of transition mutations, Ti, (between purines (A <-> G), or between pyrimidines (C <-> T)) to transversion mutations, Tv, (between purines and pyrimidines (A <-> T or C <-> G)) (Q. Liu et al., 2012). In general, transitions are more frequent than the tranversions (due to methylation of C in CpG islands), thus, this ratio can indicate how much your variant list deviates from general expectations. (DePristo et al., 2011) states that this ratio should be 0.5 for FPs and a good quality variant list for an exome should have a Ti/Tv ratio around 2.8. Therefore, this ratio can be used to evaluate the effect of different filters or their combination on the raw variant list (cf. Figure 2.8). For example, after filtering FPs, the ratio should increase and tend to reach the expected value, but if it is decreasing then the filter might not be good for this type of data. The Ti/Tv ratio can be calculated by vcftools (Danecek et al., 2011) or GATK's VariantEval method[54]. Both tools provide many other different filtering or variant evaluation methods (e.g. mis-sense/non-sense ratio, heterozygous/homozygous ratio, etc.). Besides these tools, we use our SQL server to compute such kind of information from the variant list.

54

https://www.broadinstitute.org/gatk/gatkdocs/org_broadinstitute_gatk_tools_walkers_varianteval_VariantEval.php

2.4.2 Benchmarking of variant lists with the GIAB dataset

The Genome in a Bottle (GIAB)[55] (hosted by NIST[56]) consortium generated a highly confident list of SNPs, Indels and homozygous reference genotype calls for the HapMap[57]/1000 Genomes[58] CEU female genome NA12878[59] by integration of data sequenced with different sequencing technologies and processed by different sets of data analysis tools/algorithms (Zook et al., 2014). This multi-resources data integration avoids almost all types of biases (sequencing bias, platform bias, alignment or variant calling bias, etc.), and provides a highly accurate list that can be used to compare in-house generated variant lists.

We performed exome sequencing for NA12878 and processed the sequencing data with our exome pipeline to check the sensitivity and specificity of the variant list compared to the list from GIAB. Different variant callers can report the variant in different notations; for example, one MNP (multi-nucleotide polymorphism) can be called as an MNP or as two or more different variants (SNPs) with different locations. Therefore, we first perform normalization of our variants by using the vcfallelicprimitives method of the vcflib tool[60], which (by default) splits MNPs into multiple SNPs. This tool also normalizes the GIAB list, thus it makes variant coordinates in both lists comparable. In order to compare lists, we used the VCF comparator function of the USeq tool[61] (Nix, Courdy, & Boucher, 2008). It compares the variants (both Indels and SNPs within the shared region, i.e. the target regions of the test sample) present in both variant lists and reports the number of matches and non-matches, false discovery rate (FDR), true positive rate

[55] https://sites.stanford.edu/abms/giab

[56] http://www.nist.gov

[57] http://hapmap.ncbi.nlm.nih.gov

[58] http://www.1000genomes.org

[59] https://catalog.coriell.org/0/Sections/Search/Sample_Detail.aspx?Ref=NA12878&Product=DNA

[60] https://github.com/ekg/vcflib#vcfallelicprimitives

[61] http://useq.sourceforge.net/cmdLnMenus.html#VCFComparator

(TPR), false positive rate (FPR) and positive prediction value (PPV)[62] at different quality thresholds (QUAL).

FDR=non-match/(match+non-match)

TPR=match/total shared variants in Control list (i.e. GIAB list)

FPR=non-match/total shared variants in Control list

PPV=match/(match+non-match)

Callers	QUAL	Match	Non-Match	FDR	TPR	FPR	PPV	Ti/Tv
UG RecalSNPs	none	23327	394	0.016609	0.995349	0.016811	0.9833903	2.81
	37.77	23269	245	0.010419	0.992874	0.010454	0.9895807	
UG HardFilt SNPs/Indels	none	24338	107	0.004377	0.986222	0.004335	0.9956228	2.87
	38.77	24322	103	0.004216	0.985574	0.004173	0.9957830	
HC RecalSNPs	none	23286	238	0.010117	0.993599	0.010155	0.9898826	2.85
	37.77	23250	150	0.006410	0.992063	0.006400	0.9935897	
HC HardFilt SNPs/Indel	none	24429	53	0.002164	0.989910	0.002147	0.9978351	2.88
	37.73	24416	52	0.002125	0.989383	0.002107	0.9978748	

Table 2.4 Benchmarking results of different variant lists of control sample NA12878 with GIAB dataset.

Besides measuring the accuracy and sensitivity of a variant list, this benchmarking can be used to compare variant callers or to monitor filtering strategies with respect to the achieved TPR and/or FPR. Thus, we compared the 4 different following variant lists with the GIAB list:

- UG HardFilt SNP/Indels: Both SNPs and Indels called by Unified Genotyper (UG) and filtered by hard filters suggested by GATK's best practice guideline.
- UG RecalSNPS: SNPs called by UG and recalibrated by VQSR.

[62] https://en.wikipedia.org/wiki/Positive_and_negative_predictive_values

- HC HardFilt SNP/Indels: Both SNPs and Indels called by Haplotype Caller (HC) and filtered by hard filters suggested by GATK's best practice guideline.
- HC RecalSNPS: SNPs called by HC and recalibrated by VQSR.

Moreover, we compared results with and without quality filtering (QUAL~= 38 and QUAL = none, respectively), to see the effect on sensitivity and specificity of variant lists. Table 2.4 shows the benchmarking results of the above-mentioned variant lists generated for the in-house control sample NA12878 with the GIAB dataset. With all different lists, we have achieved the expected Ti/Tv ratio, which is around 2.8 for exome sequencing data. Moreover, we achieved good sensitivity (varying from 99% till 99.78 %) and specificity (varying 98% till 99.78 %) for all four variant lists.

In general, the QUAL filter shows little improvement in specificity but can decrease sensitivity (in fractions). This is expected, as QUAL is the confidence of a variant call and can filter out some low confident variants. However, sometimes a true variant also can have low confidence due to certain sequencing biases and artefacts like coverage bias, paralogous alignments etc. Without QUAL filtering, both variant callers are showing similar performance. At QUAL > 37, HC gives better results in both aspects (TPR and FPR). The variant list from HC at this threshold shows improvement in specificity without compromising sensitivity. This indicates that the QUAL score is more reliable or accurate from HC, which is expected, as it does not only rely on alignments. It performs local realignments (assembly), which can avoid some alignments errors (cf. Section 2.3). Moreover, the numbers in the table clearly show that HC is better than UG in terms of both sensitivity and specificity. After the applications of both of the filtering strategies VQSR and hard filters, the variant list generated by HC is better and has slightly high Ti/Tv ratio than the variant list generated by UG. Overall, this benchmarking suggests that the HC is the better option compared to UG (if only one variant caller has to be selected), which is also recommended by the GATK's best practice guidelines.

2.5 Chapter summary

In this chapter,

- I presented the data analysis steps required to convert raw fastq files into a list of significant variants. First, quality control of the raw data should be performed and then the alignment of reads to the reference sequence followed by SNP/Indel calling. At last, proper filtering strategies and evaluations of variant calls should be applied.
- I presented the effect of quality trimming. By removing some bad quality bases during quality trimming, we can improve the overall sample quality and the alignment accuracy.
- I presented testing of BWA's different algorithms and showed, how sensitive alignment is helpful to align more reads but it can be devastating and might results in many FPs.
- I have also presented that the post processing of the aligned reads is necessary to avoid alignment artefacts or some systematic errors to control the FDR.
- I have also described two widely used approaches for variant calling: Bayesian Model and Haplotype based approaches. Usage of a single approach is not enough, thus, the variant calling should be done using both approaches to get a better trade-off between specificity and sensitivity.
- I mentioned some variant filtering and evaluation strategies. Different filtering can be applied to the raw variant list in order to filter false positives. Moreover, the effects of filters on variant list can be monitored by different evaluation strategies.

Chapter 3
Leveraging the power of high performance computing

"There are lots of diseases to uncover and DNA sequencing has become less expensive and more promising - so let's do the sequencing". These thoughts are very common nowadays, resulting in generation of huge amounts of data. Analysis of these data requires a series of analysis actions before relevant mutations can be found like data cleaning, sequence alignment, variant calling, etc. All of these actions are interconnected, build on each other, and usually involve running some tool or script. Hence, analysing sequencing data can require a significant amount of time and lots of manual work in the daily routine. In order to reduce the manual efforts and chances of manual errors during data analysis operations, automation of data processing is necessary. Moreover, analysis of large data sets like whole exome and whole genome sequencing data requires more storage and compute power for fast and efficient data processing than conventional desktop computers or servers usually provide. Thus, the implementation of automated next generation sequencing (NGS) data analysis workflows on high performance computing (HPC) clusters has become very essential.

HPC systems utilize the power of a large number of processors (cf. Section 3.2) to speed up the task which makes their architecture more complex and less stable As they are built up of different components, the failure of a single component (e.g. parallel file system or job scheduler) can destabilize the system as a whole. Moreover, if the HPC system is a multi-user environment or a shared cluster, the probability of failure increases. As it is hard to monitor and control each user's activity on the system, any single wrong operation can cause system instabilities. These instabilities can originate from an individual workflow and can affect other's operations or vice versa. Therefore, a workflow's stability, robustness and maintainability are as important as speed and should be considered carefully during the implementation of an automated workflow.

In this chapter, we will provide details of our dedicated solutions, applied during the development of our NGS data analysis workflow implementation on an HPC system. These solutions enable us to achieve high throughput in significantly less amount of time than the conventional ways. Moreover, they enhance stability, robustness as well as easy maintainability of our workflow. At first, we will provide an overview of our exome analysis workflow followed by a few details about our HPC system. Later on, details of the design principles applied during its implementation on the HPC system will be described. These design principles contributed significantly to making an automated (little manual intervention or debugging) and stable workflow for the HPC environment. The work in this chapter is the joint work of me and Dr. Susanne Motameny, which was published recently (Kawalia et al., 2015). All of the supporting information files (S1 to S6) mentioned in this chapter are available with the online version of this paper[63].

3.1 Exome analysis workflow

Our exome analysis workflow is a collection of open source third party tools and self-developed software that are stitched together into a pipeline via bash scripts. It is divided into several modules that combine analysis steps with the same purpose (cf. Figure 3.1). For a complete analysis, a single start of the workflow can run all modules. In addition, these modules can be started individually or in any combination for specific analyses. This workflow is fully automated where some modules run in a sequential manner and some run in parallel. It contains several checkpoints and waiting points between the modules (depicted as diamonds in Figure 3.1). If there is a failure at any checkpoint, the analysis is aborted. In this case, the workflow waits for other running modules (parallel ones) to finish and then exits with an error message.

In general, we perform a full analysis for every sample individually which runs in following steps (cf. Figure 3.1): The first module of workflow prepares fastq files by using in-house developed scripts. This includes merging of all available fastq files for the

[63]http://journals.plos.org/plosone/article?id=10.1371/journal.pone.0126321#sec036 (This and the subsequent URLs have been accessed on 25 August 2015)

sample (in-house script), quality check by FastQC (Andrews S, 2010), quality trimming (in-house script) and adapter trimming (Martin, 2011) (cf. Figure 3.1, (1)). The second and third modules take the prepared fastq files and start the alignment pipeline (cf. Figure 3.1, (2)) and structural variant calling pipeline (cf. Figure 3.1, (3)), respectively. Both pipelines first split the fastq files into smaller chunks and align these chunks (to speed-up the process) to the reference genome. For structural variant calling, a strict alignment allowing only perfect matches is performed using mrsFast (Hach et al., 2010) and then structural variants are detected based on discordant mate pair signatures by VariationHunter (Hormozdiari, Alkan, Eichler, & Sahinalp, 2009). The alignment pipeline performs sensitive alignment by using BWA (H. Li & Durbin, 2009, 2010), which can allow gaps or mismatches during the alignment. After the alignment, all chunks of BAM files are merged by samtools (H. Li et al., 2009) and PCR duplicates are removed by using Picard[64]. After that, post-alignment improvements, like, Indel realignment and base quality score recalibration (BQSR) are performed by using GATK (DePristo et al., 2011; McKenna et al., 2010). To speed up the variant calling and some other processes, the BAM file is split into 25 BAM files by samtools (one for each chromosome and the mitochondrion). To get a comprehensive variant list, both SNP/Indel and copynumber calling is performed by a set of four different tools. Samtools mpileup, GATK UnifiedGenotyper and GATK HaplotypeCaller, and Platypus (Rimmer et al., 2014) (cf. Figure 3.1, (6)) are used for SNP/Indel calling. Copynumber variants are called by CoNIFER (Krumm et al., 2012), XHMM (Fromer et al., 2012), cn.nops (Klambauer et al., 2012), and ExomeDepth (Plagnol et al., 2012) (cf. Figure 3.1, (4)). Furthermore, to get an overview of sample quality, an in-house developed script is used to compute statistics about exon coverage throughout the genome (cf. Figure 3.1, (5)) and Picard is used to compute some other statistics (e.g. about the alignment and enrichment) (cf. Figure 3.1, (7)). In order to distinguish between good and bad quality SNPs, variant quality score recalibration (VQSR) is performed. As VQSR needs some annotations that are only produced by GATK's caller, it is only performed on variant lists generated by GATK (cf. Figure 3.1, (8)).

[64] http://broadinstitute.github.io/picard/

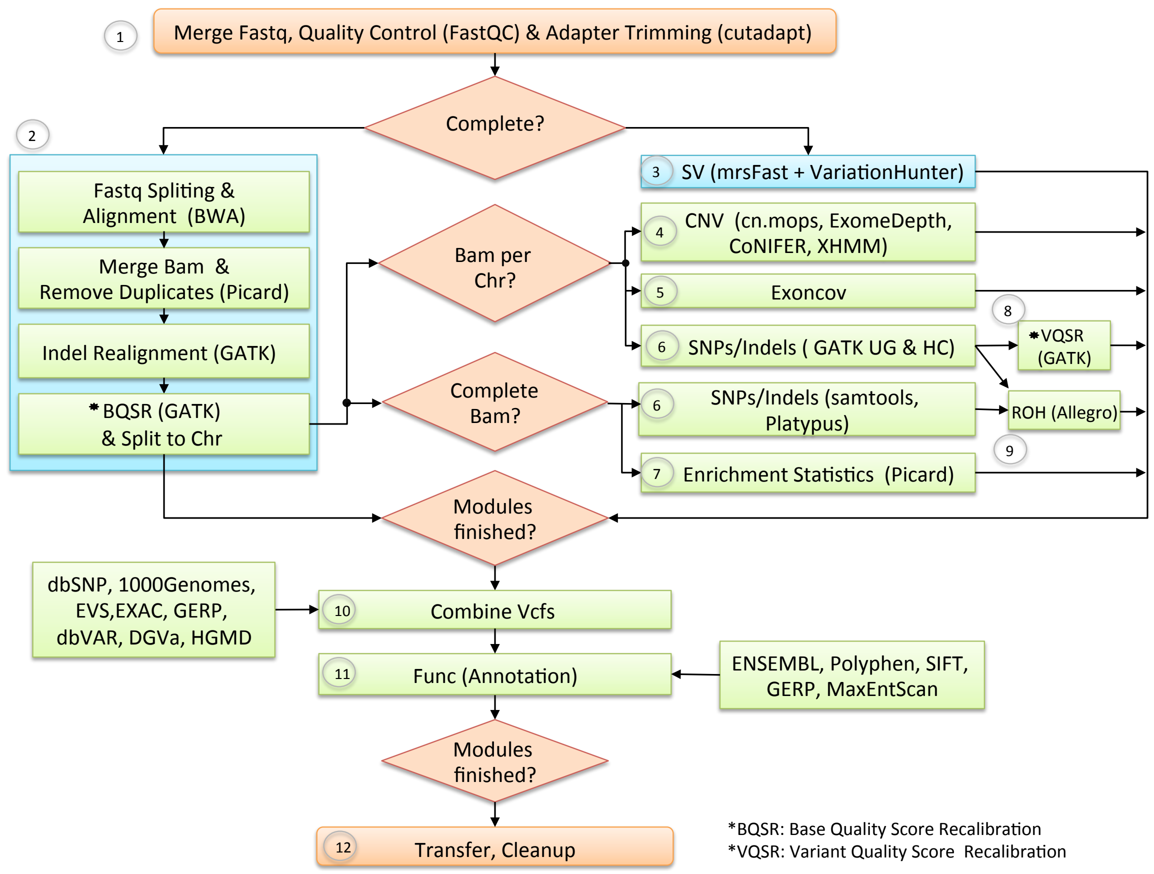

1
Merge Fastq, Quality Control (FastQC) & Adapter Trimming (cutadapt)
Complete?
2
Fastq Spliting & Alignment (BWA)
Merge Bam & Remove Duplicates (Picard)
Indel Realignment (GATK)
*BQSR (GATK) & Split to Chr
3
SV (mrsFast + VariationHunter)
Bam per Chr?
4
CNV (cn.mops, ExomeDepth, CoNIFER, XHMM)
5
Exoncov
6
SNPs/Indels (GATK UG & HC)
8
*VQSR (GATK)
Complete Bam?
6
SNPs/Indels (samtools, Platypus)
ROH (Allegro)
9
7
Enrichment Statistics (Picard)
Modules finished?
dbSNP, 1000Genomes, EVS,EXAC, GERP, dbVAR, DGVa, HGMD
10
Combine Vcfs
11
Func (Annotation)
ENSEMBL, Polyphen, SIFT, GERP, MaxEntScan
Modules finished?
12
Transfer, Cleanup
*BQSR: Base Quality Score Recalibration
*VQSR: Variant Quality Score Recalibration

Figure 3.1 Automated workflow for exome sequencing data analysis. The workflow has one starting and ending point showed in orange colour. Green colour boxes are the modules or tasks of the workflow executed by the master script or sub-pipeline (highlighted by blue colour boxes). Red colour diamonds are control points that check for completion of previous tasks to make a decision about the next task's execution.

Regions of homozygosity (ROH) are detected using Allegro (Gudbjartsson, Jonasson, Frigge, & Kong, 2000) Figure 3.1, (9)). For detection of denovo mutations, we use deNovoGear (Ramu et al., 2013), which runs on exome data from the affected child of a trio or the affected twin of a sibling pair or the tumour of a tumour-normal pair. After that, the results produced by all above mentioned analysis steps are collected and combined into one table (self-written script[65], Figure 3.1, (10)). This script screens several databases (dbSNP (Sherry et al., 2001), 1000 Genomes Project (Abecasis et al., 2012), Exome Aggregation Consortium (ExAC)[66], dbVAR and DGVa (Lappalainen et al.,

[65] This script is written by Dr. Holger Thiele

[66] http://exac.broadinstitute.org

2013), GERP (Davydov et al., 2010), ENSEMBL (Flicek et al., 2014), and the commercial HGMD professional database (Stenson et al., 2014) to annotate known variants in the combined variant list. For functional annotation of variants, POLYPHEN (Adzhubei et al., 2010), SIFT (Kumar, Henikoff, & Ng, 2009), and in-house developed algorithms are used. Additionally, splice site analysis, based on the framework described in (Yeo, Burge, Liebert, Yeo, & Burge, 2004), is performed. At last, the annotated variant list, sample statistics, fastq files and BAM files are transferred to our storage server and the intermediate results are deleted (self-written scripts, Figure 3.1, (12)).

3.2 HPC system

The workflow is implemented on the HPC clusters CHEOPS and SuGI of the Regional Computing Center Cologne (RRZK). These are two main clusters of the University of Cologne, serving the computational demands of many researchers from different scientific fields. CHEOPS is the larger compute cluster with a peak performance of 100 Teraflop/s and linpack performance of 85.0 Teraflop/s. It has 841 nodes with 9712 cores and 35.5 TB RAM in total for computation, and provides 500 TB Lustre parallel file system for storage. On the other hand, SuGI is smaller and has 32 compute nodes with 256 cores and 1 TB RAM in total. It provides 5 TB Panasas parallel file storage and can achieve a peak performance of 2 Teraflop/s. Both clusters run a Linux operating system.

Due to their complex architecture (cf. Figure 3.2) HPC systems are more susceptible to system instabilities than ordinary hardware. This implies certain working rules for both of HPC clusters. Any computation should not be directly run on the login nodes (or frontend nodes). Frontend nodes can be either used to submit a job that contains all computation or used for login and data transfer to (or from) the parallel file system. Each job can be submitted via SLURM (Jette & Grondona, 2003) or TORQUE/Maui batch system on CHEOPS or SUGI, respectively. Every job submission contains a request for appropriate computational resources, like number of cores, required memory, and runtime etc. Based on the requested resources, the batch system's scheduler assigns an appropriate compute node to the submitted job.

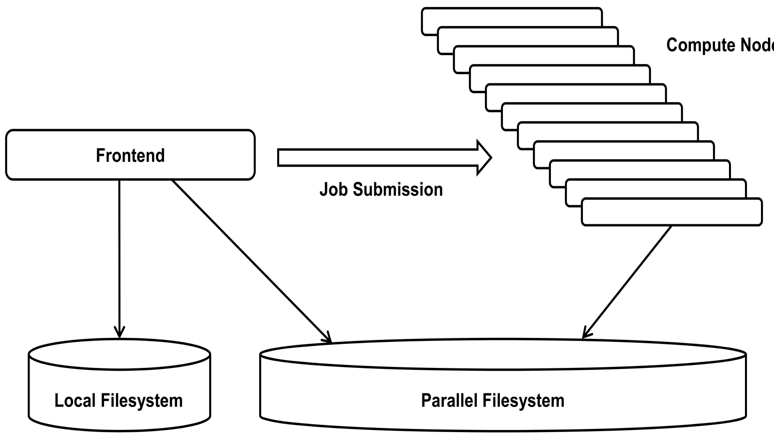


Figure 3.2 Overview of HPC architecture.

3.3 Technical components of workflow

3.3.1 Workflow overview

We considered all HPC working rules during the development of our exome workflow. It is a collection of BASH scripts that are mainly divided into one masterscript and several jobscripts. The masterscript executes all tasks of the workflow by submission of the job scripts and also monitors their execution (cf. Figure 3.5). It processes each sample individually and gets all information related to the sample form a configuration file in XML (Extensible Markup Language[67]) format, which is generated from our LIMS (Laboratory Information Management System[68]) database. The LIMS keeps track of all samples and their processing in the wet-lab and contains all necessary information relevant to the analysis. The masterscript provides the required information (according to task) to all job scripts, which then do the actual computation. The masterscript also prepares the results (produced by the job scripts) for transfer to our storage server from where they are further uploaded to an Oracle database (cf. Figure 3.3). Researchers can fetch information from this database via an in-house developed webinterface called “varbank” (https://varbank.ccg.uni-koeln.de), which provides access and download

[67] http://www.w3.org/TR/2006/REC-xml-20060816/

[68] https://en.wikipedia.org/wiki/Laboratory_information_management_system

options for the fastq-, BAM-, and VCF files as well as annotated variant tables and coverage statistics.

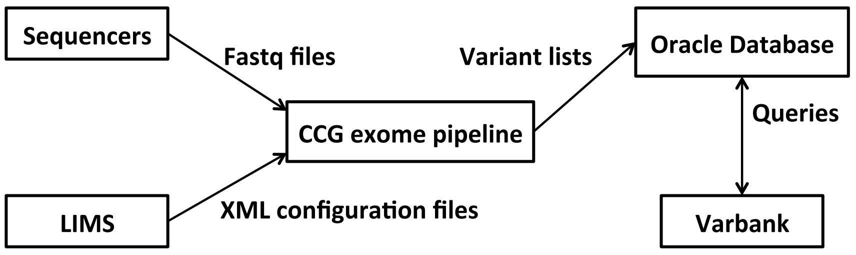


Figure 3.3 Infrastructure surrounding the exome workflow.

3.3.2 Workflow interaction with HPC systems

As mentioned above, the workflow is designed to run on both HPC systems: CHEOPS and SuGI. This is beneficial in two different ways: first to increase the throughput when data analysis is running on both systems in parallel. Additionally, one system can serve as a backup system in case of the failure of the other. For example, CHEOPS is a lager cluster and more complex which makes it more prone to system instabilities. Thus, during downtime of CHEOPS or maintenance periods, SuGI can be used for data analysis. In order to run the workflow on both systems, 3 different directories are shared between these systems: XML stack, Data and Results directory (cf. Figure 3.4). All of these directories are located in the Panasas file system of SuGI and mounted on CHEOPS. The XML stack contains XML configuration files of new samples, which need to be analysed. The data directory contains the required fastq files for analysis and the results directory stores the results for each analysis step like BAM files from alignment or VCF files from variant calling etc. Every system has its own scripts directory and status directory in the local file system, which contains all analysis scripts (including master scripts) and status messages generated by the workflow, respectively (see next paragraph). Each system has its own scratch directory mounted in its parallel file system (Lustre for CHEOPS and Panasas for SuGI). These scratch directories store temporary results (or intermediate results) and status messages generated from different analysis steps executed on the respective HPC system (see next paragraph).

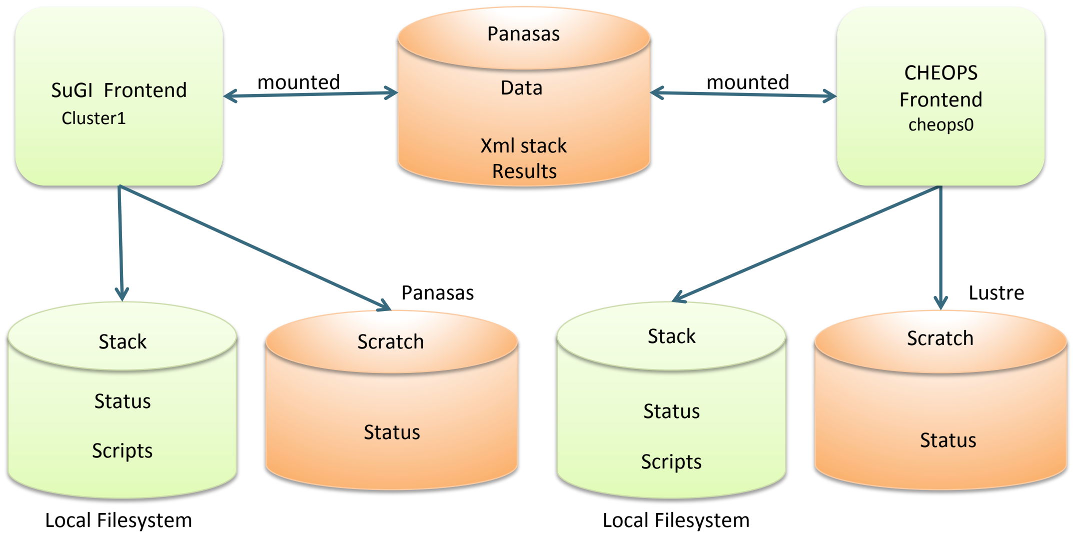


Figure 3.4 Overview of workflow interaction with both CHEOPS and SUGI HPC systems.

The workflow execution works in the same way for both HPC systems. The cron deamon on the frontend node of both systems checks the presence of new samples in the XML stack directory. When a sample is available in the stack directory for processing, then the respective cron deamon starts an instance of the masterscript on the respective system (cf. Figure 3.5). The cron deamon of both systems takes turns to check stack directory and if number of running samples are less than the set limit (e.g. < 20 samples can run in parallel on Cheops) of the respective system then it starts the masterscript for the new sample. The general flow of job submission from the masterscript is shown in Figure 3.5. The masterscript submits the job (compute job) for an analysis task to the compute nodes and waits for its completion. The Compute job gets the required data from the data directory and puts intermediate files (generated during computation) to the scratch directory. After completion of a task, the results are stored in the results directory. The status messages for a task (i.e. successful completion or an error), are stored in the status directory of scratch (where other compute jobs can pick them up) and in the status directory of the local file system (where the masterscript can access them). The masterscript checks these messages and reacts accordingly: either it resubmits the job (in case of failure) or submits the next compute job. This whole process of job submission and monitoring goes on till the complete analysis is finished. The cronjob starts multiple instances of the masterscript with some delay when more than one sample is available in the XML stack.

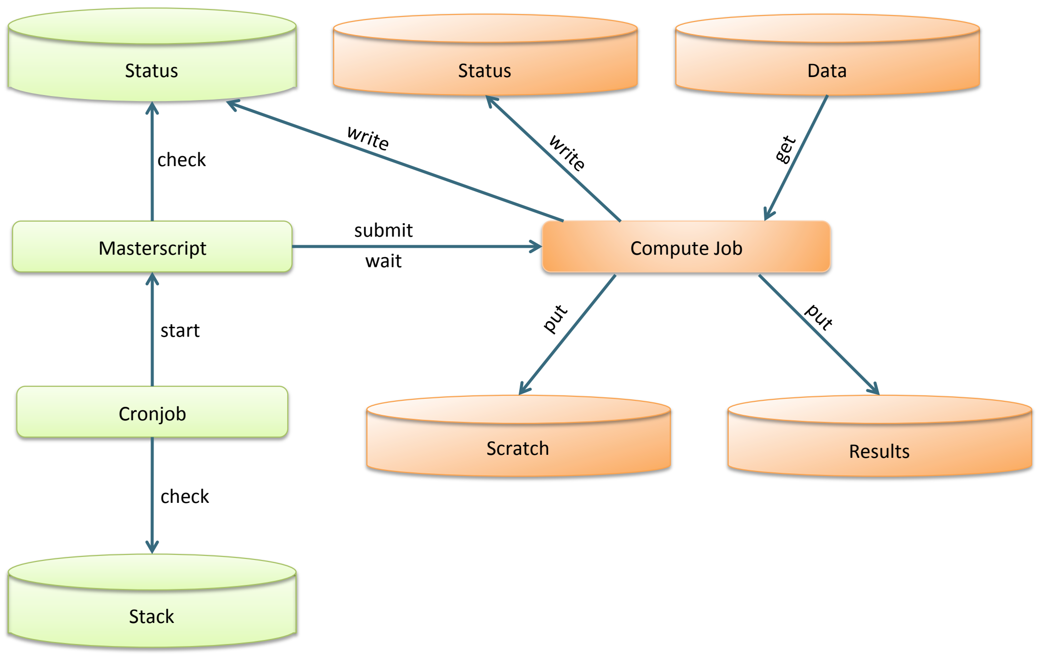


Figure 3.5 Overview of workflow implementation at HPC.

3.3.3 The masterscript

As implied by its name, the masterscript is the main script of the workflow that organizes the data analysis tasks on the HPC clusters. It submits jobscripts (which perform the actual computation) on the clusters and monitors their progress. It processes a single data set (e.g. an exome sample) at once, but several instances can be started simultaneously to achieve high throughput. When more than one sample is available for processing, the cron deamon on the frontend node starts multiple instances of masterscript in parallel (cf. Figure 3.5).

The masterscript has a modular structure that means it puts all tools that perform similar tasks into one module. When a complete exome analysis has to run then the masterscript executes all modules as shown in (cf. Figure 3.1). However, execution of a single module, due to its failure or reanalysis, can be selected by a command line option of the masterscript (Supporting Information S5[69]), which saves a significant amount of time in these cases. Moreover, when some specific analysis is required then the

[69] This supporting information file and other files (S1 to S6) are available with online version of our paper (Kawalia et al., 2015): http://journals.plos.org/plosone/article?id=10.1371/journal.pone.0126321#sec036

masterscript can be started with only the desired modules. For example, we skip the SV module for samples having single reads, as it requires paired-end information for SV detection.

3.3.4 Jobscripts

The jobscripts are bash scripts that contain the commands to execute some tool or some script for a specific analysis task. Moreover, they can perform some data management like preparing data for analysis or merging results etc. To monitor the completion of any task, every jobscript writes a status message based on its exit status. This status message is transferred to the local file system on the frontend node and used by the masterscript to monitor completion or failure of that task. The masterscript checks the status message at checkpoints and in case of failure it aborts the workflow with an error report. If the batch system aborts the job due to too small walltime or memory requests, then this status message will not be written. Therefore, this strategy can distinguish between job abortion/failure errors and execution errors.

3.3.5 Job submission

The job submission function is the main function in the masterscript. It encapsulates job submission and also monitors the status of the job and checks its successful completion or failure (cf. Figure 3.6). If job submission fails due to the unavailability of the batch system at first attempt, then it tries resubmission for two more times, after that it reports an error for an unsuccessful job submission. After successful submission, it waits for the completion of job. In the case of job failure (if the resource limit was exceeded), it increases the requested resources (runtime and memory) and re-submits the job automatically. If this job fails again then an error is reported by this function. It also supports both batch systems: SLURM and TORQUE/Maui running on CHEOPS and SUGI, respectively. The masterscript submits a job with a general job submission syntax that is further translated (by the job submission function) into job submission commands of the used batch system. The code of the job submission function can be found in the Supporting Information S1[69].

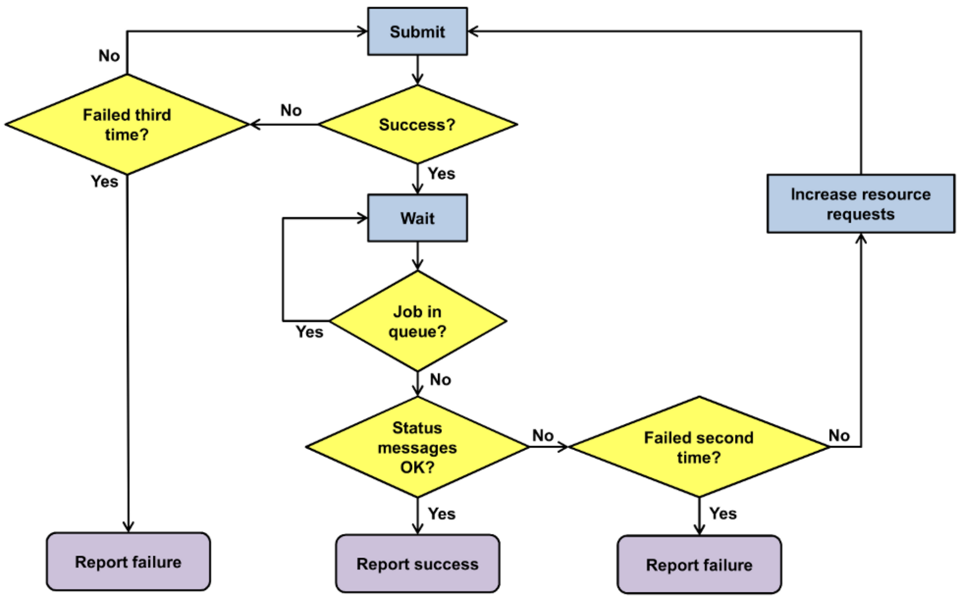


Figure 3.6 Flow chart of the job submission function.

There are two different modes of job submission: sequential or parallel. In the sequential manner, a job is submitted by calling the submission function directly from the masterscript, causing the masterscript to wait until it is finished before proceeding with the next job submission. This manner of sequential job submission is utilized when jobs depend on each other. On the other hand, some jobs that do not depend to each other, they can be submitted in parallel. In this case, a child process started by the masterscript in background that calls the job submission function. For example, the masterscript performs a sequential job submission for fastq preparation and quality control tasks as this is the first task and all other tasks depend on its outcome. After the completion of this job, the alignment jobs for SNP/Indel calling and SV detection are submitted in parallel via sub-processes, as they are not depending on each other.

3.3.6 Job monitoring

According to the different tasks and their dependencies, the masterscript contains some checkpoints and waits for the completion of submitted jobs and child process at every checkpoint. It stores all job ids assigned by the batch system and all process ids of started child processes. The submit function keeps track of jobs via their job-ids and after job completion, it checks the status message returned from a job (i.e. error or finished). The masterscript stores these status messages from the submit function at the local file system and waits for all computations corresponding to a specific checkpoint to finish. As soon as all computations are finished, masterscript checks the stored job status messages and list of process ids. In case of errors or failures, it aborts the workflow with an error message reporting the failed modules. If the checkpoint reports successful completion, then the masterscript performs the next round of job submissions for the remaining tasks. In the case of any interruption or abortion of the workflow, the masterscript cancels all submitted jobs and kills all child processes by using a cleanup function. This cleaning is triggered by the “trap” statement and executed whenever the masterscript exits. This function also cancels jobs submitted by child processes. This strategy ensures a clean exit and frees already allocated resources. Details about this function can be found in the supplementary file S3[69].

3.4 Design principles of workflow

In order to develop an efficient workflow, we focused on the following four design principles:

1. Speed: Optimum usage of HPC systems to speed up data analysis.
2. Stability: Appropriate handling of the HPC environment to prevent system instabilities.
3. Robustness: Automatic detection and correction of some processing errors
4. Maintainability: Easy to maintain and expand.

3.4.1 Speed

Parallelization by jobarrays

Parallelization is one of the most frequently used approaches for exploitation of the compute power of clusters. In our workflow, we apply “parallelization by chunks”, which is splitting of large data into small chunks followed by a parallel processing of every chunk individually. Mapping of reads is an independent process for every read, thus, we split the fastq file into several chunks and perform the alignment process on each chunk by using BWA and mrsFast. Although, BWA provides multi-threading to achieve more speed, however, we cannot rely only on threading due to limited resources of our smaller cluster SuGI. It has only 8 core nodes, so BWA can only use a maximum of 8 threads. Besides workflow compatibility to both the smaller and bigger cluster, using a combination of threading (with fewer cores) and parallelization by chunks works best for us (esp. in terms of resource usage optimization). We also use this strategy for other tasks like Indel realignment by GATK, SV calling by VariationHunter and denovo variant calling by deNovoGear. For these applications, we split the BAM file by chromosome (similar to (Lam et al., 2012; Puckelwartz et al., 2014)) and run the analysis for each chromosome in parallel.

We are using jobarrays to perform parallelization by chunks. A jobarray is a collection of jobs (executing the same task for different data chunks) submitted at once and can be tracked by a single jobid, which simplifies the job submission and monitoring. It can be used for tasks where the same computation has to be run for different input files. The size of a job array and runtime is directly proportional to number of chunks and size of chunks. In order to take optimum advantage of jobarrays, selection of a moderate array size and runtime is required. If too many jobs are submitted via a job array (means too many chunks of data), then a single task can finish very fast, but some jobs have to wait in the queue. On the contrary, if few jobs are submitted (means big chunks of data), then processing time will be longer and little gain in speed would be achieved by parallelization. Therefore, we implement jobarrays in a way that a single job should run at least one hour and all jobs (or tasks executed by jobs) should be finished in similar time. As the completion of a jobarray depends on the completion of all jobs it contains and if one task is running longer than other tasks, then the jobarray has to wait and parallelization power would not be exploited properly. Therefore, when defining the data chunks on a per-chromosome basis, we distribute the 24 chromosomes across 14 tasks of a jobarray as mentioned below:

- chr 1 to 7: each chr in a single task
- chr 8 to 17: combined two chromosomes per task
- chr 18,19 and 20 processed in one task
- chr 21, 22, X and Y processed in one task

After some trial and error, we found that this strategy works satisfactorily for our workflow. However, the chromosome size is not the only influential factor for run time of the individual array task. Other factors like coverage or mutation load can change runtime significantly. Thus, getting the optimum combination of chromosomes is almost impossible and it can vary in different data sets.

Parallelization by threads

Parallelization can also be done by multi-threading, where a process brakes into multiple threads that run on different cores simultaneously to reduce the overall completion time. BWA, GATK and Picard come with multi-threading options and we utilize this to speed-up their runtime. In general, the number of threads should be inversely proportional to the runtime of the process, but not all tools follow this principle. We measured the walltime of BWA and GATK with different numbers of threads (on a compute node with 4 Nehalem EX Octo-Core Processors, Xeon X7560, 2.27GHz). BWA shows a significant decrement in walltime with higher number of threads. However, GATK haplotype caller's walltime does not decrease significantly beyond 4 threads (cf. Figure 3.7). We are still investigating the reason behind this behaviour but believe that this is due to the Java implementation of GATK. We have observed that Java applications produce an overhead of I/O operations that limit their acceleration capacity on the HPC system. Moreover, GATK VariantRecalibrator method generates many intermediate output files (so called vcf stubs), when it runs with more than one thread. For an exome analysis, the number of these files can easily reach up to 100000 and can exceed the maximum number of files quota of file systems (most of the file system have this quota). This limits the number of parallel analyses and reduces the throughput of the workflow. To avoid this scenario, we run GATK VariantRecalibrator with one thread, which still runs fast without producing intermediate files.

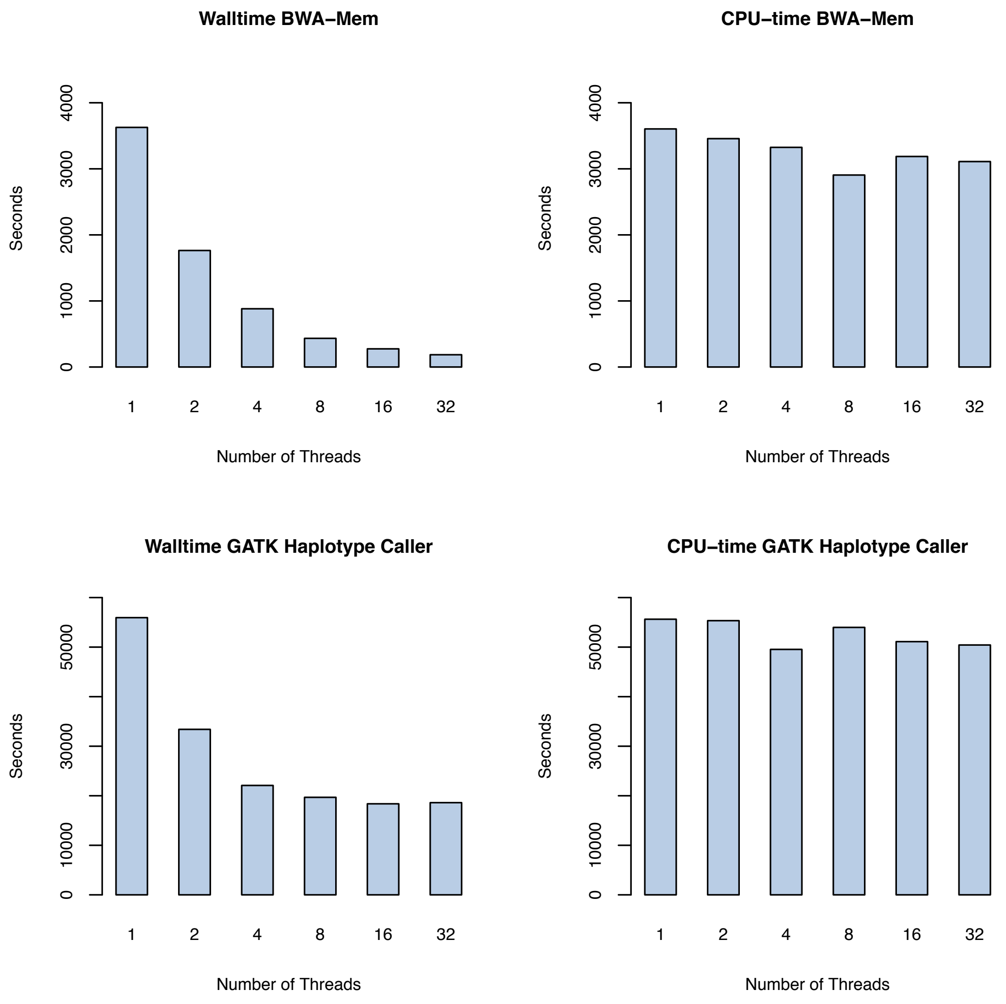


Figure 3.7 Performance of multi-threading with BWA-MEM and GATK HaplotypeCaller. Left-side histograms show the decrement in walltime usage of both tools with more threads. Whereas, histograms on the right side show the CPU-time with different number of threads.

Scalability

Scalability is an important measure of the efficiency of an application when it is used on large data with HPC. It can be categorized into strong and weak scalability. Strong scalability measures how the runtime of an application changes with the number of cores for a fixed problem size (BWA and GATK HaplotypeCaller shows strong scalability (cf. Section “Parallelization by threads"). Weak scalability measures how the runtime varies when a fixed amount of work is directed to a single core but more cores are used due to a larger problem size. We use “parallelization by chunks” for BWA and mrsFast, and these modules scale almost perfectly in the context of weak scalability. This is because when splitting of data into chunks of equal size the processing time is equalized for every chunk, thus, the larger problem can be processed on multiple cores in comparable time to the processing of smaller problem size on a single core. Therefore, these parts of the workflow take almost the same amount of time for data sets of different sizes (not including queuing time). However, assessing the scalability of the entire workflow is difficult as most of the parts of workflow use a fixed number of cores.

Resource usage optimization

Optimum usage of computational resources, i.e. number of cores and random access memory (RAM) is the key to exploit the power of HPC systems to achieve the desired performance. In our case, a single analysis should be fast enough and high throughput can be achieved by running several analyses in parallel. With this objective, we performed several trial and error steps, as there are no fix rules for resource optimization to achieve a satisfactory performance. However, there are three main points that need to be considered to optimize the resource requests:

Selection of job size

In general, based on their resources requirements, jobs can be categorized into two main categories: large jobs and small jobs. Large jobs use a high number of cores and much RAM while the small jobs need only few cores and little memory. In practice, large jobs have longer queuing time and can waste the allocated memory (i.e. granted exclusively to that job), if it is not completely used by the running task. Moreover, if there are many jobs requesting more memory than they actually require, then the cluster becomes partly idle while jobs have to wait in the queue. Therefore, assessment of the required memory and number of cores is very important and resources should be requested as close as to actually required resources.

Balanced memory usage for jobarray tasks

A jobarray is a collection of jobs, either executing the same analysis task for different data chunks or can perform different analysis tasks in different jobs. The resources requested during job submission are valid for every job of a jobarray. However, some jobs in a job array can behave differently and use more memory than the other jobs (either due to different analysis task or due to the different chunk of the data). In this scenario, a jobarray should be submitted first with a low memory request and later on, the failed jobs could be resubmitted with higher memory requests. This strategy prevents unnecessary blocking of resources, thus more analyses can run in parallel with less job queuing time.

Resource requests based on cluster architecture

The computing cluster architecture is also another important criterion for resource selection. In order to exploit the optimum capacity of the compute nodes, the number of requested cores should preferably be a divisor of the number of cores available on the nodes. For example, we run jobs with 1, 2, or 4 cores in workflow so we can fill 8,12, and 32 core nodes of our cluster CHEOPS properly. On the other hand, if we submit a job with 5 cores, then we would end up with 3 or 2 cores (depending on node type) that cannot be used for another job of the workflow. Besides the number of cores, a similar logic should be applied during memory (RAM) selection and it should be fragmented according to the RAM of a node. For example, our cluster has 24 or 48 GB RAM for the majority of nodes and a few nodes are having 96 or 512 GB RAM. Thus, theoretically, a 24 GB node can be filled with 8 jobs requiring 3 GB RAM each or 6 jobs requiring 4 GB RAM each. However, not the full nominal amount of memory is really available to the jobs. Therefore, a small reduction in the memory requested is required to fill a node completely. In our implementation, we are requesting 5 % less memory for each job. For example, we submit a nominal 4 GB job with a 3.8 GB RAM request and only 7.6 GB RAM request for a nominal 8 GB job.

With our current parallelization strategy, we fit 173 hours average CPU time into an average runtime of 21 hours per exome. In comparison to an exome analysis on a single core computer, we are able to speed it up more than 8-fold by using the HPC systems. This is not a very high gain in speed but it can be increased by trading with the workflow’s throughput. We optimized our workflow for high throughput rather than the speed, as it fits better to our requirements. By starting an exome analysis every 30 minutes, we can analyse 290 exomes per week, which is sufficient for our current sequencing capacities as well as for the processing of external data. Moreover, it should be noted that the total load of the HPC cluster also hinders both speed and throughput, which is not completely controlled by us. Nevertheless, if required in the future, the workflow can be further parallelized and more analyses can be run in parallel.

3.4.2 Stability

The stability of the HPC environment is also another important design principle for the workflow development. On multi-user HPC systems, a certain etiquette should be followed to avoid system instabilities and to maintain the smooth operation for other users. We included the following measures that contribute to the stability of the cluster and our workflow:

No computation on frontend node

The frontend node of our clusters, which usually has less computational power, is mainly for login, data transfer and job submission. Running computations on the frontend node can increase the load to a level that causes the system to be unresponsive or even fail. Therefore, our masterscript only does organizational work on the front node and submits jobscripts for all computations.

Controlled accesses to the parallel file system

If the parallel file system is not responding due to any reason then it should not be accessed by the workflow. Otherwise a build-up of queries to the parallel file system can occur that can hinder the debugging process initiated by system administrators. Therefore, we are using a lockfile mechanism (Supporting Information S2[69]) that prevents the automatic execution of file system accesses during its unresponsive state. Before every access to the parallel file system (for example, file listing (by `ls`) or directory creation (by `mkdir’), the masterscript sets a lockfile and removes it after the file system access finished. If the file system is hanging (i.e. the access command does not return), then the masterscript does not delete this lock file and it stays in the local filesystem on the frontend node. On the contrary, if the parallel file system is responding then these lock files are deleted within a few seconds (less than 5 seconds). At every new start, the masterscript checks for the existence of the lock file. If the lockfile is present and persists for 25 seconds, then the masterscript exits. This strategy provides an automatic shut-down of our workflow, when the file system is not responding.

Shut-down switch

In case of cluster maintenance or downtimes, a workflow termination is required. This can be activated by the HPC administrators via the workflow's shut-down switch. It is implemented as a semaphore file in the local file system of the frontend node (Supporting Information S6[69]). Moreover, it can also be activated during system instability caused by workflow. If the shut-down switch is activated (i.e. if the semaphore file is present in the local filesystem), then the masterscript exits right at the beginning or if the workflow is already running, then it exists at dedicated exit points.

Delayed start of multiple runs

Our workflow processes new data in 30 minutes intervals. This delayed start is controlled by the cron deamon, which starts one instance of the masterscript every 30 minutes. This strategy prevents overloading of the scheduler due to too many simultaneous job submissions. It also prevents creation of too many lock files by the masterscript when accessing the parallel file system. Moreover, it makes the workflow more efficient, with respect to the throughput, by balancing resource usage of the workflow. The workflow contains small jobs and large jobs with lower and higher resource requirements respectively. Thus, this delayed start avoids simultaneous submission of large jobs and prevents a subsequent increase of queuing time.

No access to remote servers

To process a sample, the workflow requires some sample related information that is stored in the LIMS. Accessing remote servers (via ssh) to query the LIMS from the masterscript can get stuck in non-returning ssh commands. Thus, we do not access remote servers from the masterscript and perform LIMS querying on a separate server. After querying, all sample-related information is stored in an XML configuration file that is uploaded to the cluster together with the sequence data. Moreover, the transfer of the results to the Oracle database and varbank webserver, are also organized in the similar way. The masterscript writes a transfer XML file that contains the location of the result files. Then, a script running on another server performs the data transfer via scp by using the information present in the transfer XML file. We also do not access remote servers via job scripts because the outgoing network connections of the compute nodes are reserved for the exchange of software license information only on our HPC clusters. Moreover, access to remote servers can slow the computation and should be generally avoided in HPC workflows.

Automatic deletion of results

Every run of a sample generates a huge amount of data including input data, results and temporary files. Temporary files are generated from one process and are usually required as input data for subsequent processes and can occupy a great amount of space. Thus, these temporary files are automatically deleted after successful completion of the analysis. However, in case of workflow failure these files remain stored for debugging purposes. Similarly, the largest part of the results i.e. BAM files and input fastq files are transferred to the database and webserver after successful completion of workflow. To make sure that the transfer is completed successfully, we compare md5 checksums[70] of the original and the transferred files. As soon as the transfer completes, BAM files and fastq files are deleted from the cluster. The remaining part of the results, i.e. VCF files, statistics files, and variant tables are stored in a dedicated results directory on cluster.

3.4.3 Robustness

An automated workflow should run without manual intervention as long as possible. It should not stop if a single job fails and should be able to run all other modules that do not depend on the failed job. Moreover, it should be able to cope with errors that occurred during the execution in order to produce the most complete result. We applied the following strategies in the implementation to make our workflow robust.

[70] http://tools.ietf.org/html/rfc1321

Dynamic job submission

Two frequent reasons of workflow failure are the overload of the scheduler and job abortions due to lack of resources. If the scheduler is already having a significant load and lots of jobs are submitted simultaneously, then the scheduler can reject all or some job submissions. In general, these situations are resolved after a short time. Thus, to cope with this type of failure, the masterscript retries a failed job submission twice after waiting a few minutes. When the job is aborted due to more required computational resources (memory or runtime) than requested at submission time, the masterscript detects these jobs and resubmits them. This tracking of aborted jobs is managed via status messages written by the jobscripts after their completion. After a job submission, the masterscript (or the child process in case of several jobs started in parallel) checks the status of the job by querying the batch system's queue (cf. Supplement S1, wait_for_job function). These queries are only performed in every two minutes to avoid overloading the scheduler. As soon as the job disappears from the queue, the masterscript (or child process) checks the generated status messages. If the status messages are missing it means that the job did not finish regularly and was aborted by the scheduler. In this case, the masterscript (or child process) automatically resubmits the aborted job with increased memory and runtime requests. Similarly, jobs with an error status message are also resubmitted. If the job fails again after resubmission, the masterscript throws an error message and at this point manual intervention is required (cf. Figure 3.6).

Clean exit

The masterscript keeps track of all submitted jobs and running child processes. If the workflow is terminated at some point during the analysis (e.g. by a kill command), the masterscript automatically kills all child processes and deletes all submitted jobs. It also writes the appropriate status messages (Supporting Information S3[69]) in the logfile. It is extremely important to ensure such a clean exit. Otherwise, jobs run unsupervised and can potentially overwrite results when the workflow is restarted for the same sample again. Moreover, status messages (written in the logfile and status table) reporting the interruption are useful for debugging purposes.

3.4.4 Maintainability

Modularity

The masterscript is constructed from different modules for different kinds of analysis (alignment, SNP/Indel calling, CNV calling, SV calling, enrichment performance, functional annotation). All modules can be run separately or in any combination, which gives flexibility to the workflow. Thus, the workflow can be used for different types of data analysis like data generated from genome, exome or other targeted experiments. For example, in case of a whole genome sequencing data set, execution of alignment, SNP/Indel calling, SV calling, and functional annotation modules is required; CNV detection and the enrichment performance have to be skipped as these modules are suited for target enriched data only.

Logfile and status table

The masterscript writes all job submission calls and jobids for every analysed sample in a logfile. These stored jobids can be used to identify stdout/stderr output of the jobs that contains detailed error messages of executed tasks. Thus, this logfile facilitates the debugging of failed runs. The masterscript also writes the status (Running, Error, Finished) of every module to an sqlite table that is accessible via a webinterface (Supporting Information S4[69]). This table provides an easy monitoring of the workflow, thus, the administrators can see at a glance, which samples generated errors and react accordingly.

Configuration file

The workflow is configured for each sample with the help of an XML configuration file. This file contains sample specific information fetched from the LIMS, as well as all input and output paths for processing. Thus, the directory structure on the cluster for data input or output can easily be changed without changing the masterscript. In this case only the script that writes the XML files has to be updated.

3.5 Chapter summary

In this chapter,

- We provided an overview of our exome analysis workflow, which contains information of all required analysis steps.
- We described our HPC system and gave an overview of the implementation of our workflow and its components.
- We highlighted the significance of the essential components of workflow, like the masterscript, jobscripts, job submission and job monitoring function.
- We described the four main design principle of our workflow: speed, stability, robustness and maintainability, which makes the workflow more stable and robust as well as more efficient in terms of both speed and high-throughput.
- We used parallelization by jobarrays (including parallelization by chunks) and parallelization by threads to speed up our workflow.
- We performed resource optimization to exploit the power of the computing cluster.
- We provided certain measures that contribute to the stability of the cluster and our workflow, like controlled accesses to the parallel filesystem, no computation on the frontend node, no access to remote servers, etc.
- We mentioned some strategies that contribute to the workflow's robustness like dynamic job submission and clean exit.
- We presented our strategies to maintain the workflow in an efficient manner via modularity and the logfile.

Chapter 4
Detection of systematic sequencing errors

Continuous development of bioinformatics tools and automated pipelines has enabled the bioinformatics community to handle sequencing data in a fast and efficient manner. There are different alignment algorithms varying from sensitive to strict mapping of reads to the reference accompanied with post alignment improvements and different variant callers with reasonable accuracy. However, getting a variant list containing only true variants is still a challenge and false positive calls (FPs) remain a problem. Some of the FPs are due to the limitations of current tools. Moreover, errors occurred during sequencing (systematic or random errors) also produce a significant amount of FPs.

Some of the FPs generated by sequencing errors can be easily detected and filtered out. For example, errors at the ends of the reads can be avoided by trimming of these bases or by discarding bad quality reads (i.e. reads containing many low confident basecalls) (cf. Chapter 2). Modern alignment algorithms are also aware of this type of error and can avoid misalignments due to bad quality bases (by their clipping or trimming). GATK's base quality score recalibration (BQSR) tries to correct some errors that occur during the calculation of base quality scores based on statistics computed over sequencing lane, machine cycle and di-nucleotide context. Moreover, Indel realignment by GATK can also avoid the calling of false substitutions near Indels (introduced by wrong alignments of Indels on the reference sequence) (cf. Chapter 2). Besides the handling of some systematic errors, there are many alogrithms that perform correction of random base calling errors. These algorithms can correct the wrongly called base on some reads by utilizing the other reads that have the correct base. This correction is performed either by using shared k-mers among the reads or by performing mutiple sequence alignment of the reads mapped to a certain position. Yang and colleagues have reviewed these two approaches and provided a comparision of tools belonging to each category (Yang et al., 2013).

However, there is still much to explore in this class of error as many systematic errors are hard to detect or not yet detected. There are not many studies on error detection or correction methods for sequence specific and other types of systematic errors. Nakamura and colleagues reported first that dephasing can be induced by the presence of some specific sequence patterns (Nakamura et al., 2011). They reported that inverted repeats and GGC sequences are the two major sequence patterns that trigger sequence-specific errors (SSEs). After that, a few more studies have been conducted, in which the authors tried to discover some more SSEs or developed systematic error detection methods (Allhoff et al., 2013; Meacham et al., 2011; Ross et al., 2013; Zook, Samarov, McDaniel, Sen, & Salit, 2012).

This chapter aims to explore the different types of systematic errors. First, I will provide an overview of the published systematic error detection methods. Thereafter, I will present the drawbacks or limitations of these methods, which is the motivation behind this work: a newly developed approach to detect systematic errors in Illumina sequencing data. I will also present the comparisons of different data-sets generated with different target enrichment techniques, which is used for further classification and exploration of the detected systematic errors. Finally, I will present a new class of errors "RSE" and its characteristics. Moreover, I will provide details of the newly developed tool "FilterRSEs" that filters out RSEs from any variant list.

4.1 Systematic errors

4.1.1 Previous work

After the detection of certain biases and position specific errors by Dohm and colleagues in Illumina data (Dohm et al., 2008), Nakamura and colleagues found another class of errors (Nakamura et al., 2011). They observed that some mismatches occur in specific regions either near certain sequences or induced by some sequence patterns, and named these errors sequence-specific errors (SSEs). Moreover, they found that these mismatches are present mainly on one strand of reads (either forward or reverse) and accompanied by other mismatches downstream of the error. Based on their observations, they formulated the following two criteria that have to be met by an error to be called an SSE: a) At least 30% of reads in the same direction should carry these mismatchtes. b) Four other errors should be present within 40 bases downstream, but not within 40 bases upstream. After the analysis of selected SSEs, they found that the majority of SSEs are either present near GGC base triplets (or GCC for reverse strand SSEs) or inverted repeats. They also found that most of these mismatches are similar to the first or second preceding reference base and suggested that SSE associated mismatches originate due to dephasing. Moreover, they postulated that inverted repeats trigger folding of DNA single strand templates and so inhibit the base elongation process during sequencing in both directions (5' or 3') which causes an SSE. On the other hand, the GGC motif affects the preference of the DNA polymerase (like disabling of blocking effect which leads to dephasing) during the base elongation process and this is the most likely cause of the GGC-associated SSEs.

In another study, Meacham and colleagues (Meacham et al., 2011) also focused on systematic errors (in general) by using overlapping paired end reads from a methyl-Seq experiment. They selected this data because this experiment produces high average coverage and overlapping paired end reads provides two base calls for each location (irrelevant to strand directionality). Both characteristics of data helped them to avoid random errors and to distinguish between basecalling errors and true heterozygosity calls (Meacham et al., 2011). They defined systematic error as a "statistically unlikely accumulation of errors at specific genome (or transcriptome) locations". They observed that systematic errors are present in approximately 1 in 1000 base pairs and are reproducible across different experiments. They confirmed the pattern observed by (Nakamura et al., 2011), that systematic base calling errors are present only in one sequencing direction (either forward or reverse). They observed that the bases preceding the error contain information about the presence of the error and the base quality scores of error locations are lower than those at their neighbouring sites. They also found that most of the errors occurred at GGT motifs where base T is having the substitution error. Moreover, they provided a classifier called "SysCall[71]" that uses logistic regression based classification to distinguish heterozygous sites from systematic errors. Here, the training of the logistic regression model is based on the characteristics of systematic errors observed during their study.

In the first study of SSE, Nakamura and colleagues provided some insights into this type of error, but did not provide any framework for detecting SSEs (Nakamura et al., 2011). Moreover, the method used by Meacham and colleagues is not distinguishing between SSEs and other types of systematic error (Meacham et al., 2011). Thus, Allhoff and colleagues developed a statistical method to detect only SSEs and referred to SSE as CSE (Context-Specific Error) (and error-inducing sequence motifs as contexts) (Allhoff et al., 2013). As it is already mentioned in the previous studies that the SSEs are present only on one strand (known as strand bias positions), they used this fact to distinguish between true SNPs and CSE. They screened the genomic positions with strand bias and associated base calling errors with sequence contexts at these positions. By using these positions, they developed a statistical method ("discovering-cse"[72]) that can identify context-specific sequencing errors (CSEs) including the associated sequence context. Moreover, they provided a list of error-prone genomic positions in BED format[73], which can be used to filter false positives due to CSEs.

4.1.2 Motivation

The idea of this work originated when we saw lots of variants shared across a significant number of different human DNA sequencing data sets (around 500 exome sequencing samples). These are not common variants present in the population, which implies that these shared variants are false positives caused by some sort of systematic errors. We refer to these errors as "Recurrent Systematic Errors (RSEs)". These RSEs are called as variants by samtools and GATK's unified genotyper and some of them also fulfil the

[71] http://bio.math.berkeley.edu/SysCall/ (This and subsequent URLs in this chapter are accessed on 16 July 2015)

[72] https://bitbucket.org/tobiasmarschall/discovering-cse

[73] https://genome.ucsc.edu/FAQ/FAQformat.html#format1

standard SNPs/Indels good quality call thresholds suggested by GATK's best practice guidelines (cf. Chapter 2). For example, the SNP (which is systematic error) shown in Figure 4.1 (highlighted with vertical bar in black colour) was detected in the control sample (NA12878) by two different pipelines (BWA-aln+GATK Unified genotyper and BWA-MEM+GATK Haplotype caller) (cf. Chapter 2). Moreover, it is supported by reads aligned in both forward and reverse direction with good mapping quality and not filtered out by all standard filtering criterias for FPs. Therefore, the detection of this type of error is very essential to reduce the number of false positives.

As mentioned above, previous works explained some systematic errors on sample level (errors in an individual sample) and associated causes, but not RSEs. They also mentioned that there are many more systematic errors than explained in their studies. Moreover, these studies mainly focused on the base calling errors (i.e. a mismatch or SNP), but do not provide any information about the effect of systematic errors on Indels. Besides the errors originating from the sequencing process, certain genomic regions, like LCR regions, can also cause systematic errors. It is known that both sequencing and mapping of LCR regions are difficult or not accurate, which can lead to lots of mismatches (Treangen & Salzberg, 2012). These facts motivated us to explore RSEs in human sequencing data to find both platform-specific as well as genome-specific systematic errors.

Besides the exploration of these errors, we also felt the need to develop a new systematic error detection approach due to some drawbacks of the previous methods. Nakamura and colleagues presented an approach to detect sequence specific errors (SSEs), but, their SSE position detection criteria are very specific (Nakamura et al., 2011). It captures certain SSEs only due to the following limitations of their selection criteria (cf. Section 4.1.1). a.) The first selection criteria can only capture SSEs above 30x coverage (for one strand), but a mismatch caused by SSE below this threshold can easily be called as a SNP (depending on datasets, sometimes 10x coverage is enough to call a confident SNP). b.) The second criteria only yields a region of consecutive mismatches which can be due to bad quality bases. These mismachtes can be filtered out by quality trimming in good quality data (with good read coverage and basecalling quality) or by variant callers (with or without FPs filtering strategies) (cf. Chapter 2). Moreover, they did not provide any algorithm/tool for the detection of SSEs.

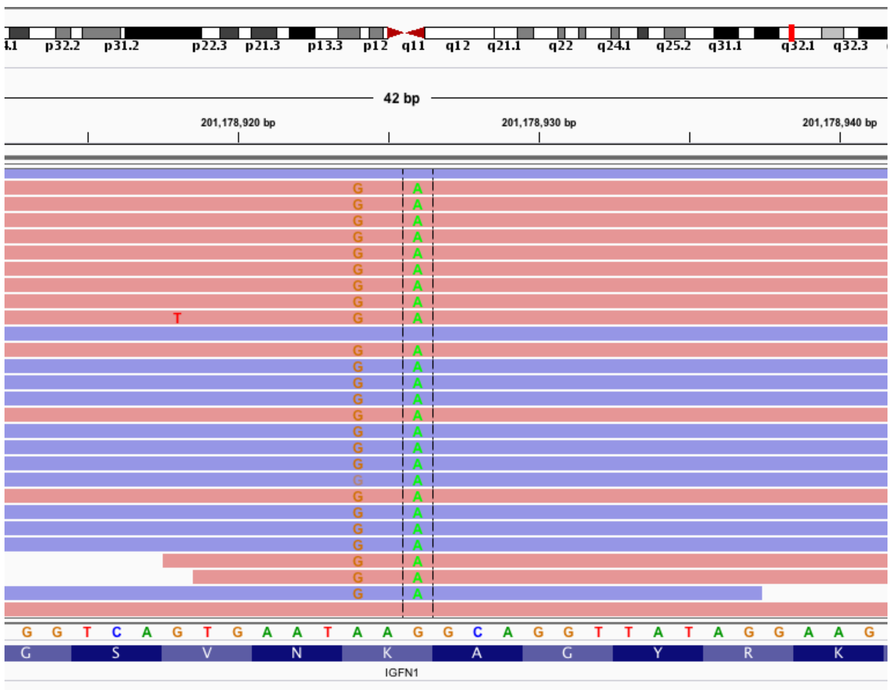


Figure 4.1 Alignment visualization of two systematic errors in a control sample (NA12878). This figure is generated by IGV[74] (Robinson et al., 2011) in which horizontal bars are representing aligned reads at chr1 (location: 201178924 and 201178926) on both forward (red colour) and reverse (blue colour) reference strand. Nucleotide bases G and A shown in the figure depict mismatching bases (or systematic errors), which are supported by both forward and reverse strand reads.

Although the other two studies (Allhoff et al., 2013; Meacham et al., 2011) provided tools for systematic error detetion, they also focused only on one certain type of errors. Both methods consider errors that are present only on one strand but systematic errors can be present on both strands (cf. Figure 4.1). Thus, slection based on a strand bias criterion can miss many errors and most of these strand bias errors can be easily filtered by current variant calling tools (Samtools, Platypus, GATK) (cf. Chapter2). Moreover, SysCall uses a specific training dataset containing certain error charatetristics, and is not having good accuracy for other types of errors e.g. run-specific errors. The other tool

[74] https://www.broadinstitute.org/igv/

"discovering-cse" also deals with only one class of errors i.e. CSE/SSE on strand bias locations. The top ten reported sequence motifs from this study are containing "GGC" motif, which is already reported cause for few SSEs. However, there are many other detected SSEs by both of the studies (Allhoff et al., 2013; Nakamura et al., 2011), which are not yet explained and many other undetected SSEs which need to be detected.

4.1.3 Method

As mentioned above, I want to explore RSEs which I define them as mismatches (both SNP and Indels) present at the same genomic location throughout many data sets (sequenced from the same sequencing technology). This definition is based on the assumption that if there is a mismatch to the reference genome sequence at the same location in multiple datasets then either this is a common variant present in the population or it is occurring due to the same systematic error during the DNA sequencing of these datasets. I assume that the RSEs can be explained by the presence of some motifs upstream or downstream of the mismatch position. This assumption is based on the following facts derived from previous studies (cf. Sections 4.1.1, 4.1.2):

- Sequence motifs in the DNA template influence the sequencing process, which leads to base calling errors.
- Sequence motifs in the reference genome cause faulty read mapping, which leads to mismatches in the alignment.

Therefore, I decided to screen all sequence motifs of a certain length throughout the human exome sequencing data (sequenced on Illumina Hi-Seq sequencer), which is aligned to the human reference genome sequence (GRCh37) (cf. Chapter 2). The screening of sequence motifs throughout the complete target region of the exome enables the comparative analysis of thier occurrence near RSEs as well as in the other part of the genome. This way we can distingush between a true association and a random co-ocurrence of sequence motifs near RSEs.

I refer to these sequence motifs as "kmers" and categorize them in the following two classes:

- Observed kmer: A string of consecutive bases of length k in the read sequence of a sample.
- Expected kmer: A string of consecutive bases of length k constructed from the reference genome sequence.

Similarly, their counts can be categorized in the following two classes:

- Observed kmer count: number of occurrences throughout all the aligned sequencing reads on the reference genome sequence.
- Expected kmer count: number of times this reference kmer is sequenced. The expected kmer count is computed from the reference sequence based on the coverage of each reference base (i.e. number of aligned reads at that reference base position) (cf. Section "Expected kmer count computation").

The idea behind these classes is that I should see equal observed and expected counts for all kmers from the sequenced regions which mapped perfectly to the reference sequence (without any mismatches). In other words, the ratio of obeserved against expected count (I call it OEratio) should be equal to 1 in the perfectly mapped regions. Kmers with an OEratio < 1 have an observed kmer count which is less than their expected kmer count, which means that they are frequently deviating from the reference sequence in the sequenced sample. The reason for this can either be a true variant or a sequencing error within the kmer. On the contrary, an OEratio > 1 means that the observed count is higher than the expected count, which can be caused by a mixture of effects that cannot be differentiated easily. For example, both systematic and random sequencing errors in multiple expected kmers can produce the same observed kmer in the reads increasing the observed count and so the OEratio. Therefore, kmers with low OEratios are better suited as markers for systematic sequencing errors than those with high OEratios.

In the following sections, I will describe how expected and observed kmer counts are computed and the application of the OEratio to detect systematic errors. The work is divided into 4 main sections (cf. Figure 4.2). At first I aligned the read sequences of a sample to the reference genome sequence. The aligned data from each sample is then used to construct the kmers and to compute both expected and observed kmer counts. After that, I use these counts to compute the OEratio. At last, I use the generated list of kmers and their OEratios to detect systematic errors. I narrow down the detected systematic erros by a series of filters including RSEs and validate them. Each step is described below in detail.

Read alignment & post processing

As the kmer construction is based on the alignment of the reads to the reference sequence, I first need to perform read mapping. For this purpose I use our alignment pipeline (cf. Chapter 3), which uses backtrack and MEM algorithm of BWA for the alignment and performs post processing of aligned reads for further improvements (cf. Chapter 2). I extracted only aligned reads on the target regions (as defined by the exome enrichment kits) including 100 bp flanking regions both upstream and downstream of the targets. This avoids a lot of random false positives due to little coverage or wrong read mapping in the off-target regions. I also removed duplicate reads as they can provide false evidence about a variant (cf. Chapter 2). Moreover, I removed unmapped reads to avoid an unnecessary overhead (in terms of file size and computation time). After all of the above-mentioned steps, I got aligned reads in a BAM format, which is a compressed format of SAM (Sequence Alignment Format) (cf. Chapter 2). I further split these BAM files into 48 BAM files, two for each chromosome (excluding the mitochondrion) generating separate BAM files for reads mapped on the forward and the reverse reference strand, respectively. The kmer construction process is computationally very expensive and was performed on the created chunks of BAM files, which sped-up the entire process (kmer generation and their counts) more than 4 fold.

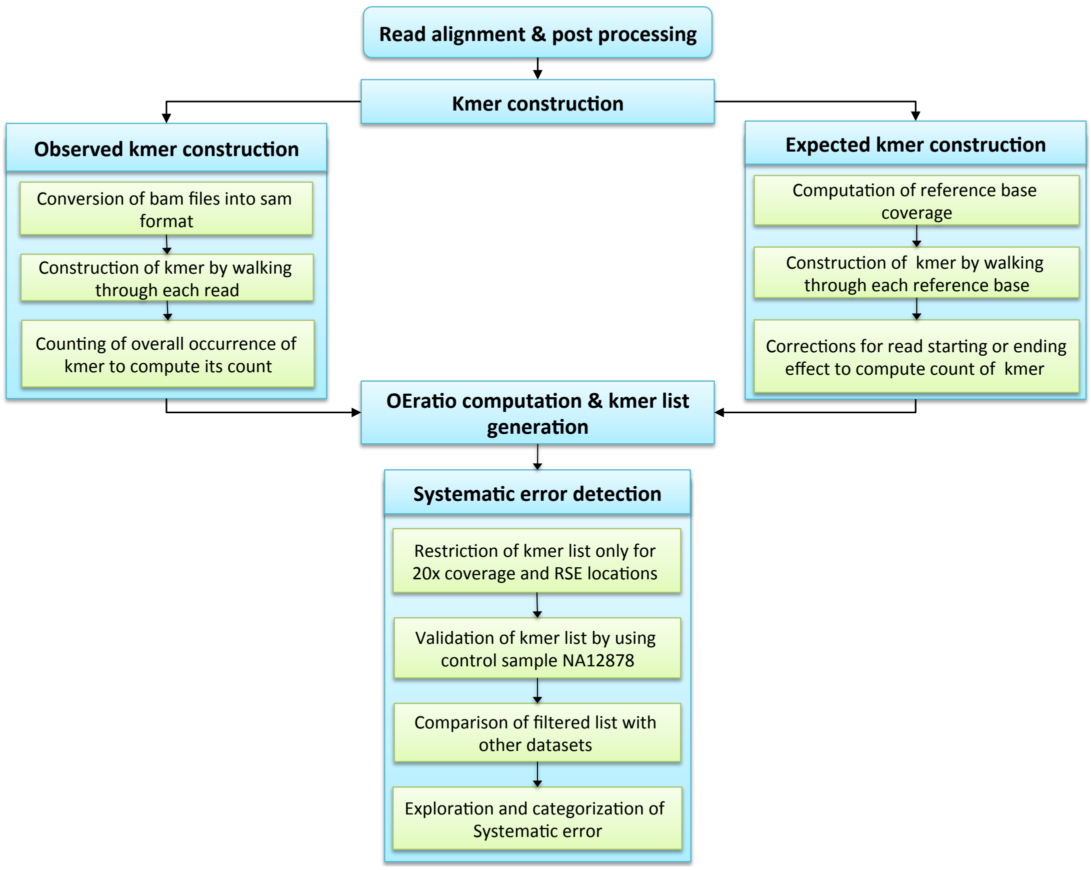

Read alignment & post processing
Kmer construction
Observed kmer construction
Conversion of bam files into sam format
Construction of kmer by walking through each read
Counting of overall occurrence of kmer to compute its count
Expected kmer construction
Computation of reference base coverage
Construction of kmer by walking through each reference base
Corrections for read starting or ending effect to compute count of kmer
OEratio computation & kmer list generation
Systematic error detection
Restriction of kmer list only for 20x coverage and RSE locations
Validation of kmer list by using control sample NA12878
Comparison of filtered list with other datasets
Exploration and categorization of Systematic error

Figure 4.2 The four main components of the systematic error detection workflow.

Observed kmer construction

In order to parse the aligned read sequences, I converted BAM format into SAM format and also removed the reads containing ‘N’. As these ‘N’ bases are randomly mapped by the aligner on any of the four references bases (A, T, G, C), the kmers containing ‘N’ would not be very informative. To construct all kmers that are present in the reads, I screen all reads from their starting till end position. I perform a separate screening for reads mapped on forward and reverse strands. The observed (OBS) kmer is the string of consecutive bases from ith till jth [j= i + (k -1)] position in the read.

The construction of the first kmer starts from the first base (i=1) and ends with the base at 9th position (j=9), if the kmer length (k) is equal to 9. Similarly, the second kmer generation starts from the second base of the read (i=2) and ends at 10th base (j=10). This process continues till the end of the read, generating all kmers that are contained in the read (cf. Figure 4.3).

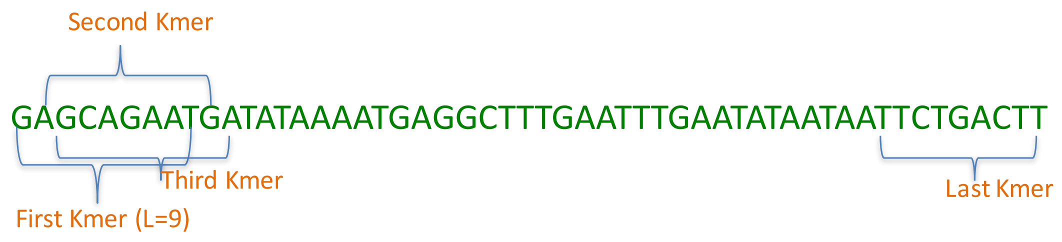


Figure 4.3 Observed kmer construction from a sequence read.

I perform this kmer generation on all reads in a sample and count the occurrence of each kmer throughout all reads. For reads from the reverse strand, I have to take the reverse complement of kmers. This is because reverse strand reads are stored in SAM format as the reverse complement of the actually sequenced read. At last, I generate a list of these observed kmers and their respective counts in the underlying sample. All of the above-mentioned process is implemented in a Perl[75] script and its pseudocode can be found in Appendix.

Expected kmer construction

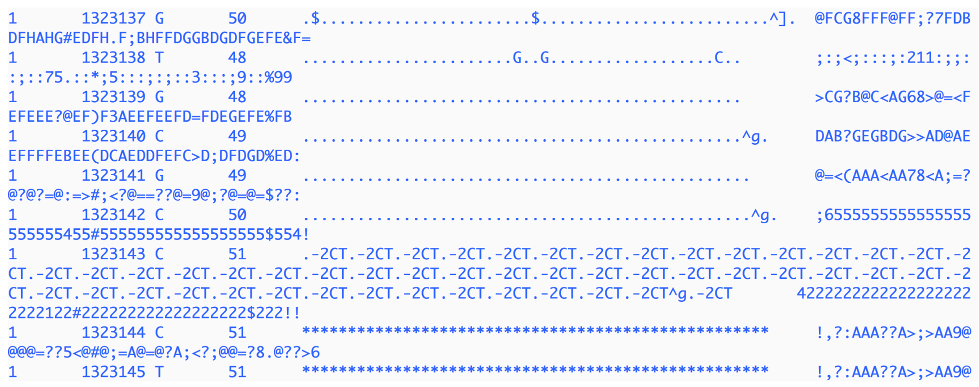
To construct the expected kmers, I first computed the coverage of each reference base (i.e. number of aligned reads at that reference base position) within the target regions including 100 bp flanking regions both upstream and downstream. I used the Samtools mpileup function with the –D parameter, which generates a pileup format[76] of aligned reads containing the coverage of each reference base. I generated the pileup of all bases including bases with zero base quality score (Q) (default Q=13) and extracted only those bases having coverage greater than 10. Figure 4.4 shows the pileup format of aligned reads. It is a tab-delimited format where the first column is the chromosome name, the second column is the location, and the third column is the reference base. The number of reads covering the reference base position (i.e. base coverage) is shown in the 4th column followed by read bases (5th column) and base qualities (6th column). The read base column contains alignment details, where dot (.) (cf. Figure 4.4 (A) or comma (,) (cf. Figure 4.4 (B)) represents a match to the forward and reverse strand respectively. A

[75] https://en.wikipedia.org/wiki/Perl

[76] http://samtools.sourceforge.net/pileup.shtml

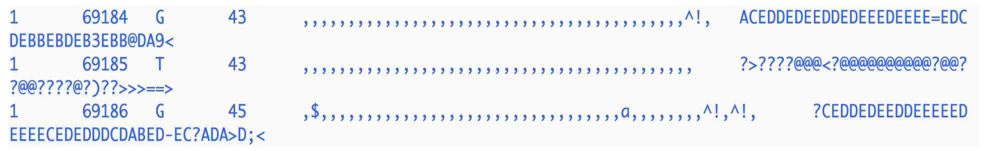
mismatch is shown as one of the bases from ‘ACGTN’ or ‘acgtn’ on forward and reverse strand, respectively. Insertions between the current reference position and the next reference position are shown by `\+[0-9]+[ACGTNacgtn]+' pattern. Similarly, the pattern `-[0-9]+[ACGTNacgtn]+' shows a deletion of base pairs in the read compared to the reference sequence. For example, in the Figure 4.4 (A) we can see a deletion of CT bases after the current base denoted by -2CT and stars (*) in the next two positions show these bases are deleted. The start of the read is denoted by ‘^’, followed by an ASCII character which encodes the mapping quality as ASCII value -33 (e.g. ^g. in the 4th line in (cf. Figure 4.4 (A)). The end of the read is denoted by ‘$’ (e.g. ‘.$’). The understanding of the pileup format is necessary; as the information stored in the above mentioned symbols is used in the expected kmer counts calculation.

A.)



```
1       1323137 G       50      .$......................$.........................^].  @FCG8FFF@FF;?7FDB
DFHAHG#EDFH.F;BHFFDGGBDGDFGEFE&F=
1       1323138 T       48      .......................G..G..................C..        ;:;<;:::;:211:;;:
:;::75.::*;5:::;:;::3:::;9::%99
1       1323139 G       48      ................................................        >CG?B@C<AG68>@=<F
EFEEE?@EF)F3AEEFEEFD=FDEGEFE%FB
1       1323140 C       49      ...............................................^g.     DAB?GEGBDG>>AD@AE
EFFFFEBEE(DCAEDDFEFC>D;DFDGD%ED:
1       1323141 G       49      .................................................       @=<(AAA<AA78<A;=?
@?@?=@:=>#;<?@==??@=9@;?@=@=$??:
1       1323142 C       50      ................................................^g.    ;6555555555555555
555555455#555555555555555555$554!
1       1323143 C       51      .-2CT.-2CT.-2CT.-2CT.-2CT.-2CT.-2CT.-2CT.-2CT.-2CT.-2CT.-2CT.-2CT.-2CT.-2
CT.-2CT.-2CT.-2CT.-2CT.-2CT.-2CT.-2CT.-2CT.-2CT.-2CT.-2CT.-2CT.-2CT.-2CT.-2CT.-2CT.-2CT.-2CT.-2CT.-2
CT.-2CT.-2CT.-2CT.-2CT.-2CT.-2CT.-2CT.-2CT.-2CT.-2CT.-2CT.-2CT.-2CT^g.-2CT      4222222222222222222
2222122#222222222222222222$222!!
1       1323144 C       51      ***************************************************     !,?:AAA??A>;>AA9@
@@@=??5<@#@;=A@=@?A;<?;@@=?8.@??>6
1       1323145 T       51      ***************************************************     !,?:AAA??A>;>AA9@
```

B.)



```
1       69184   G       43      ,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,^!,    ACEDDEDEEDDEDEEEDEEEE=EDC
DEBBEBDEB3EBB@DA9<
1       69185   T       43      ,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,     ?>????@@@<?@@@@@@@@@?@@?
?@@????@?)??>>>==>
1       69186   G       45      ,$,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,a,,,,,,,^!,^!,       ?CEDDEDEEDDEEEEED
EEEECEDEDDDCDABED-EC?ADA>D;<
```

Figure 4.4 Pileup format of aligned reads. Part (A) and part (B) show the pileup of aligned reads on the forward strand and on the reverse strand, respectively.

Expected kmers are constructed in a similar manner as described for observed kmers. These kmers are also strings of consecutive bases from ith till jth positions in the reference genome sequence. Instead of walking through the read sequence (start to end), here, the program walks from top to bottom of the pileup of aligned reads and construct the kmer from the consecutive reference bases (3rd column). For example, if the starting position is “1323137” (first line of the Figure 4.4 (A)), then the first kmer of length 7 is generated by using reference bases till position “1323143” and the kmer is “GTGCGCC”. Similarly, the second kmer is from position “1323138” till “1323144” (kmer: “TGCGCCC”), and so on.

With my approach I want to capture systematic sequencing errors that occur in the context of certain sequence motifs (cf. Section 4.1.3). I assume that such motifs can be present either upstream or downstream of the error position. Therefore, I construct the expected kmers according to two different models:

- Prefix model (PM): Prefix model captures the motif sequences upstream of a mismatch position. For example, if k=9 (9mer=TATCTCGCG), then the first 8 bases (in green colour) are the motif sequence and the last reference base (9th base) (in red colour) is the base of interest, which carries an error (mismatch in read sequence compared to reference).
- Suffix model (SM): Suffix model captures the motif sequences downstream of a mismatch position. For example, if k=9 (9mer=TATCTCGCG), then the first reference base (in red colour) carries the mismatch (or error) and the remaining 8 bases (in green colour) are the motif sequence.

The computation of the expected kmer counts is different and much more complicated than the computation for the observed kmers. The count (Count) of the expected (EXP) kmer (for both PM and SM) is the number of times this kmer was sequenced in the underlying sample, which is the number of reads covering this kmer completely in the alignment. It can be computed as, the average of coverage values (Cov) of each base (B) present in the kmer of length (k) minus over-counts (OC) of reads starting and ending within the kmer (cf. Figure 4.4). Specifically, the over-counts that have to be subtracted are:

1. Over-counts due to reads either starting (^. or ^,) or ending (.$ or ,$) at any middle position of the kmer (e.g. for k=5, at positions 2 to 4 (k-1)) (OCmid).
2. Over-counts due to reads ending at the first position of the kmer (at position 1) (OCstart).

3. Over-counts due to reads starting at the last position of the kmer (at position k) (OCend).

Thus, the exact count of a kmer is calculated by the following formula:

$$Count(EXP\ kmer) = \left(\sum\nolimits_{i}^{j} Cov(B) - (\mathrm{OCmid} + \mathrm{OCstart} + \mathrm{OCend})\right)/k;$$

where i and j are the start and end positions of the expected kmer in the reference sequence.

While OCstart and OCend are simply the number of reads ending at the first position and starting at the last position of the kmer, respectively, the computation of OCmid is more complicated. If a read starts at any of the middle positions, then it contributes to the coverage values of all following bases. Similarly, when a read is ending at middle position, then its count is already included in the coverage value of the previous bases. Thus, I need not only to subtract the number of reads starting or ending at a middle position, but also their contributions to the coverage values of the following or previous bases, respectively. Therefore, the OCmid correction term is calculated as:

$$OCmid = \left(\sum_{l=2}^{k-1} OCs(l) * (k - l + 1) + \ OCe(l) * l\right);$$

where OCs(l) and OCe(l) are the number of reads starting and ending at middle base l of the kmer. During the kmer construction, I also capture and store its genomic location on the reference sequence, which I am using later in systematic error detection (cf. Section “Systematic error detection”). The kmer location is the location of the last base in the kmer string (base of interest) for PM, whereas, it is the position of the first base (base of interest) for SM. The expected kmer construction and their counts computation are implemented in Perl scripts. Their pseudocode can be found in Appendix.

OEratio computation

After the generation of the kmers, I first merged the kmers and their counts (Count) from both forward and reverse strand. If the kmer is present in both strands, then I added both counts and report this kmer with this total count in the merged kmer list. The kmers that occur only on one strand are reported as they are (with their count) in the merged kmer list. I applied this merging for both observed (OBS) and expected (EXP) kmer lists. Thereafter, I computed the OEratio for each kmer based on the following formula:

$$OEratio = Count(OBS\ kmer)/Count(EXP\ kmer)$$

If a kmer is present in both the OBS and EXP kmer list, then simply the above mentioned formula is used to calculate OEratio. Whereas, if a kmer is exclusively present in only one of the lists , then I simply use zero as the count of the absent kmer to compute the OEratio. For example, if a kmer is not in the OBS kmer list, then its OEratio will be zero. In the other case (absence of the kmer in EXP kmer list), the OEratio is infinity.

Systematic error detection

I used the above mentioned kmer approach to explore systematic errors in human exome data. For this purpose, I selected a set of 10 samples from an epilepsy study randomly from our in-house data set. Additionally, I included a control sample NA12878 for validation purpose (cf. Section "Validation of kmer list"). The paired-end exome sequencing of these samples were performed on the CCG's Illumina HiSeq sequencer and NimbleGen SeqCap EZ Human Exome Library v2.0 (**V2_IN**) kit was used for target enrichment. All of these samples were analysed by our exome analysis workflow (cf. Chapter 3) where read mapping is performed by BWA-aln algorithm (with n=7) followed by post alignment improvements and variant calling by both mpileup and GATK's unified genotyper (UG) (cf. Chapter 2). This data set is having a mixture of both male and female samples with more than 90% of the target region covered at 20X or higher.

I constructed both observed and expected kmers and computed the OEratios for all 11 samples (as described above). I selected the kmer length as 9 (k=9) for systematic error detection. The reasons behind this selection are:

a) Previous studies only focused on sequence motifs of length 4 and were not able to detect/explain all systematic errors (cf. Section 4.1.2).
b) (Allhoff et al., 2013) already reported that longer motifs are more specific and able to detect more SSEs than the shorter motifs. They used 8 as motif length for their study.

c) After comparing kmers of lengths 4 to 10, I also found that a higher motif length is more specific as it avoids the additive effect of normal kmers. Here, a normal kmer means a kmer having no mismatch at the position of interest, thus, not representing any error. At low kmer length, the number of possible kmers formed by the 4 nucleotide bases is very small. For example, at k=4 we get $4^4 = 256$ different kmers, on the other hand, at k=9 the number of possible kmers is $4^9 = 262144$. Thus, at the lower kmer lengths we can have a large number of normal kmers that dominate the OEratio computation, which makes it difficult to detect error-containing kmers. I did not go beyond k=9, as it increases the computational cost. Moreover, the combinations of kmers generated from both prefix and suffix model are containing 17 bases (having base of interest in the middle), 8 bases upstream and downstream of the error position. Thus, I assume that 9mers (or 17mers in the combination) are enough to capture the systematic errors.

The generated kmer lists contain 6 columns: the expected kmer (which is the motif of interest), expected and observed kmer counts, OEratio, kmer locations on the reference genome and read strand information (whether the kmer is constructed from reads mapped to the forward or reverse strand or both strands). To exclude badly covered kmer genome-wide, due to coverage bias or misalignments, I filtered out kmers having less than 20 expected counts. After this initial preparation, I performed the following steps to explore RSEs.

Extraction of kmers having RSEs

In order to focus only on RSEs, first I extracted only those kmers that are having RSEs variants (either SNP or Indel) at the last or the first base of the kmer from PM and SM, respectively. I considered all variants called in a sample as RSEs if they are also present in more than 300 samples among 511 epilepsy samples of our in-house data set. However, I make sure that these labelled variants are not common variants shared among different human populations (are only shared in our in-house data set). As in-house data set (called InhouseDB[77]), we have variant calls of 511 epilepsy patient samples also generated by our exome analysis workflow. We integrated variant calls (called DBall[77]) provided by the 1000 genomes project (phase 3[78]) and the ExAC[79] consortium. Moreover, we annotated these integrated calls with rsids[80] from DBSNP[81] database. The 1000 genome consortium (Abecasis et al., 2012) provided lists of variants called by genome sequencing of 2504 individuals from 26 populations. The ExAC consortium preformed exome sequencing of 60,706 unrelated individuals and provided an integrated list of called variants. Both variant lists contain population specific information, like total number of alternate alleles in called genotypes (AC), total number of alleles in called genotypes (AN) or allelic frequency (AF) in different populations. I computed minor allele frequencies (MAF[82]) (i.e. AC divided by AN) of variants present in DBall. If a variant is present in both variant lists, then I computed MAF by using AC and AN values from ExAC list, as it is containing variants from more than 60,000 individuals. I considered a variant as a common variant if it has MAF > 0.30 (30%).

To extract kmers having RSEs only, first I took the overlap between variants in InhouseDB and DBall, and filtered out the common variants among these overlapping variants. Then, I selected only those variants (i.e. RSE locations) that are shared among at least 300 samples of InhouseDB. To narrow down this list, I further extracted only those RSE locations that are shared by at least 7 samples among the 10 selected samples. I compared these RSEs locations with the kmer locations and kept only those kmers whose last or first base are at the location of an RSE (for PM and SM, respectively). I performed this filtering on all selected 11 samples and this way prepared lists of kmers associated with RSE locations for each sample.

[77] Both InhouseDB and DBall construction is performed by Dr. Holger Thiele

[78] http://www.1000genomes.org

[79] http://exac.broadinstitute.org

[80] http://www.ncbi.nlm.nih.gov/SNP/get_html.cgi?whichHtml=how_to_submit#REFSNP

[81] http://www.ncbi.nlm.nih.gov/SNP/

[82] http://hapmap.ncbi.nlm.nih.gov/hapmart.html.en

Alignment or random artefacts filtering

To filter some alignment artefacts, I generated two different lists of kmers associated with RSE locations for each sample by using aligned reads by BWA-aln and BWA-MEM algorithms. As I am using BWA-aln with increased sensitivity (n=7), it can produce many FPs. On the other hand, BWA-MEM has well balanced sensitivity and specificity (cf. Chapter 2). Therefore, I am taking only those kmers that are present in both kmer lists, those constructed from the BWA-aln alignment and those from the BWA-MEM alignment, to filter out possible alignment artefacts produced by the BWA-aln algorithm. Thereafter, I merged the filtered kmer lists from all 11 samples and retained only those kmers common to at least 7 of the 11 samples. This excludes random errors occurring in an individual sample or sample specific systematic errors and focuses the resulting kmer list to mainly RSEs.

Validation of kmer list

As explained in Chapter 2, sample NA12878 can be used as a control sample to validate lists of variants. Thus, I compared the variant locations (which is RSE location) in the kmer list that are shared by the NA12878 sample with NIST's variant list to validate that these variants are only FPs. At first, I compared variant calls (called by UG) of this sample with the NIST list as recommended to get lists of true variants (or highly confident variants) (cf. Chapter 2). Then, I checked how many variants from this list are present in the kmer list and marked them with a keyword. I also used validated calls[83] for this sample provided by the Broad institute[84], as an additional annotation. Moreover, I compared variant calls in the list with variant calls for the same sample but called by a different data-analysis pipeline, where I performed read mapping by BWA-MEM (followed by GATK's post alignment improvement) and variant calling by GATK haplotype caller. I also marked those locations, which are not present in the variant list generated by the new pipeline (as these can be artefacts produced by variant calling). Furthermore, I annotated the RSE locations with their presence in DBall and reported

[83] http://ftp.ncbi.nlm.nih.gov/1000genomes/ftp/technical/working/20130806_broad_na12878_truth_set

[84] http://gatkforums.broadinstitute.org/discussion/1292/which-datasets-should-i-use-for-reviewing-or-benchmarking-purposes)

MAF for matched entries. All of the above mentioned comparisons are enough to validate whether the locations in our list are FPs in all data sets or can appear as true variants in any individual like NA12878 or another from some population analyzed in the 1000 genome project. At last, I compared our kmer list, with kmers generated by the tool "discovering-cse"[85] (they have also provided a list of FP location) (cf. Section 4.1.1).

Comparison with other datasets

The detected RSEs can simply be some artefacts introduced during library preparation with a certain enrichment kit or can be technology (Illumina) specific artefacts or sequencing centre specific artefacts (batch effect). Thus, in order to figure out the nature of these errors, I compared this kmer list with the following different paired-end exome sequencing (sequenced on Illumina Hiseq) datasets, in which at least 90% of the target region is covered at 20X or greater:

- **V3_IN**: Set of 11 samples sequenced in-house by using the NimbleGen SeqCap EZ Human Exome Library v3.0 target enrichment kit.
- **SS5_EX:** Set of 11 samples sequenced in another sequencing centre by using the Agilent SureSelect Human All Exon v5 target enrichment kit.
- **V2_EX**: Set of 11 samples sequenced in another sequencing centre by using the NimbleGen SeqCap EZ Human Exome Library v2.0 target enrichment kit.

To make these datasets comparable, all of these samples are analysed by the same data analysis pipeline. Kmers (k=9) construction and OEratio generation are also performed in the same way as described above. Extraction of kmers at RSE locations is also performed in the same way, except the final list of FPs used here is only variants that are not common and shared by at least 300 samples in InhouseDB (not requiring the additional sharing of the variants between 7 of 10 samples) (cf. Section "Extraction of kmers having RSEs"). Thereafter, the kmer lists from all 11 samples of each data set are filtered from alignment artefacts and merged into one list to avoid random errors. At last, these 3

[85] https://bitbucket.org/tobiasmarschall/discovering-cse

different kmer lists are compared with our final kmer list from **V2_IN** and kmers are marked according to their presence or absence in all of these datasets.

Filtering to get strong RSEs

To focus on strong RSEs (i.e. hard to detect or filter out by standard filters), I applied best practice filters[86] (cf. Chapter 2) to filter out some easily detected FPs and focused only on the remaining variant locations. In order to narrow down the screening further, I used the OEratio. I computed the average of OEratio for all kmers throughout the 11 samples of the **V2_IN** data set and focused on only those kmers present either on forward or reverse or both strands that have an average OEratio less than 1. As explained above (cf. Section 4.1.3), a kmer with a low OEratio is more likely to be associated to a systematic sequencing error than one with a high OEratio, which is why I focus our investigation on this class of kmers.

4.2 Results

Table 4.1 shows the statistics of the generated 9mers from our dataset of interest (**V2_IN**) where I selected 10 epilepsy samples and 1 control sample (column 1), which were sequenced in-house (exome sequencing) on an Illumina HiSeq and enriched by NimbleGen v2.0. Approximately 6 million (column 2) 9mers (a few might be identical to each other, but appears at different chromosomal location) from different chromosomes (1-22, X, Y) are constructed by both Prefix model (PM) and Suffix model (SM) from the target regions (±100bp flanking regions) of the sequenced samples. At the raw level (all generated 9mers without any filtering), these numbers are the same for both PM and SM model, as they represent the same kmer but with different RSE location. The filtering based on coverage (column 3), where I filtered out 9mers having less than 20x read coverage, filtered out approximately 6% of data. The extraction of 9mers at RSE locations according to InhouseDB shortened the 9mers list drastically (column 4). Furthermore, the alignment artefact filter (column 5) also filtered out a few alignment errors (approx. 5% of RSE containing 9mers).

[86] http://gatkforums.broadinstitute.org/discussion/2806/howto-apply-hard-filters-to-a-call-set

Sample	**Number of 9mers**			
(Dataset: V2_IN)	**Raw**	**After Filter > 20x**	**Only on RSE locations**	**After Aln artefact filtering**
NA12878	6107687	5790795	3740	3492
S1	6112969	5799549	4129	3969
S2	6066480	5729634	3383	3199
S3	6159831	5823955	3589	3408
S4	6086135	5782131	3797	3606
S5	6058726	5786777	3967	3799
S6	6172392	5819470	3841	3636
S7	6199838	5832016	3709	3562
S8	6058524	5724690	3665	3503
S9	6190481	5825560	3798	3643
S10	6071964	5691843	3681	3513

Table 4.1 Statistics of generated 9mers from **V2_IN.**

Category	**Number of 9mers (Overlapping 9mers of V2_IN with other datasets)**							
	Raw Overlap		**After Avg. OE < 1 on raw list**		**After BestPrac. Filt. on raw list**		**After OEFilt on BestPracFilt. list**	
	PM	**SM**	**PM**	**SM**	**PM**	**SM**	**PM**	**SM**
With all datasets	1649	1697	242	239	1461	1513	211	211
With V2_EX & V3_IN)	1069	1059	187	178	820	813	127	118
With V3_IN & SS5_EX	75	97	5	6	71	86	5	6
With V2_EX & SS5_EX)	67	81	5	2	65	79	5	2
With V3_IN	112	91	14	11	96	78	9	8
With V2_EX	45	38	3	3	41	35	3	3
With SS5_EX	17	9	1	2	16	9	1	2
No Overlap	28	31	3	3	26	28	3	3
Total	3062	3103	460	444	2596	2641	364	353

Table 4.2 Overlap between the test dataset and the 3 other datasets.

As mentioned in the method section, I extracted only 9mers that are shared by at least 7 of the 11 samples considered to filter out some random errors (or sample specific errors). This step further reduced the 9mer list by approx. 15% (cf. Table 4.1). Thereafter, I compared this list with those produced from the 3 other datasets (V2_EX, SS5_EX, V3_IN) to assess the reproducibility and nature of these errors (cf. Table 4.2). The comparison of the V2_IN list with the lists from the other datasets shows that approximately 94% of RSEs are reproducible (in at least three datasets, sum of the first four categories). Only 1% of the RSE associated 9mers (category: No Overlap) are not present in any other dataset. The remaining 5% RSE associated 9mers in V2_IN are only shared by one other dataset (e.g. V2_IN with V3_IN), which is still a sign of reproducibility. This high level of reproducibility across different data sets supports our hypothesis that the majority of RSEs in the V2_IN list are indeed systematic errors. 54% of these systematic errors are shared by all datasets irrelevant to enrichment kit and sequencing centre. Thus, these errors are most likely induced by the Illumina sequencing process or due to standard library preparation protocols (used in a similar way in most of the sequencing centres). I refer to this error type as class 1: Platform dependent errors. Besides this class, I also detected another class of error (approx. 35% of the total), which occurred due to the combination of Illumina sequencing with NimbleGen SeqCap EZ Human Exome Library kit (both V2 and V3) only. I refer to this error type as class 2: Target enrichment dependent errors. I also compared my 9mer list with the 9mer list generated by the tool "discovering-cse"[87] on the control sample. I found only 1% overlap between these two lists, which is expected as "discovering-cse" only focuses on strand biased positions and detects systematic errors at the sample level. In contrast, my 9mer list contains shared errors across different datasets and is not restricted to strand biased positions. The systematic errors in both of my classes are shared variants among 300 from 511 samples in the InhouseDB data set (but these are not common variants having MAF < 30 %). Moreover, I filtered the 9mer list with the GATK best practice filters for SNPs and Indels (cf. Chapter2) and found that only 11% of these

[87] https://bitbucket.org/tobiasmarschall/discovering-cse

systematic errors can be detected or filtered out by the best practice filters (which include a filter for strand bias).

The RSEs reported in the two error classes can be used to filter or mark systematic errors in any other variant list. However, I found that some of these errors are true variants in the NIST/Broad truth set list (cf. Table 4.3 & Table 4.4) or present in DBall. This is not surprising as some rare variants or sample/disease specific variants can be true even at systematic error prone regions. Moreover, I also observed that Haplotype caller (HC) did not call some of these errors (especially errors in regions of paralogous alignment (cf. Chapter 2) (cf. Table 4.3 & Table 4.4). The new generations of variant callers like Haplotype caller or Platypus, don't rely on the alignment entirely but perform local reassembly of haplotypes[88] containing mismatches on the reference, thus, they can avoid some of the systematic errors (but not all) due to paralogous alignment. However, they can also miss some true variants in these regions because of this behaviour (cf. Chapter 2). Therefore, in RSE lists (cf. Appendix) I have provided some additional annotations showing whether a variant is present in NIST or DBall list or whether it is called by HC etc. This can help users to decide (based on the aim of the study) that up to which extent they should filter systematic errors in their variant lists.

FilterRSEs tool

To filter RSE locations (FPs due to RSEs) from any variant list (in the VCF format), I developed a script (named: "FilterRSEs") in the Perl programming language[89]. It compares the variant locations in the given VCF file and the RSE list, and performs relevant action in two different modes: annotation and filter. In annotation mode, it just annotates the variant with RSE associated annotations (if the variant location is present in RSE list): PA, NIST, DBALL, 17mer. It appends the INFO field of the VCF file[90] with all of these annotations and their respective values. For example, if an RSE location is called by other pipeline I used (BWA-MEM and GATK HC), then "PA=Yes" will be appended to

[88] http://ghr.nlm.nih.gov/glossary=haplotype

[89] https://en.wikipedia.org/wiki/Perl

[90] http://samtools.github.io/hts-specs/VCFv4.2.pdf

the INFO field of that variant, otherwise “PA=NO” (if the RSE is not called by the other pipeline). Similarly, NIST and DBALL also contain two values “Yes” or “NO”, depending on the presence of the RSE location in the NIST list and in DBall (cf. Section “Validation of kmer list”). Along with the “Yes” value, the DBALL field also contains the rsid and MAF value of the RSE location. The annotation “17mer” contains the string of 17 bases around the RSE location (8 bases upstream, reference base at the RSE location, 8 bases downstream). The description of these annotations will also be added to the header of the VCF file. Therefore, the annotation mode provides some flexibility to user, who can consider the RSE annotations presented in the VCF file when deciding about a variant. For example, if the study focuses on the common variants (present in different human populations), then MAF values in DBALL field can be helpful in order to decide whether a certain variant should be considered as an RSE or a common variant.

In “Filter” mode, the tool filters out the FPs due to RSEs from the given VCF file and produces two different VCF files: one is free from RSEs and the other contains only the filtered FPs due to RSEs. This mode provides different filtering options using different combinations of annotations like P, C_MAF, C, N etc., which also offer lots of flexibility to user. For example, the default option of tool filters out all RSE locations from the VCF file irrelevant to the associated annotations, whereas the “C_MAF” option offers filtering of RSE locations based on a given MAF value. Overall, these different filtering options can provide “safe” filtering, even in the filter mode. Details of all of these filters as well as the pseudocode of “FilterRSEs” can be found in the Appendix.

Exploration of systematic error

I further investigated the detected RSEs to explore their characteristics (cf. Table 4.3 & Table 4.4). I observed that the majority (74-77% in class 1 and 64-67% in class 2) of RSEs are mapped at only (or mainly) one strand (either forward or reverse) of the reference sequence. However, most of these are not detected by the standard strand bias filter used in GATK’s best practice filtering strategies. I also found that some of the errors are mapped on both strands (23-26% in class 1 and 33-36% in class 2) as well, which supports our assumption that the investigation around only strand biased (i.e.

supported by only one strand) errors (as performed in previous studies) is not enough to explore all systematic errors (cf. Section 4.1.2). Moreover, I found that a significant portion of total errors, approximately 27% in class 1 and 35% in class 2, are pipeline artefacts and can be avoided by using the recent sophisticated tools/algorithms (like Haplotype variant caller) (cf. above paragraph).

In order to see the distribution of these errors throughout the genome, I calculated the number of RSEs per chromosome in class 1. The distribution of RSEs captured by the Prefix model (PM) (cf. Figure 4.5) and Suffix model (SM) (cf. Figure 4.6) is almost identical as both models are capturing the same errors with upstream and downstream 9mers to the error position, respectively. However, SM captured a few more RSEs (not a significant difference) than PM due to the differences in the 9mer construction procedure.

In All Dataset	**Prefix Model (PM)**			**Suffix Model (SM)**		
	FPs	**In NIST**	**Not called By HC**	**FPs**	**In NIST**	**Not called By HC**
At both strand	357	53	126	324	48	112
At only one strand	1031	260	264	1134	290	281
Total FPs	1388	313	390	1458	338	393

Table 4.3 Overview of systematic errors in class 1: Platform dependent (shared by all datasets).

In only Nimblgen datasets	**Prefix Model (PM)**			**Suffix Model (SM)**		
	FPs	**In NIST**	**Not called By HC**	**FPs**	**In NIST**	**Not called By HC**
At both strand	300	6	123	278	5	113
At only one strand	551	84	172	583	75	192
Total FPs	851	90	295	861	80	305

Table 4.4 Overview of systematic errors in class 2: Target enrichment dependent (shared by only Nimblgen Kits).

The RSE distribution of class 1 errors showed that a few chromosomes (CHR) are having a significantly higher number of errors than the others. Thus, I focused on only these chromosomes: 1,2,6,7,11,17 to investigate the cause of the high error load. I observed that these chromosomes are containing some genes, which are having a major portion of errors. For example, approximately 55% of the total errors on chromosome 11 (mainly SNPs) are on the MUC6/2 genes (notation means *MUC6* and *MUC2*), with the majority of them in *MUC6*. The *ANKRD36* gene is having 47% of total errors on chromosome 2, the *MAP2K3* gene (on chromosome 17) is having around 44% and the *CDK11A/B* genes (mainly in *CDK11A*) (on chromosome 1) are having around 19% of the total errors on that chromosome.

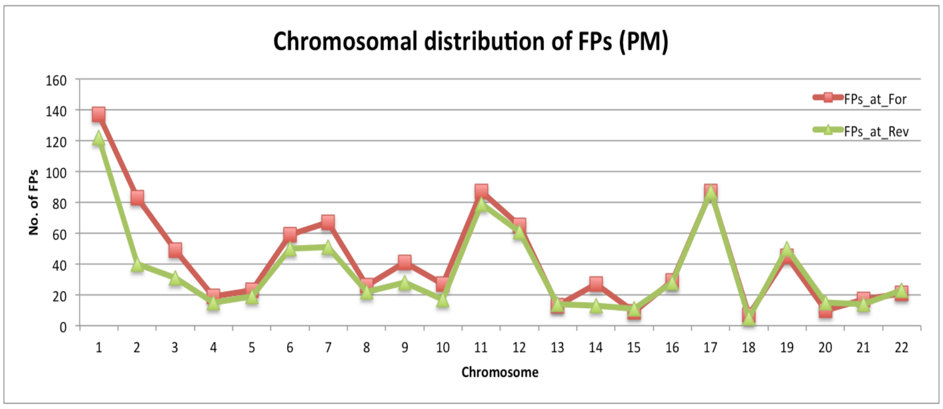


Figure 4.5 Distribution of errors throughout the genome captured by Prefix model (PM). In the figure, X-axis shows the chromosome name, whereas the Y-axis shows the number of errors. The red line shows errors present on the forward strand, whereas the green line shows errors on the reverse strand.

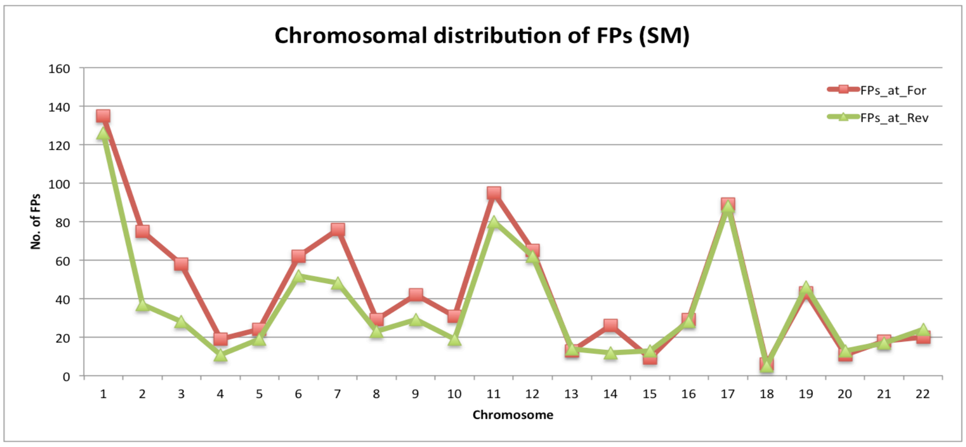


Figure 4.6 Distribution of errors throughout the genome captured by Suffix model (SM).

After the investigation of all of these genes (having a high number of RSE), I found that these are having many paralogous genes or similar genes as computed by Similarity Matrix of Proteins (SIMAP) (based on their proteins alignments) (Rattei et al., 2006, 2010) or contain highly polymorphic repetitive regions or LCR regions (cf. Chapter 2). These facts are the main cause of misalignments by alignment algorithms in these regions that result in lots of mismatches. Alignment algorithms can find multiple mapping hits in these regions. In other words, they can align reads that belong to one gene to one of its paralogous genes due to high sequence similarity (cf. Chapter 2). Figure 4.7 shows the IGV[91] (Robinson et al., 2011) visualization of aligned reads (aligned by both BWA-MEM and backtrack algorithm) on the reference sequence at the *MUC6* gene. The grey colour (wavy) part below the coordinate axis in both diagrams is the coverage track, which shows the read coverage at every genomic position. It is coloured in grey when only the reference allele is present in the reads while the presence of alternative alleles at a position is marked in green, blue, orange, and red colour for A, C, G, and T, respectively. Both diagrams show lots of coloured bars in the coverage track, which depicts lots of mismatches in the visible region. It is known that the MUC6 gene is highly polymorphic having short tandem repeats (STRs) containing G and T nucleotides (e.g. GGT, GCT, GT, etc.) (cf. Figure 4.8). Moreover, it is having many paralogous genes like *MUC2*, *OTOGL*, *OTOG*, *ZAN* etc.[92]. These two facts are the main cause of misalignments by both alignment algorithms, which result in a lot of mismatches in the MUC6 gene. However, I found that Haplotype caller can avoid very few of these errors.

I fetched the information of paralogous genes of highly mutated genes (cf. Table 4.5) from gene cards[93] (Belinky et al., 2015) and found that some of these genes are either belonging to one gene family or are paralogous to each other or are having other highly mutated genes as similar genes (based on the protein sequence alignment: SIMAP (Rattei et al., 2010). For example, *CDK11A/B*, *CLCNKA/B* and *MUC6/2* belong to the same gene family and are paralogous to each other. *PRIM2* gene contains 38% of RSE in chromosome (CHR) 6, but no paralogs for this gene cards. However, its cryptic paralogs,

[91] https://www.broadinstitute.org/igv/

[92] http://www.genecards.org/cgi-bin/carddisp.pl?gene=MUC6#paralogs

contain only exons 6–14 of the original transcript, are mentioned in study (Genovese et al., 2013). They also found that these paralogs are the cause many FPs (SNPs and CNVs).

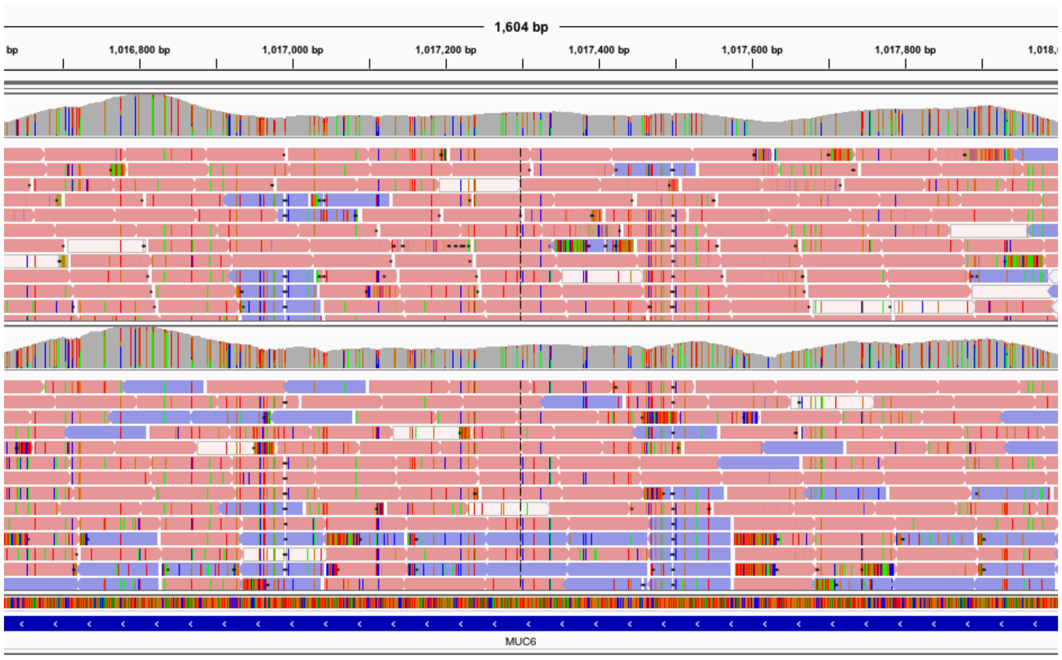


Figure 4.7 Alignment visualization by IGV in the MUC6 gene region. Upper diagram shows alignment by BWA-backtrack (aln) algorithm, lower diagram shows alignment by BWA-MEM algorithm.

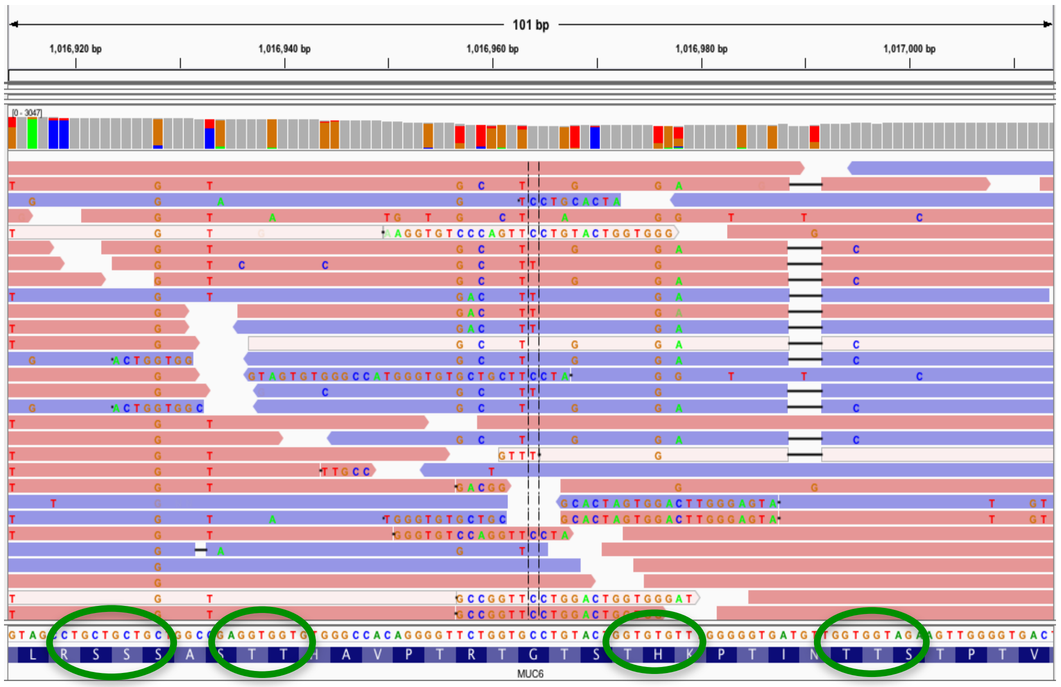


Figure 4.8 Alignment visualization by IGV at location 11: 1016963 of MUC6 gene. Surrounding this position (highlighted by vertical line in black colour) there are lots of STRs (repeats of GGT, GCT, GT, etc.) highlighted with ovals in green colour.

Frequent RSE gene	**% RSE**	**CHR**	**Paralogous genes**[93]
CDK11B	3%	1	CDK12, CDK13, CDK10, CDK11A, CDK9,
CDK11A	16%	1	CDK12, CDK13, CDK10, CDK11B, CDK9
CLCNKB	5%	1	CLCN7, CLCN2, CLCN3, CLCN, CLCN1, CLCN4, CLCN6, CLCNKA
PDE4DIP	10%[94]	1	CDK, RAP2
ANKRD36	47%	2	POTEJ, ANKRD18A, ANKRD30BL, POTEC, ANKRD20A, POTEF, ANKRD7, ANKRD62, ANKRD30A, ANKRD36C, etc.
ANKRD36B	3%	2	SIMAP similar gene to ANKRD36
SEC22B	4%	1	SEC22A
PRIM2	38%	6	No paralogs are available in Gene cards, but its cryptic paralogs are mentioned in study (Genovese et al., 2013)
DUSP22	6%	6	DUSP26, DUSP1, DUSP21, DUSP13, STYX, DUSP18, DUSP27, DUSP19, DUPD1, DUSP28, DUSP3
MUC12	5%	7	MUC17
TRBV7-3	7%	7	TRBV2, TRBV24-1, TRBV3-1, TRBV, TRBV6-8, TRBV4-1, TRBV2-1, TRBV23-1, TRBV10-2, TRBV28, TRBV7-1, etc.
PRSS1	26%	7	KLK12, KLK14, KLK11, KLK4, PRSS37, KLK13, PRSS8, PRSS3, KLK1, KLK, KLK6, KLK7, KLK1, KLK2, KLK10, etc.
MUC6	42%	11	OTOGL, OTOG, ZAN, TECTA, VWF, BMPER
MUC2	13%	11	OTOGL, OTOG, MUC6, ZAN, TECTA, VWF, BMPER
OR4C3	7%	11	OR4A, OR4C46, OR4B1, OR4C11, OR4S2, OR4A47, OR4C13, OR4C16, OR4X1, OR4S1, OR4A1, OR4C1, etc.
OR9G1	5%	11	OR9G4 and SIMAP similar gene OR4C3
USP6	7%	17	USP4, USP43, USP1, USP11, USP19, USP32, USP16, USP31, USP4,
CCDC144NL	5%	17	CCDC144A CCDC144B
MAP2K3	44%	17	MAP2K7, MAP2K2, MAP2K6, MAP2K4, MAP2K, MAP2K1
KCNJ12	10%	17	KCNJ3, KCNJ, KCNJ4, KCNJ14, KCNJ1, KCNJ1, KCNJ16, KCNJ10, KCNJ9, KCNJ8, KCNJ2, KCNJ6, KCNJ11

Table 4.5 List of genes having high RSEs and their paralogous genes.

[93] Information taken from Gene cards database: http://www.genecards.org/

[94] This gene is also observed in class 2 (not analyzed in detail) with a significant level of FPs.

Moreover, I found that a few of these paralogous genes are located on the same chromosomes. I also compared my list of genes with the list of genes provided in the study conducted by (Fuentes Fajardo et al., 2012). In their study, they analysed exome-sequencing data from 118 individuals in 29 families and reported lists of genes having FPs (heterozygous calls). I found that the *MUC2/6/16*, *MAP2K3*, *PDE4DIP* and *USP6* genes are common in both lists. Overall, the high sequence similarity to other genes or presence of highly polymorphic repeats (like in the *MUC6/16* genes), make these genes more prone to misalignments, which produce the high number of FP variant calls in these genes.

Investigation of strong RSEs

In order to find patterns associated to RSEs, I further investigated class 1 (RSEs shared by all datasets) in detail. I filtered the list with OEratio < 1 (filtered-out approximately 85% of list) followed by best practice filter that reduced the 9mers list by up to 87% (cf. Table 4.2). In this filtered list, there were no RSEs from highly mutated genes except for some in the *MUC6* gene. In total, this list contains only 134 RSEs (with 211 associated 9mers) and 20% RSEs (or 33% 9mers) of these are due to the misalignments in the *MUC6* gene.

For the further exploration, I removed 9mers belonging to the *MUC6* gene regions. I investigated whether the remaining RSEs are sequence specific errors (SSEs). For this purpose, I examined the remaining 9mers (I will refer to them as “SSE-associated 9mers”) to find out if the remaining 9mers share some small sequence motifs or whether any nucleotide (or combination of nucleotides) is highly frequent throughout the list. I used the online tool Weblogo[95] to generate sequence logos, i.e. “graphical representations of the patterns within a multiple sequence alignment” (Crooks, 2004) (Schneider & Stephens, 1990). I generated sequence logos (with frequency plots options) for 9mers generated by both prefix and suffix model (cf. Figure 4.9). However, I found that these logos are not very informative and the only fact I observed is that the error mainly occurred at reference base ‘T’ (tallest base at last and first position of PM

[95] http://weblogo.berkeley.edu/logo.cgi

and SM respectively), which has been also observed by (Meacham et al., 2011) where they found the GGT motif upstream of the error position (with T reference base at the error site). Additionally, the presence of the GGC motif up to 10 bases upstream of the error base (cf. Figure 4.9) can be anticipated in some 9mers, which are responsible for some systematic errors and were first observed by (Nakamura et al., 2011). However, the logo for the SM model shows that some 9mers might also have GGC motifs present in 10 bases downstream as well. To confirm the presence of GGC or GGT, I screened the 9mer list (excluding MUC6 9mers) for their presence and found that approximately 10% and 11% of the error sites are having GGC and GGT upstream of the error position, respectively (cf. Table 4.6). In contrast, 9% and 18% error sites have a GGC or GGT motif downstream, respectively. (Nakamura et al., 2011) proposed that GGC motifs can be a cause of dephasing, which can induce base calling errors. They found that the wrongly called base (mismatch) at the error site is either the first or the second preceding reference base. I also observed such dephasing patterns due to the GGC motif in the upstream of one of the RSE in our list, where the mismatch ‘T’ is the second preceding reference base (cf. Figure 4.10).

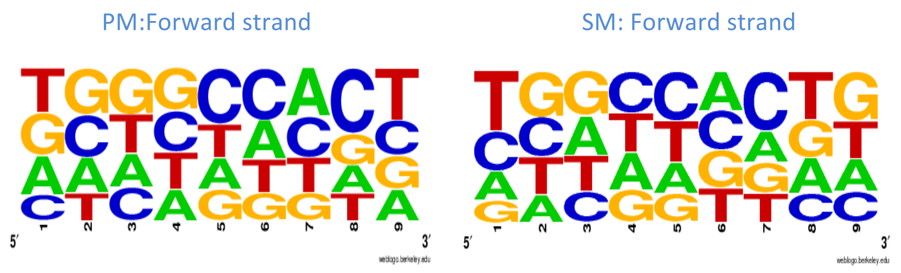


Figure 4.9 Sequence logos (from Weblogo tool) for 9mers at forward reference strand. The top row contains the most frequent nucleotide at each position (at X-axis, range 1-9 represents first base till last base of the 9mer) and the length of each character is directly proportional to its frequency (the tallest character is having the highest frequency at the respective position).

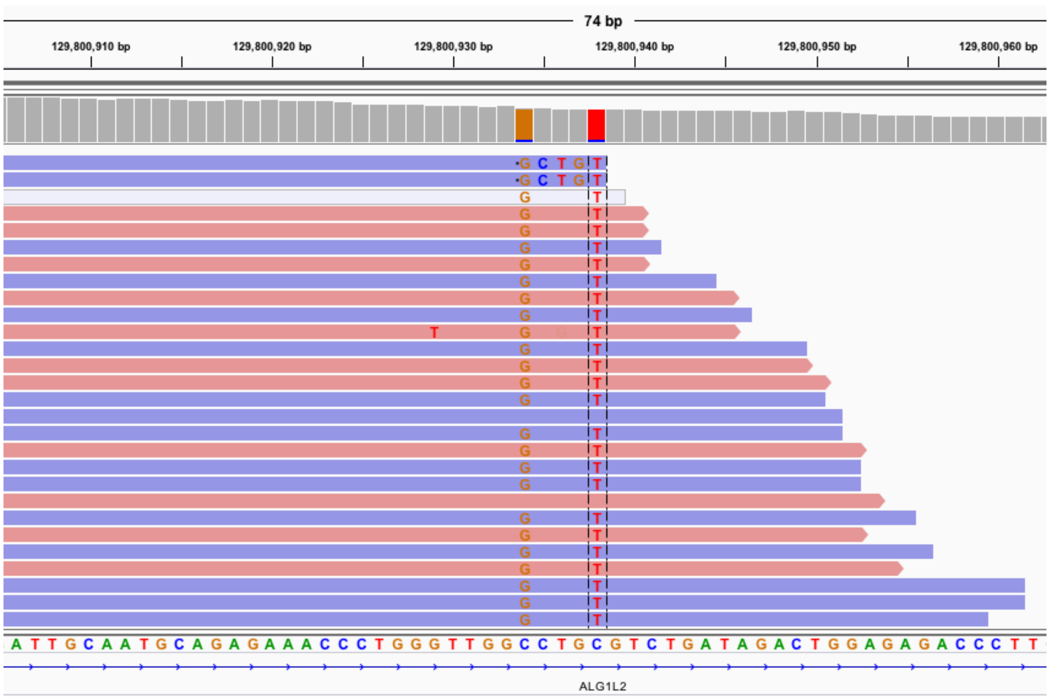

74 bp
129,800,910 bp
129,800,920 bp
129,800,930 bp
129,800,940 bp
129,800,950 bp
129,800,960 bp
A T T G C A A T G C A G A G A A A C C C T G G G T T G G C C T G C G T C T G A T A G A C T G G A G A G A C C C T T
ALG1L2

Figure 4.10 Alignment visualization of a systematic error by IGV. The wrong base T is highlighted with vertical lines in black colour at chr 3: 129800938.

3mer (Upstream)	Occurrence (%)	3mer (Downstream)	Occurrence (%)
CTG	19.87	CTC	17.73
ACC	19.21	GGT	17.73
CCT	17.89	TCC	17.09
CCA	16.56	TGG	16.46
GCA	15.24	CGG	16.46
TGC	14.57	TCA	15.19
GCC	13.91	CAC	15.19
GAC	13.91	CCT	15.19
GTG	13.91	GTC	13.93
TGG	13.91	TTC	13.93
GGC	9.94	GGC	8.8
GGT	11.26		

Table 4.6 List of top 10 common motifs of length 3 (including known motifs occurrences) in RSEs associated 9mers in both upstream and downstream.

To find some other common or highly frequent sequence motifs, I considered all shorter kmers of lengths 2 to 8 contained in our list of 9mers. I found a few intersting 3mers like

“CCT”, which appreared either in upstream or downstream of the error positions (18% and 15% of total 9mers in upstream (151 9mers) and downstream (158 9mers) respectively) (cf. Table 4.6). In the context of 2mers, I found approximately 46% and 48% of 9mers are having “CT” and “CC”, respectively, upstream of the error positions. 44% and 40% of 9mers are having “CT” and “CC”, respectively, downstream of the error positions. The higher ocurrence of the “CT” motif is also observed in the 3mer “CTG” upstream and “CTC” downsteam. Both motifs showed the highest occurrence among all top 10 motifs (cf. Table 4.6) with approximately 20% and 18 % for “CTG” and “CTC”, respectively. Furthermore, the reference base “T” is found to be the most error affected nucleotide. Approximately, 34% of 9mers (among 211) are having “T” (“A” at reverse strand) as the reference base at the error position, which can be also observed from the logos (cf. Figure 4.9) and was also concluded in the study by (Meacham et al., 2011) (“stronger tendency of error at A or T than at C or G”). Moreover, approximately at 20% of the error postions, “T” is having “C” as an error base (i.e. T>C sustitution) and 15% show an A>G substitution on the forward strand. At the reverse strand, I found 18% of postions with A>G and 17% with T>C substitutions. Among 211 RSE associated 9mers, only 5 are having Indels which is expected as Illumina sequencing has a higher substitution error rate than Indel error rate. As known, I also found all Indel errors in STRs which is due to the alignment inacurracy in repetitive regions (cf. Chapter 2).

Secondary structure analysis

I also performed secondary structure analysis on sequence motifs in the SSE-associated 9mers list. At first, I combined the 9mers from both PM and SM to construct 17mers, which contains the surrounding sequence sequence motifs in both upstream and downstream direction of the error position, which is the 9th base of the 17mer. This merging further reduced the list to 120 (was arround 150) 17mers. The idea behind the secondary structure analysis is based on the observation found in study (Nakamura et al., 2011), where they observed that a few sequence specific errors can be triggered by inverted repeats which might cause dephasing during sequencing (cf. Section 4.1.1). Thus, the aim of this analysis is to find some 17mers having high probabilty to fold or form some sort of secondary structure like a hairpin loop, which might be triggering associated RSEs.

CHR	Location	17mer	MFEstruct	MFE	PairProb	EnsblFE
14	106361561	AGGCCTGGCGGTAGGTT	.((((((....))))))	-4,7	.{(((((....)))))}	-5,16
11	1093158	ACCACCACTACGGTGAC	..((((.....))))..	-4,9	..((((.....))))..	-4,91
8	124664873	GAGGCACATATTGCCAC	..((((.....))))..	-4,8	..((((.....))))..	-4,91
8	124664873	GTGGCAATATGTGCCTC	..((((.....))))..	-4,6	..((((.....))))..	-4,78
12	2791130	GCACGTTCCGATGTGTG	((((((....)))))).	-4,4	((((((....)))))).	-4,57
21	11049395	GCGCCTAGTAATAGGGT	((.((((....))))))	-3,8	,{.((((....))))}}	-4,32
6	31922360	CGGGATCGAGACCGAGA	(((........)))...	-3,8	(((........)))...	-3,84
21	34915324	AATTGGCCCGTGCGCCT	....(((......))).	-3,6	....(((......))).	-3,65
9	130932398	CTTTGACTGTTGTCATT	...((((....))))..	-3,5	...((((....))))..	-3,63
9	130932398	AATGACAACAGTCAAAG	..((((....))))....	-2,9	..((((....))))....	-3,04
7	142168662	CAGTGGACGCTGGAGTC	((((....)))).....	-2,7	((((....)))).....	-2,85
2	97820434	CTGTGCAAAACGGTCCA	((((.....))))....	-2,1	((((.....))))....	-2,16
10	65225244	GGGGCGCTGACTCTCTT	(((.......)))....	-1,3	((({......)}}},..	-2,13
12	9573224	AACCACACCGCACAGGT	......(((.....)))	-1,9	......(((.....)))	-2,1
17	21319523	CCTGGAGACGGACGACT	.(((....)))......	-1,9	.(((....)))......	-2
6	31922360	TCTCGGTCTCGATCCCG	...(((........)))	-1,3	...(({....,...}},	-1,79
14	106361561	AACCTACCGCCAGGCCT	..(((......)))...	-1,4	..{{{..,...}))...	-1,78
12	9573224	ACCTGTGCGGTGTGGTT	(((.....)))......	-1,4	(((.....}}}......	-1,75
21	42843962	GTTCATCCACTGAGAGC	((((.((....))))))	-1,4	((((.{{....))))))	-1,73
8	27101204	ATGTGTTAAGGGGCGTA	((((........)))).	-1,1	((((.,......)))).	-1,63
3	129147418	GGGTATCAGATAAACCG	.(((.........))).	-1,2	,({{.........})).	-1,63
7	142168662	GACTCCAGCGTCCACTG	(((......)))....	-1,4	(((......)))....	-1,55
3	129147418	CGGTTTATCTGATACCC	.(((.........))).	-1,1	.(({.........})).	-1,52
21	11058227	CTGAAAGGTGTCGGCTC	((((......)))...	-1,2	((({......})))...	-1,51
6	112508769	ACTGATGCACTGCGGTT	((((........)))).	-1,1	{(((........)))}.	-1,46

Table 4.7 List of top 25 17mers according to the nucleotide pairing probabilities (highest to lowest).

I used the RNAfold method of the ViennaRNA Package 2.0 (Lorenz et al., 2011) to predict the secondary structures of single stranded DNA sequences in 17mers. In brief, I used RNAfold with partition function and MEA algorithms ("RNAfold --MEA –p") [96] to compute the minimum free energy (mfe) structure, partition function, pair probabilities and the maximum expected accuracy (MEA) structure. The detailed information of this alogorithm and usage of ViennaRNA Package can be found in (Hofacker, 2009; Lorenz et al., 2011). Table 4.7 shows the list of 17mers having high nucleotide pairing probabilities. The 4th ("MFEstruct") and 5th ("MFE") column of table ("MFEstruct") shows the

[96] https://www.tbi.univie.ac.at/RNA/RNAfold.1.html

predicted mfe structure of the 17mer in bracket notation and its free energy in kcal/mol. The bracket notation depicts a pairing of two bases i and j by a pair of matching parentheses, whereas unpaired positions are depicted by dots. The pictorial representation of the bracket notation of the mfe structure for the first and last 17mer from the table is shown in Figure 4.11. The 6th ("PairProb") and 7th ("EnsblFE") column contain the bracket notation of the pair probabilities and the ensemble free energy in kcal/mol. In this context, the parentheses show the positions having high probability to pair and dots show mostly unpaired positions, whereas curly brackets and commas show positions with low pairing probabilities. Higher negative value of free energy mean more stable structures.

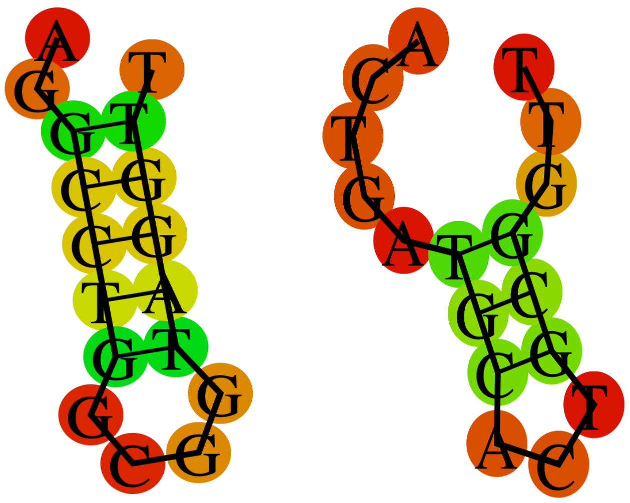


Figure 4.11 MFE secondary structure of first (left) and last (right) 17mer from the list of top 25 17mers.

Based on the computed mfe and the pairing probabilities of 17mers, I found that 25 (approx. 21%) among 120 SSE-associated 17mers could form some sort of secondary structure (cf. Table 4.7). So far the role of the detected 17mers in the origination of these SSEs is not clear, however, they can function in a similar way as inverted repeats explained by Nakamura and colleagues (Nakamura et al., 2011). These sequencing motifs are a part of the DNA template used during sequencing. If these motifs form a loop during sequencing, then the base elongation process (cf. Section 1.1.1) will be hampered for a certain time, which might cause dephasing. As explained previously, dephasing is known to trigger systematic errors.

4.3 Chapter summary

In this chapter,

- I presented a new approach to search for sequence motifs surrounding sequencing error sites. I screened 9mers in both upstream and downstream direction of the error position (in total 17mers).
- I restricted the 9mer list to the RSE locations by selecting only those variants that are shared (but are not common variants in the population) between at least 300 from 511 exome sequencing samples of our in-house dataset. I also validated this list with available resources for the control sample NA12878 (like confident variant calls provided by NIST).
- I compared errors in our list with 3 different datasets to investigate the nature of these systematic errors. I found that most of these are reproducible which supports our assumption that these are systematic errors.
- I constructed two classes of errors: Platform dependent errors and Target enrichment dependent errors. I prepared the lists of RSEs for these two classes. These lists are having all-important annotations (like location, MAF, etc.) and can be used to filter or mark systematic errors in any other variant lists. Moreover, I presented a tool “FilterRSEs” to mark or filter out RSEs from any variant list.
- I found that most of the errors are due to a few genes having paralogous genes or having highly polymorphic repeats. I have provided lists of these genes, which can also be used to mark/filter the variants in these genes.
- I screened only those RSE having OEratio < 1, for short common sequence motifs (3mer or 2mer) and found the presence of all known motifs near the error sites like GGC and GGT. Moreover, I also observed that the reference base “T” is having errors most of the times as already observed in previous studies. However, I observed the presence of a few sequence motifs of length 2 or 3 in either upstream or downstream of the error site, which is not reported so far.
- I found 25 sequence motifs that might trigger RSEs. These motifs have high probabilities to form some sort of secondary structure (like a loop), thus, can hamper the sequencing process for a while and result in a systematic error.

Chapter 5
Conclusion and outlook

5.1 Discussion and conclusion

Next generation sequencing (NGS) techniques are now well established and have been successfully used in many applications for both fundamental research and diagnostics purposes (e.g. whole exome or genome sequencing, gene panel sequencing, transcriptomics etc.) (Rabbani et al., 2014). Due to the rapid decrement in the costs, it is accessible to almost all research or diagnostic laboratories. A simple keyword search for "Next generation sequencing" in pubmed[97] results in approximately 10,000 entries with approximately 6000 entries since the last two years. However, the huge amount of data generated by NGS techniques requires an efficient and accurate bioinformatics analysis to make it meaningful.

In this work, I addressed different challenges associated with NGS data analysis. I categorized these challenges into two main categories: Efficient data processing and accuracy of analysis. We developed an automated data analysis workflow (Kawalia et al., 2015) for targeted DNA sequencing experiments (exome sequencing, gene panel sequencing, amplicon sequencing). During the development of our workflow, I performed extensive testing of bioinformatics tools required for analysis, both in terms of their efficiency on HPC cluster and the accuracy of analysis. Besides the open-source tools, I also developed some in-house tools for certain parts of the analysis workflow. In addition, I developed a few strategies for the efficient implementation of our workflow on the HPC cluster. To enhance the accuracy of the results, I explored systematic errors, which are usually undetected by state-of-the-art algorithms.

Analysis of sequencing data can be judged by two criteria: its specificity and sensitivity (cf. Chapter 2). A variant list, which represents the final result of analysis, should be

[97] http://www.ncbi.nlm.nih.gov/pubmed This and other subsequent URLs are accessed on 22 July 15.

highly specific (contain few false positive variants (FPs)) and highly sensitive i.e. should not miss any causative or true variant (no false negatives (FNs)). However, there is a trade-off between these two factors causing highly specific variant lists to miss some variants while highly sensitive variant lists can contain lots of FPs. Thus, a well-balanced adjustment of specificity and sensitivity is very important and still a challenge for the bioinformatics community. As far as I know, there is no analysis workflow, which can achieve 100% specificity and sensitivity. In chapter 2, I addressed these issues with detailed information about the individual analysis steps and the factors that can damage results in terms of both specificity and sensitivity. Raw sequencing data require a series of data analysis steps for quality control, sequence alignment, variant calling and variant filtering/validation. All steps are interconnected and each can have a bad or good effect on the consecutive step. For example, wrong base trimming (both bad quality bases and adapter bases), can discard a significant amount of the sequenced bases which might lead to misalignments or uncovered regions of the reference sequence and can cause false negatives. On the contrary, avoiding the quality control steps (especially on low quality data) can lead to FPs due to wrong alignments or alignments with many mismatches of bad quality bases or reads.

The alignment of the reads to the reference sequence has a significant impact on the specificity and sensitivity of the variant list. A specific alignment (zero or few mismatches allowed during alignment) can reduce FPs but can increase FNs. On the other hand, a sensitive alignment (allowing many mismatches during alignment) can increase FPs but lowers the risk of FNs. BWA-MEM, as one of the current alignment algorithms, performs read mapping with a good balance between both sensitivity and specificity. It tries to find an optimal alignment (maximum exact match extended by some mismatches) of a read sequence on the reference DNA sequence. However, it is not possible to align all reads accurately on the reference genome, especially reads having the typical read lengths between 100 to 200 bp generated from short read sequencing technologies (like Illumina). There are several reasons behind this, like sequencing errors, low complexity regions (LCRs) or highly divergent regions (HDRs) (less than 99.5% identity in the human genome) (cf. Chapter 2), paralogous genes and the incompleteness or errors in the human reference genome (Fuentes Fajardo et al., 2012; Sims et al., 2014; Treangen & Salzberg, 2012). Every cause mentioned above leads to alignment errors followed by variant calling errors that result either in FPs or FNs.

In addition, other sources that lead to FPs or FNs include sequencing errors like PCR errors, base-calling errors towards ends of reads (esp. 3' end), substitution errors near Indels etc. Some of these errors can be corrected by post alignment improvements like Duplicates removal, BQSR, Indel Realignment. I implemented these corrections in our data analysis workflow (cf. Chapter 3) and they are described in detail in Chapter 2. Moreover, a certain sets of FPs can also result from the limitations of a variant calling algorithm, either the implemented approach or the default parameter settings. Thus, to overcome the limitations of a single variant caller, I used four different variant callers: Samtools'mpileup, Platypus, GATK's Unified Genotyper and Haplotype Caller. These callers belong to two different categories of variant calling algorithms: nucleotide based and haplotype based variant detection. The nucleotide based algorithms use a Bayesian approach (H. Li, 2011) to call SNPs and Indels and treat each position independently (used by Samtools and Unified Genotyper). This approach relies on the alignment accuracy and can produce many FPs but provides sensitive variant calling. On the contrary, the Platypus and Haplotype Caller are the most recent and sophisticated haplotype-based callers, which can avoid some alignment artefacts (e.g. due to paralogous alignment). They do not rely only on the alignment but perform local denovo assembly (by building a De Bruijn-like graph) in order to find the correct haplotype, which is used for SNP/Indel identification. However, they are also not able to find the optimal alignment in LCRs (H. Li, 2014; Rimmer et al., 2014). Moreover, due to the different algorithms and parameter settings, the overlap between the variant calls from these tools is not very high. Overall, I found that the integration of the variant callers using different algorithms is better than using a single variant calling algorithm to achieve highly accurate and sensitive variant calls. This fact is also reported by many data analysis and tool comparison studies (Bao et al., 2014; H. Li, 2014; Trubetskoy et al., 2014).

After performing quality control on raw data, alignment and post alignment improvements followed by sophisticated variant calling, the final variant list still contains lots of FPs. Thus, FPs filtering and variant evaluation also plays an important role in the analysis and can drastically change the variant list. The principle of trade-off between sensitivity and specificity also applies here. Stringent filtering can reduce the number of FPs but can also increase the number of FNs. Therefore, I apply standard filters like VQSR and GATK best practice filters (like MQ, Qual, DP, Fisher strand) (cf. Chapter 2) for every exome sequencing data set by default. I also apply specialized filters for different data types. For example, sequencing data from Ion torrent sequencers contain more Indel errors (due to homopolymer errors during sequencing) compared to Illumina sequencing data. Thus, I filter Indels called in homopolymer regions to avoid Indel errors. Recent variant callers, like Haplotype Caller and Platypus, provide lots of other useful annotation like %GC content, variant called in homopolymer or LCRs, string of the bases surrounding the variant, allelic frequency (heterozygous or homozygous), genotype quality score, etc. It is known that some systematic errors are surrounded by certain sequence motifs like inverted repeats, GGC or GGT (Meacham et al., 2011; Nakamura et al., 2011). Moreover, coverage bias has been observed mainly in GC rich regions and Indels error mainly in LCR regions (Sims et al., 2014). Thus, all of these annotations can also be used, either as a combination of two or more annotations or individually, to filter out some specific artefacts. Many studies have shown the effects of other filters than the standard best practice filters in false positive reductions (Fang et al., 2014; Hwang et al., 2014; H. Li, 2014; Reumers et al., 2011). At last, variants can be filtered or prioritized based on the prior knowledge about the disease or design of the study (e.g. based on pedigree, trios etc.) or based on the predicted functional effects of the called variant on the phenotype and the variant frequency from publicly available data like 1000 Genome project, DBSNP, ExAC etc. (cf. Chapter 3). We have implemented most of these filters in our exome analysis pipeline Varbank (https://varbank.ccg.uni-koeln.de/) and are continuously exploring effects of other filters. Varbank provides access to the variant list with different filtering options, which facilitates variant prioritization or candidate variants list creation.

The variant filtering is a very critical step as it can filter FPs but can also produce FNs. Sometimes a standard filter can have a devastating effect on certain types of data. For example, I filtered out a true variant by using the QD < 2 filter (filters variants below the mentioned score) in one of our highly covered old gene panel sequencing data (sequenced in 2010 on an Illumina sequencer). I found that at the down-sampled read coverage of 250 reads this variant is not filtered by the QD filter, but at high coverage (> 500 reads) this variant disappeared from the final variant list. I assume that this is because of many low quality reads in the called region whose accumulation is decreasing the QD score as it is a normalized value of the variant quality score with respect to read depth. Thus, the evaluation of filtering effects is very important. Recently, the Genome in a Bottle Consortium (GAIB)[98] (hosted by NIST[99]) has provided highly confident SNP, Indel and homozygous reference genotype calls for the genome of the public NA12878 sample[100] (one of the samples of the 1000 Genome project) (Zook et al., 2014). They have used different sequencing technologies like Illumina, Ion torrent, 454, complete genomics to sequence the genome of this sample and used different data analysis pipelines to call the genotypes. This integration avoids the systematic errors from a single platform, alignment or pipeline artefacts and provides a highly accurate variant list. These data has been used in many studies for validation of their variant calls (Hwang et al., 2014; Kelly et al., 2015; Linderman et al., 2014). We also performed exome sequencing of this individual and used this variant list to benchmark our pipeline and the generated variant list in terms of specificity and sensitivity. The transition versus transversion ratio (Ti/Tv) is another criterion to evaluate a variant list or the effect of a filter. As the transitions are more frequent than the transversions (due to methylation of C in CpG islands), this ratio can indicate how much your variant list deviates from general expectations. The Ti/TV ratio should be 0.5 for FPs and a good quality variant list for an exome should have a Ti/Tv ratio around 2.8 (DePristo et al., 2011). Thus, after filtering of FPs, this ratio should increase and tend to reach the expected value, but if it is decreasing then the filter might not be good for this type of data and should not be

[98] https://sites.stanford.edu/abms/giab

[99] http://www.nist.gov

[100] https://catalog.coriell.org/0/Sections/Search/Sample_Detail.aspx?Ref=NA12878&Product=DNA

applied. Overall, the selection of data analysis tools, their parameter settings and filtering strategies have significant impact on the final results and require extensive testing and evaluation. Moreover, a data analysis pipeline should be configured based on data characteristics like read length, coverage, sequencing platform, aim of study etc.

As mentioned above, the recent data analysis tools or pipelines (including our data analysis workflow) are capable of detecting and filtering most of the known errors. However, certain systematic errors due to sequencing bias or alignment artefacts (like paralogous alignment) are hard to detect by existing tools. These errors appear systematically throughout multiple sequencing experiments from the same sequencing platform and analysed by the same analysis pipeline. We coined these errors as "Recurrent Systematic Errors" (RSEs). In Chapter 4, I presented a new approach to detect these errors. I screened 9mers (sequence motifs of 9 consecutive nucleotide bases) in 11 exome sequencing samples sequenced in-house and counted their observed and expected occurrences. Furthermore, I computed OEratios for the screened 9mers (i.e. Expected 9mer counts/Observed 9mer counts). In order to focus on systematic errors only, I restricted the 9mer list to those occurring at RSE sites. These RSE sites were derived from shared (but not common variants in the population) variants in 511 exome sequencing samples of our in-house dataset. I also validated this list with the confident variant calls for sample NA12878 provided by NIST and the variant calls for the same sample but called by a different analysis pipeline. I compared the shared 9mers throughout the 11 samples at RSE locations with those from 3 other datasets to validate the reproducibility of these systematic errors. From this list, I formed two different classes of systematic errors: Platform dependent errors (shared by all datasets sequenced irrespective of target enrichment kit) and target enrichment dependent errors (shared by only those datasets sequenced using the same target enrichment kit). I prepared the lists of systematic errors for these two classes. These lists are having all important annotations (like location, strand, MAF, called by NIST or other pipeline etc.) (cf. Chapter 4) and can be used to filter or mark the systematic errors in any other variant list. For this purpose, I developed a tool "FilterRSEs", which can just mark or filter out RSEs from any variant list. It is a very flexible tool and provides different filtering options to customize the filtering.

I further analysed the platform dependent errors to explore the characteristics of RSEs. I found that a wrong read alignment in LCRs and regions belonging to paralogous genes is the major cause of RSEs. I observed a high error load in some chromosomes (1,2,6,7,11,17) compared to others. These chromosomes contain some genes containing the majority of errors, for example, approximately 42% and 44% of the total variants (mainly SNPs) were found in the *MUC6* gene on chromosome 11 and in the *MAP2K3* gene on chromosome 17. I found that all of these genes having high RSEs either belong to one gene family, are paralogous to each other, or have a high sequence similarity (based on the protein sequence alignment) with other genes. Moreover, in the vicinity of error sites, presence of LCRs or highly polymorphic repeats in certain genes (like in the *MUC6* or *MUC16* genes) is observed.

LCRs, like short tandem repeats (microsatellites) are present throughout the human genome (approximately 50% of the human genome is filled with repetitive sequence) (Treangen & Salzberg, 2012). Thus, a read from a repetitive region can map at multiple positions with many optimal alignments to the reference genome with or without mismatches. In this case, the aligner picks the mapping with the highest alignment score or, in case of equal scores, picks one randomly which can lead to either a FP or a FN. If two reads map at the same location, for example, one with a mismatch of one base and another with a deletion of a few bases, then the first read gets a higher alignment score (by standard scoring functions) and a FP SNP can be reported if the true variant is the deletion present in the second read (Treangen & Salzberg, 2012). Similar situations can arise in regions that belong to one of the genes from a family of paralogous genes where a read can have multiple mappings due to the sequence homology of the genes.

Nevertheless, many false positives in these regions can be avoided by filtering of variants belonging to LCR regions which has been successfully used in many studies (H. Li, 2014; Lucas Lledó & Cáceres, 2013; Reumers et al., 2011). Similarly, lists of the most notorious paralogous genes and significantly mutated genes (SMGs) (containing variants most of the time irrelevant to sequencing experiments) can be used to filter out the variants belonging to these genes (Dees et al., 2012; Fuentes Fajardo et al., 2012; Watson, Takahashi, Futreal, & Chin, 2013). I also compiled a list of genes having a high number of RSEs (in Chapter 4), which can also be used to filter-out or mark the variants belonging to these genes. However, these filtering strategies should be opted based on the aim of study/experiment, as they can lead to the loss of true variants (FNs). The FNs might be more devastating than the FPs (esp. in diagnostics settings) as these variants can be the main contributor to a genetic disease.

I expanded our analysis further to focus only on a particular type of systematic error: Sequence-specific errors (SSEs). I filtered platform-dependent errors for OEratio < 1 and performed screening for common motifs (of different lengths: 2 to 8mers) among RSE associated 9mers. I found that the significant portion of errors is surrounded by a few motifs (3mers) either in upstream or downstream direction. This detected motif is so far not reported in association with systematic errors like known the "GGC" and "GGT" motifs (Meacham et al., 2011; Nakamura et al., 2011). However, I have not analysed its effect on systematic errors or the probable reasons behind its presence near error sites. I combined the 9mers both upstream and downstream direction of the error position to construct 17mers (the 9th base is error position) and performed secondary structure analyses on these 17mers. I found that 25 SSE-associated 17mers could form some sort of secondary structure. So far the role of these 17mers in the origination of the SSEs is not clear. However, it has already been hypothesized that the inverted repeats could cause dephasing (cf. Chapter 4), which is one of the causes of systematic errors. These inverted repeats could form loops during the sequencing process and inhibit the base elongation process for some time, which results in dephasing (Nakamura et al., 2011). Similarly, the constructed 17mers can be a part of the DNA template used during sequencing. If these motifs form a loop during sequencing, then the base elongation process (cf. Section 1.1.1) will be hampered for certain time, which might cause dephasing.

In addition to the accuracy, efficiency of the data analysis workflow is also an important aspect. Analysis of sequencing data requires a series of actions until the identification of a relevant mutation is possible (e.g. data cleaning, sequence alignment, variant calling, etc.). This requires a significant amount of time and lots of manual work. Thus, to manage and process huge amounts of sequencing data generated by the NGS sequencers at the CCG with little manual efforts (and manual errors), an automated data analysis pipeline is required. Furthermore, the workflow should be fast and should provide an easy access to the analysis results. So that, it can allow the researchers or clinician to analyse data and assess the results in a short time, which can speed-up research or diagnostics around diseases.

We have developed a fast and fully automated workflow for NGS data analysis (cf. Chapter 3). It is implemented and optimized on HPC systems and able to process 290 exomes per week on the current IT-infrastructure. This throughput is achieved by different parallelization strategies that enable proper exploitation of HPC resources. We used the MapReduce[101] approach in which large data is split into chunks and processed in parallel. In this approach, the number of chunks needs to be selected carefully when splitting the data, as it is directly proportional to the number of the parallel processes. In case of many chunks, too many processes run in parallel, which leads to longer waiting time in the queue when resources are not available. On the contrary, if chunks are big in size, then completion of a single task takes a long time and less gain in speed is achieved by parallelization. Similarly, the number of threads should be selected carefully. Some tools do not show significant improvement in speed beyond a certain number of threads. In that case using more threads blocks some resources, which can be better used for another parallel task. Besides threads or number of processes, appropriate memory required by the tools should be used for optimum exploitation of the computational resources. The HPC systems are complex and can be destabilized easily by a wrong action of any user when they are multi-users or shared clusters. Thus, we have used some design principles and special measures to make our workflow more

[101] https://www.usenix.org/legacy/event/osdi04/tech/dean.html

stable, robust, and easy to maintain so it can run smoothly on the HPC infrastructure. The workflow is organized as a collection of modules depending on the analysis task. These modules can run individually or in combination, thus, the workflow can be used to analyse various types of NGS (DNA) sequencing (both single-read and paired-end) data like whole exome or genome as well as any target enriched data. We mainly analyse human sequence data but the workflow can handle any organism with a known reference genome. The results from the workflow can be downloaded or analysed via the varbank web interface. The user can browse through the variant lists, can apply different filtering strategies (to make candidate gene lists) and can also crosscheck the variant by viewing the alignments. Since its launch in October 2012, we have analysed around 6000 exomes and uncovered the genetic background of various diseases (Ehmke et al., 2014; Lal et al., 2014; Schubert et al., 2014).

5.2 Outlook

A decade after their emergence, NGS technologies are still evolving at rapid pace. The introduction of HiSeq X Ten also broke the 1000 dollar genome barrier for whole genome sequencing. This increased the accessibility of whole genome sequencing, which produces approximately 8-10 times more data than exome sequencing. I expect that it will pose new data processing and storage challenges in future. We will adapt our data analysis workflow accordingly. In this context, we will examine the advantages of Big Data solutions for NGS data analysis. Nowadays, a lot of developments is going on to build some dedicated hardware for fast and efficient execution of some typical tasks in NGS data analysis. Moreover, cloud-computing strategies are also becoming popular for NGS data analysis. We will continuously investigate these developments to make our workflow fit for future requirements. With the performance growth of NGS technologies, bioinformatics development efforts for better data analysis are also growing. Recently, many new sophisticated algorithms have been released for either alignment of reads or for variant calling. I expect many more new tools and algorithms in the future and will constantly evaluate these tools for inclusion in our data analysis workflow.

As NGS technologies are now moving into clinical research or diagnostics, the accuracy of data analysis is a very critical aspect. Recently, gene panel sequencing has become more popular in these settings, as it is highly specific (with high coverage) and provides better accuracy than whole exome and genome sequencing at lower cost. However, this is a very specific approach and not suitable for all research and diagnostic applications. Exome and genome sequencing offer a broad range of applications, but they are more prone to sequencing errors like systematic errors. I will expand my systematic error detection approach and try machine-learning algorithms for better classification or detection of sequencing errors. I will expand analysis on our newly detected 3mers and other 17mers (forming secondary structure), to find out its role in the generation of systematic errors. Moreover, I will perform statistical tests like Fisher's exact test to validate that their occurrence is not by chance. Furthermore, I will investigate existing or new strategies to filter or avoid FPs and will also develop some new ones to provide more accurate results.

I believe that the new developments in the sequencing field can address existing problems and can improve the accuracy of data analysis. For example, third generation technologies (or single molecule sequencing), like PacBio RS and Oxford Nanopore MinION, provide much longer reads without errors due to dephasing or PCR amplification. These techniques allow for more accurate sequencing in repetitive or low complexity regions. This can solve the problem of misalignments in these regions, which is the major cause of SNP or Indel errors. Moreover, they can improve structural variant detection, which is currently having certain limitations like coverage bias, poor break point detection etc., due to the short read technologies. Recently, many studies have used these technologies to explore repetitive regions or to detect structural variants (McFarland et al., 2015; Pendleton et al., 2015; Ritz et al., 2014). These long reads can also be used for de novo assembly to build better consensus sequences which can avoid FPs due to errors or incompleteness of the human reference genome. However, to date these technologies are not as matured as second generation (e.g. Illumina) technologies. They have high raw read error rates and achieve only low throughput. Moreover, there is a need to develop new sophisticated data analysis algorithms, as these technologies are having different error profiles and different kinds of information in the generated sequencing data (Schadt et al., 2010). Further, the genome reference consortium continuously improves the reference genome sequence. The most recent build of the human reference genome (GRCh38) has less gaps and errors than the previous one (GRCh37), which also will increase the accuracy of variant detection. To give a résumé, lots of improvements in different directions are going on to make sequencing technologies and data analysis more reliable and manageable.

Bibliography

Abecasis, G. R., Auton, A., Brooks, L. D., DePristo, M. a, Durbin, R. M., Handsaker, R. E., … McVean, G. A. (2012). An integrated map of genetic variation from 1,092 human genomes. *Nature*, *491*(7422), 56–65. doi:10.1038/nature11632

Adzhubei, I. A., Schmidt, S., Peshkin, L., Ramensky, V. E., Gerasimova, A., Bork, P., … Sunyaev, S. R. (2010). A method and server for predicting damaging missense mutations. *Nature Methods*. doi:10.1038/nmeth0410-248

Aird, D., Ross, M. G., Chen, W.-S., Danielsson, M., Fennell, T., Russ, C., … Gnirke, A. (2011). Analyzing and minimizing PCR amplification bias in Illumina sequencing libraries. *Genome Biology*, *12*(2), R18. doi:10.1186/gb-2011-12-2-r18

Aiuti, A., Cattaneo, F., Galimberti, S., Benninghoff, U., Cassani, B., Callegaro, L., … Roncarolo, M.-G. (2009). *Gene therapy for immunodeficiency due to adenosine deaminase deficiency. The New England journal of medicine* (Vol. 360). doi:10.1056/NEJMoa0805817

Alkan, C., Kidd, J. M., Marques-Bonet, T., Aksay, G., Antonacci, F., Hormozdiari, F., … Eichler, E. E. (2009). Personalized copy number and segmental duplication maps using next-generation sequencing. *Nature Genetics*, *41*(10), 1061–1067. doi:10.1038/ng.437

Alkuraya, F. S. (2014). Genetics and genomic medicine in Israel. *Molecular Genetics & Genomic Medicine*, *2*(2), 85–94. doi:10.1002/mgg3.73

Allhoff, M., Schönhuth, A., Martin, M., Costa, I. G., Rahmann, S., & Marschall, T. (2013). Discovering motifs that induce sequencing errors. *BMC Bioinformatics*, *14 Suppl 5*(Suppl 5), S1. doi:10.1186/1471-2105-14-S5-S1

Altmüller, J., Budde, B. S., & Nürnberg, P. (2014). Enrichment of target sequences for nextgeneration sequencing applications in research and diagnostics. *Biological Chemistry*, *395*(2), 231–237. doi:10.1515/hsz-2013-0199

Antonio, M. D., Meo, P. D. O. De, Paoletti, D., Elmi, B., Pallocca, M., Sanna, N., … Castrignanò, T. (2013). WEP : a high-performance analysis pipeline for whole-exome data. *BMC Bioinformatics*, *14*(Suppl 7), S11. doi:10.1186/1471-2105-14-S7-S11

Arthur, J. W., Cheung, F. S. G., & Reichardt, J. K. V. (2015). Single Nucleotide Differences (SNDs) Continue to Contaminate the dbSNP Database With Consequences for Human Genomics and Health. *Human Mutation*, *36*(2), 196–199. doi:10.1002/humu.22735

Bao, R., Huang, L., Andrade, J., Tan, W., Kibbe, W. a, Jiang, H., & Feng, G. (2014). Review of Current Methods, Applications, and Data Management for the Bioinformatics Analysis of Whole Exome Sequencing. *Libertas Academica*, *13*, 67–82. doi:10.4137/CIN.S13779.Received

Belinky, F., Nativ, N., Stelzer, G., Zimmerman, S., Iny Stein, T., Safran, M., & Lancet, D. (2015). PathCards: multi-source consolidation of human biological pathways. *Database*, *2015*, bav006–bav006. doi:10.1093/database/bav006

Benjamini, Y., & Speed, T. P. (2012). Summarizing and correcting the GC content bias in high-throughput sequencing. *Nucleic Acids Research*, *40*(10), e72. doi:10.1093/nar/gks001

Bentley, D. R., Balasubramanian, S., Swerdlow, H. P., Smith, G. P., Milton, J., Brown, C. G., … Smith, A. J. (2008). Accurate whole human genome sequencing using reversible terminator chemistry. *Nature*, *456*(7218), 53–59. doi:10.1038/nature07517

Berglund, E. C., Kiialainen, A., & Syvänen, A.-C. (2011). Next-generation sequencing technologies and applications for human genetic history and forensics. *Investigative Genetics*, *2*(1), 23. doi:10.1186/2041-2223-2-23

Biesecker, L. G., & Green, R. C. (2014). Diagnostic Clinical Genome and Exome Sequencing. *New England Journal of Medicine*, *370*, 2418–2425. doi:10.1056/NEJMra1312543

Biesecker, L. G., Shianna, K. V, & Mullikin, J. C. (2011). Exome sequencing: the expert view. *Genome Biology*, *12*(9), 128. doi:10.1186/gb-2011-12-9-128

Bodi, K., Perera, a. G., Adams, P. S., Bintzler, D., Dewar, K., Grove, D. S., … Zianni, M. (2013). Comparison of commercially available target enrichment methods for next-generation sequencing. *Journal of Biomolecular Techniques*, *24*(2), 73–86. doi:10.7171/jbt.13-2402-002

Botstein, D., & Risch, N. (2003). Discovering genotypes underlying human phenotypes: past successes for mendelian disease, future approaches for complex disease. *Nature Genetics*, *33 Suppl*(march), 228–237. doi:10.1038/ng1090

Burrows, M., & D. J. Wheeler. (1994). *A block-sorting lossless data compression algorithm*. 124, Palo Alto, CA.

Buske, F. A., French, H. J., Smith, M. A., Clark, S. J., & Bauer, D. C. (2014). NGSANE: a lightweight production informatics framework for high-throughput data analysis. *Bioinformatics (Oxford, England)*, *30*(10), 1471–1472. doi:10.1093/bioinformatics/btu036

Cavazzana-Calvo, M., Payen, E., Negre, O., Wang, G., Hehir, K., Fusil, F., … Leboulch, P. Transfusion independence and HMGA2 activation after gene therapy of human β-thalassaemia. , 467 Nature 318–322 (2010). doi:10.1038/nature09328

Chen, Y.-C., Liu, T., Yu, C.-H., Chiang, T.-Y., & Hwang, C.-C. (2013). Effects of GC bias in next-generation-sequencing data on de novo genome assembly. *PloS One*, *8*(4), e62856. doi:10.1371/journal.pone.0062856

Chesi, A., Staahl, B. T., Jovičić, A., Couthouis, J., Fasolino, M., Raphael, A. R., … Gitler, A. D. (2013). Exome sequencing to identify de novo mutations in sporadic ALS trios. *Nature Neuroscience*, (May). doi:10.1038/nn.3412

Cheung, M. S., Down, T. a., Latorre, I., & Ahringer, J. (2011). Systematic bias in high-throughput sequencing data and its correction by BEADS. *Nucleic Acids Research*, *39*(15). doi:10.1093/nar/gkr425

Chilamakuri, C. S. R., Lorenz, S., Madoui, M.-A., Vodák, D., Sun, J., Hovig, E., … Meza-Zepeda, L. a. (2014). Performance comparison of four exome capture systems for deep sequencing. *BMC Genomics*, *15*, 449. doi:10.1186/1471-2164-15-449

Clark, M. J., Chen, R., Lam, H. Y. K., Karczewski, K. J., Chen, R., Euskirchen, G., … Snyder, M. (2011). Performance comparison of exome DNA sequencing technologies. *Nature Biotechnology*, *29*(10), 908–914. doi:10.1038/nbt.1975

Consugar, M. B., Navarro-Gomez, D., Place, E. M., Bujakowska, K. M., Sousa, M. E., Fonseca-Kelly, Z. D., … Pierce, E. a. (2014). Panel-based genetic diagnostic testing for inherited eye diseases is highly accurate and reproducible, and more sensitive for variant detection, than exome sequencing.

Genetics in Medicine, (October). doi:10.1038/gim.2014.172

Coonrod, E. M., Margraf, R. L., & Voelkerding, K. V. (2012). Translating exome sequencing from research to clinical diagnostics. *Clinical Chemistry and Laboratory Medicine : CCLM / FESCC*, *50*(7), 1161–8. doi:10.1515/cclm-2011-0841

Crick, F., & Watson, J. (1953). Molecular Structure of Nucleic Acids.

Crooks, G. E. (2004). WebLogo: A Sequence Logo Generator. *Genome Research*, *14*(6), 1188–1190. doi:10.1101/gr.849004

D'Antonio, M., D'Onorio De Meo, P., Paoletti, D., Elmi, B., Pallocca, M., Sanna, N., … Castrignanò, T. (2013). WEP: a high-performance analysis pipeline for whole-exome data. *BMC Bioinformatics*, *14 Suppl 7*(Suppl 7), S11. doi:10.1186/1471-2105-14-S7-S11

Dahl, F., Stenberg, J., Fredriksson, S., Welch, K., Zhang, M., Nilsson, M., … Ji, H. (2007). Multigene amplification and massively parallel sequencing for cancer mutation discovery. *Proceedings of the National Academy of Sciences of the United States of America*, *104*(22), 9387–92. doi:10.1073/pnas.0702165104

Danecek, P., Auton, A., Abecasis, G., Albers, C. A., Banks, E., DePristo, M. A., … Durbin, R. (2011). The variant call format and VCFtools. *Bioinformatics*, *27*(15), 2156–2158. doi:10.1093/bioinformatics/btr330

Davydov, E. V., Goode, D. L., Sirota, M., Cooper, G. M., Sidow, A., & Batzoglou, S. (2010). Identifying a high fraction of the human genome to be under selective constraint using GERP++. *PLoS Computational Biology*, *6*(12). doi:10.1371/journal.pcbi.1001025

De Leeneer, K., Hellemans, J., De Schrijver, J., Baetens, M., Poppe, B., Van Criekinge, W., … Claes, K. (2011). Massive parallel amplicon sequencing of the breast cancer genes BRCA1 and BRCA2: opportunities, challenges, and limitations. *Human Mutation*, *32*(3), 335–44. doi:10.1002/humu.21428

Dees, N. D., Zhang, Q., Kandoth, C., Wendl, M. C., Schierding, W., Koboldt, D. C., … Ding, L. (2012). MuSiC : Identifying mutational significance in cancer genomes MuSiC : Identifying mutational significance in cancer genomes. doi:10.1101/gr.134635.111

Del Fabbro, C., Scalabrin, S., Morgante, M., & Giorgi, F. M. (2013). An extensive evaluation of read trimming effects on Illumina NGS data analysis. *PloS One*, *8*(12), e85024. doi:10.1371/journal.pone.0085024

DePristo, M. A., Banks, E., Poplin, R., Garimella, K. V, Maguire, J. R., Hartl, C., … Daly, M. J. (2011). A framework for variation discovery and genotyping using next-generation DNA sequencing data. *Nature Genetics*, *43*(5), 491–498. doi:10.1038/ng.806

DeWan, A. T., Egan, K. B., Hellenbrand, K., Sorrentino, K., Pizzoferrato, N., Walsh, K. M., & Bracken, M. B. (2012). Whole-exome sequencing of a pedigree segregating asthma. *BMC Medical Genetics*. doi:10.1186/1471-2350-13-95

Directors, A. B. of. (2012). Points to consider in the clinical application of genomic sequencing. *Genetics in Medicine*. doi:10.1038/gim.2012.74

Dohm, J. C., Lottaz, C., Borodina, T., & Himmelbauer, H. (2008). Substantial biases in ultra-short read data sets from high-throughput DNA sequencing. *Nucleic Acids Research*, *36*(16), e105. doi:10.1093/nar/gkn425

Ehmke, N., Caliebe, A., Koenig, R., Kant, S. G., Stark, Z., Cormier-Daire, V., … Mundlos, S. (2014). Homozygous and Compound-Heterozygous Mutations in TGDS Cause Catel-Manzke Syndrome. *American Journal of Human Genetics*, *95*(6), 763–770. doi:10.1016/j.ajhg.2014.11.004

Ewing, B., & Green, P. (1998). Base-Calling of Automated Sequencer Traces Using Phred. II. Error Probabilities. *Genome Research*, *8*(3), 186–194. doi:10.1101/gr.8.3.186

Ewing, B., Hillier, L., Wendl, M. C., & Green, P. (1998). Base-Calling of Automated Sequencer Traces Using Phred. I. Accuracy Assessment. *Genome Res.*, *8*(3), 175–185. doi:10.1101/gr.8.3.175

Fang, H., Wu, Y., Narzisi, G., O'Rawe, J. A., Barrón, L. T. J., Rosenbaum, J., … Lyon, G. J. (2014). Reducing INDEL calling errors in whole genome and exome sequencing data. *Genome Medicine*, *6*(10), 89. doi:10.1186/s13073-014-0089-z

Fischer, M., Snajder, R., Pabinger, S., Dander, A., Schossig, A., Zschocke, J., … Stocker, G. (2012). SIMPLEX: cloud-enabled pipeline for the comprehensive analysis of exome sequencing data. *PloS One*, *7*(8), e41948. doi:10.1371/journal.pone.0041948

Flicek, P., Amode, M. R., Barrell, D., Beal, K., Billis, K., Brent, S., … Searle, S. M. J. (2014). Ensembl 2014. *Nucleic Acids Research*, *42*(D1). doi:10.1093/nar/gkt1196

Fromer, M., Moran, J. L., Chambert, K., Banks, E., Bergen, S. E., Ruderfer, D. M., … Purcell, S. M. (2012). Discovery and statistical genotyping of copy-number variation from whole-exome sequencing depth. *American Journal of Human Genetics*, *91*(4), 597–607. doi:10.1016/j.ajhg.2012.08.005

Fromer, M., Pocklington, A. J., Kavanagh, D. H., Williams, H. J., Dwyer, S., Gormley, P., … O'Donovan, M. C. (2014). De novo mutations in schizophrenia implicate synaptic networks. *Nature*. doi:10.1038/nature12929

Fuentes Fajardo, K. V., Adams, D., Mason, C. E., Sincan, M., Tifft, C., Toro, C., … Markello, T. (2012). Detecting false-positive signals in exome sequencing. *Human Mutation*, *33*(4), 609–613. doi:10.1002/humu.22033

Genovese, G., Handsaker, R. E., Li, H., Altemose, N., Lindgren, A. M., Chambert, K., … McCarroll, S. a. (2013). Using population admixture to help complete maps of the human genome. *Nature Genetics*, *45*(4), 406–14, 414e1–2. doi:10.1038/ng.2565

Giannoulatou, E., Park, S.-H., Humphreys, D. T., & Ho, J. W. (2014). Verification and validation of bioinformatics software without a gold standard: a case study of BWA and Bowtie. *BMC Bioinformatics*, *15*(Suppl 16), S15. doi:10.1186/1471-2105-15-S16-S15

Glöckle, N., Kohl, S., Mohr, J., Scheurenbrand, T., Sprecher, A., Weisschuh, N., … Neidhardt, J. (2013). Panel-based next generation sequencing as a reliable and efficient technique to detect mutations in unselected patients with retinal dystrophies. *European Journal of Human Genetics : EJHG*, (November 2012), 1–6. doi:10.1038/ejhg.2013.72

Gnirke, A., Melnikov, A., Maguire, J., Rogov, P., LeProust, E. M., Brockman, W., … Nusbaum, C. (2009). Solution Hybrid Selection with Ultra-long Oligonucleotides for Massively Parallel Targeted Sequencing. *Nature Biotechnology*, *27*(2), 182–189. doi:10.1038/nbt.1523

Gudbjartsson, D. F., Jonasson, K., Frigge, M. L., & Kong, A. (2000, May). Allegro, a new computer program for multipoint linkage analysis. *Nature Genetics*. UNITED STATES. doi:10.1038/75514

Hach, F., Hormozdiari, F., Alkan, C., Hormozdiari, F., Birol, I., Eichler, E. E., & Sahinalp, S. C. (2010).

mrsFAST: a cache-oblivious algorithm for short-read mapping. *Nature Methods*. doi:10.1038/nmeth0810-576

Harismendy, O., Schwab, R. B., Bao, L., Olson, J., Rozenzhak, S., Kotsopoulos, S. K., … Frazer, K. a. (2011). Detection of low prevalence somatic mutations in solid tumors with ultra-deep targeted sequencing. *Genome Biology*, *12*(12), R124. doi:10.1186/gb-2011-12-12-r124

Hatem, A., Bozdağ, D., & Çatalyürek, Ü. V. (2011). Benchmarking short sequence mapping tools. In *Proceedings - 2011 IEEE International Conference on Bioinformatics and Biomedicine, BIBM 2011* (pp. 109–113). doi:10.1109/BIBM.2011.83

Highnam, G., Wang, J. J., Kusler, D., Zook, J., Vijayan, V., Leibovich, N., & Mittelman, D. (2015). An analytical framework for optimizing variant discovery from personal genomes. *Nature Communications*, *6*, 6275. doi:10.1038/ncomms7275

Hofacker, I. L. (2009). RNA Secondary Structure Analysis Using the Vienna RNA Package. *Current Protocols in Bioinformatics*, (October). doi:10.1002/0471250953.bi1202s26

Holley, R. W., Apgar, J., Everett, G. A., Madison, J. T., Marquisee, M., Merrill, S. H., … Zamir, A. (1965). Structure of a Ribonucleic Acid. *Science*, *147*(3664), 1462–1465. doi:10.1126/science.147.3664.1462

Hormozdiari, F., Alkan, C., Eichler, E. E., & Sahinalp, S. C. (2009). Combinatorial algorithms for structural variation detection in high-throughput sequenced genomes. *Genome Research*, *19*(7), 1270–1278. doi:10.1101/gr.088633.108

Hutchison, C. a. (2007). DNA sequencing: Bench to bedside and beyond. *Nucleic Acids Research*, *35*(18), 6227–6237. doi:10.1093/nar/gkm688

Hwang, K.-B., Lee, I.-H., Park, J.-H., Hambuch, T., Choe, Y., Kim, M., … Kong, S. W. (2014). Reducing False-Positive Incidental Findings with Ensemble Genotyping and Logistic Regression Based Variant Filtering Methods. *Human Mutation*. doi:10.1002/humu.22587

Hyman, E. D. (1988). A new method of sequencing DNA. *Analytical Biochemistry*, *174*(2), 423–436. doi:10.1016/0003-2697(88)90041-3

Jette, M., & Grondona, M. (2003). SLURM: Simple Linux Utility for Resource Management. In *Proc. of ClusterWorld Conference and Expo*. San Jose, California.

Jiang, T., Tan, M., Tan, L., & Yu, J. (2013). Application of next-generation sequencing technologies in Neurology, *2*(12), 1–11. doi:10.3978/j.issn.2306-9732.2013.03.02

Kawalia, A., Motameny, S., Wonczak, S., Thiele, H., Nieroda, L., Jabbari, K., … Nürnberg, P. (2015). Leveraging the Power of High Performance Computing for Next Generation Sequencing Data Analysis: Tricks and Twists from a High Throughput Exome Workflow. *Plos One*, *10*(5), e0126321. doi:10.1371/journal.pone.0126321

Kelly, B. J., Fitch, J. R., Hu, Y., Corsmeier, D. J., Zhong, H., Wetzel, A. N., … White, P. (2015). Churchill: an ultra-fast, deterministic, highly scalable and balanced parallelization strategy for the discovery of human genetic variation in clinical and population-scale genomics. *Genome Biology*, *16*, 1–14. doi:10.1186/s13059-014-0577-x

Kiezun, A., Garimella, K., Do, R., Stitziel, N. O., Benjamin, M., Mclaren, P. J., … Magnusson, P. (2013). Exome sequencing and the genetic basis of complex traits. *NIH Public Access*, *44*(6), 623–630. doi:10.1038/ng.2303.Exome

Kiialainen, A., Karlberg, O., Ahlford, A., Sigurdsson, S., Lindblad-Toh, K., & Syvänen, A. C. (2011). Performance of microarray and liquid based capture methods for target enrichment for massively parallel sequencing and SNP discovery. *PLoS ONE*, *6*(2). doi:10.1371/journal.pone.0016486

Kircher, M. (2011). Understanding and improving high-throughput sequencing data production and analysis. *Doctor*, (September), 216.

Kircher, M., Heyn, P., & Kelso, J. (2011). Addressing challenges in the production and analysis of illumina sequencing data. *BMC Genomics*, *12*(1), 382. doi:10.1186/1471-2164-12-382

Kircher, M., Stenzel, U., & Kelso, J. (2009). Improved base calling for the Illumina Genome Analyzer using machine learning strategies. *Genome Biology*, *10*(8), R83. doi:10.1186/gb-2009-10-8-r83

Klambauer, G., Schwarzbauer, K., Mayr, A., Clevert, D.-A. A., Mitterecker, A., Bodenhofer, U., & Hochreiter, S. (2012). Cn.MOPS: Mixture of Poissons for discovering copy number variations in next-generation sequencing data with a low false discovery rate. *Nucleic Acids Research*, *40*(9), e69. doi:10.1093/nar/gks003

Kozarewa, I., Ning, Z., Quail, M. a, Sanders, M. J., Berriman, M., Turner, D. J., & America, N. (2009). Amplification-free Illumina sequencing-library preparation facilitates improved mapping and assembly of (G+C)-biased genomes. *Nature Methods*, *6*(4), 291–295. doi:10.1038/nmeth.1311

Krumm, N., Sudmant, P. H., Ko, A., O'Roak, B. J., Malig, M., Coe, B. P., … Project, S. (2012). Copy number variation detection and genotyping from exome sequence data. *Genome Research*, *22*(8), 1525–32. doi:10.1101/gr.138115.112

Ku, C. S., Cooper, D. N., E. Ziogas, D., Halkia, E., Tzaphlidou, M., & Roukos, D. H. (2013). Research and clinical applications of cancer genome sequencing. doi:10.1097/GCO.0b013e32835af17c

Kumar, P., Henikoff, S., & Ng, P. C. (2009). Predicting the effects of coding non-synonymous variants on protein function using the SIFT algorithm. *Nature Protocols*, *4*(7), 1073–1081. doi:10.1038/nprot.2009.86

Kuruppumullage Don, P., Ananda, G., Chiaromonte, F., & Makova, K. D. (2013). Segmenting the human genome based on states of neutral genetic divergence. *Proceedings of the National Academy of Sciences of the United States of America*, *110*(36), 14699–704. doi:10.1073/pnas.1221792110

Laduca, H., Stuenkel, a J., Dolinsky, J. S., Keiles, S., Tandy, S., Pesaran, T., … Chao, E. (2014). Utilization of multigene panels in hereditary cancer predisposition testing: analysis of more than 2,000 patients. *Genetics in Medicine : Official Journal of the American College of Medical Genetics*, (April), 1–8. doi:10.1038/gim.2014.40

Lal, D., Reinthaler, E. M., Schubert, J., Muhle, H., Riesch, E., Kluger, G., … Neubauer, B. A. (2014). DEPDC5 mutations in genetic focal epilepsies of childhood. *Annals of Neurology*, *75*(5), 788–792. doi:10.1002/ana.24127

Lam, H. Y. K., Pan, C., Clark, M. J., Lacroute, P., Chen, R., Haraksingh, R., … Snyder, M. (2012). Detecting and annotating genetic variations using the HugeSeq pipeline. *Nature Biotechnology*, *30*(3), 226–9. doi:10.1038/nbt.2134

Lan, J. H., Yin, Y., Reed, E. F., Moua, K., Thomas, K., & Zhang, Q. (2015). Impact of three Illumina library construction methods on GC bias and HLA genotype calling. *Human Immunology*, *76*(2-3), 166–175. doi:10.1016/j.humimm.2014.12.016

Lander, E. S., Linton, L. M., Birren, B., Nusbaum, C., Zody, M. C., Baldwin, J., … Szustakowki, J. (2001). Initial sequencing and analysis of the human genome. *Nature*, *409*(6822), 860–921. doi:10.1038/35057062

Langmead, B., Trapnell, C., Pop, M., & Salzberg, S. L. (2009). Ultrafast and memory-efficient alignment of short DNA sequences to the human genome. *Genome Biology*, *10*(3), R25. doi:10.1186/gb-2009-10-3-r25

Lappalainen, I., Lopez, J., Skipper, L., Hefferon, T., Spalding, J. D., Garner, J., … Church, D. M. (2013). DbVar and DGVa: Public archives for genomic structural variation. *Nucleic Acids Research*, *41*(D1). doi:10.1093/nar/gks1213

Leek, J. T., Scharpf, R. B., Bravo, H. C., Simcha, D., Langmead, B., Johnson, W. E., … Irizarry, R. a. (2010). Tackling the widespread and critical impact of batch effects in high-throughput data. *Nature Reviews. Genetics*, *11*(10), 733–739. doi:10.1038/nrg2825

Li, H. (2011). A statistical framework for SNP calling, mutation discovery, association mapping and population genetical parameter estimation from sequencing data. *Bioinformatics (Oxford, England)*, *27*(21), 2987–93. doi:10.1093/bioinformatics/btr509

Li, H. (2014). Towards Better Understanding of Artifacts in Variant Calling from High-Coverage Samples. *Bioinformatics (Oxford, England)*, 1–8. doi:10.1093/bioinformatics/btu356

Li, H., & Durbin, R. (2009). Fast and accurate short read alignment with Burrows-Wheeler transform. *Bioinformatics*, *25*(14), 1754–1760. doi:10.1093/bioinformatics/btp324

Li, H., & Durbin, R. (2010). Fast and accurate long-read alignment with Burrows-Wheeler transform. *Bioinformatics*, *26*(5), 589–595. doi:10.1093/bioinformatics/btp698

Li, H., Handsaker, B., Wysoker, A., Fennell, T., Ruan, J., Homer, N., … Durbin, R. (2009). The Sequence Alignment/Map format and SAMtools. *Bioinformatics*, *25*(16), 2078–2079. doi:10.1093/bioinformatics/btp352

Li, H., & Homer, N. (2010). A survey of sequence alignment algorithms for next-generation sequencing. *Briefings in Bioinformatics*, *11*(5), 473–83. doi:10.1093/bib/bbq015

Li, H., Ruan, J., & Durbin, R. (2008). Mapping short DNA sequencing reads and calling variants using mapping quality scores. *Genome Research*, *18*(11), 1851–8. doi:10.1101/gr.078212.108

Li, R., Yu, C., Li, Y., Lam, T. W., Yiu, S. M., Kristiansen, K., & Wang, J. (2009). SOAP2: An improved ultrafast tool for short read alignment. *Bioinformatics*, *25*(15), 1966–1967. doi:10.1093/bioinformatics/btp336

Linderman, M. D., Brandt, T., Edelmann, L., Jabado, O., Kasai, Y., Kornreich, R., … Schadt, E. E. (2014). Analytical validation of whole exome and whole genome sequencing for clinical applications. *BMC Medical Genomics*, *7*(1), 20. doi:10.1186/1755-8794-7-20

Liu, Q., Guo, Y., Li, J., Long, J., Zhang, B., & Shyr, Y. (2012). Steps to ensure accuracy in genotype and SNP calling from Illumina sequencing data. *BMC Genomics*, *13 Suppl 8*(Suppl 8), S8. doi:10.1186/1471-2164-13-S8-S8

Lorenz, R., Bernhart, S. H., Höner Zu Siederdissen, C., Tafer, H., Flamm, C., Stadler, P. F., & Hofacker, I. L. (2011). ViennaRNA Package 2.0. *Algorithms for Molecular Biology : AMB*, *6*, 26. doi:10.1186/1748-7188-6-26

Lucas Lledó, J. I., & Cáceres, M. (2013). On the power and the systematic biases of the detection of chromosomal inversions by paired-end genome sequencing. *PloS One*, *8*(4), e61292. doi:10.1371/journal.pone.0061292

Lynch, A. I., Eckfeldt, J. H., Davis, B. R., Ford, C. E., Boerwinkle, E., Leiendecker-Foster, C., & Arnett, D. K. (2012). Gene panels to help identify subgroups at high and low risk of coronary heart disease among those randomized to antihypertensive treatment. *Pharmacogenetics and Genomics*, 1. doi:10.1097/FPC.0b013e3283516ff8

MacManes, M. D. (2013). *On the optimal trimming of high-throughput mRNAseq data*. *bioRxiv*. doi:10.1101/000422

Majewski, J., Schwartzentruber, J., Lalonde, E., Montpetit, A., & Jabado, N. (2011). What can exome sequencing do for you? *Journal of Medical Genetics*, *48*, 580–589. doi:10.1136/jmedgenet-2011-100223

Mamanova, L., Coffey, A. J., Scott, C. E., Kozarewa, I., Turner, E. H., Kumar, A., … Turner, D. J. (2010). Target-enrichment strategies for next- generation sequencing. *Nature Methods*, *7*(2), 111–118. doi:10.1038/NMETH.1419

Mardis, E. R. (2008). Next-generation DNA sequencing methods. *Annual Review of Genomics and Human Genetics*, *9*, 387–402. doi:10.1146/annurev.genom.9.081307.164359

Mardis, E. R. (2013). Next-generation sequencing platforms. *Annual Review of Analytical Chemistry (Palo Alto, Calif.)*, *6*, 287–303. doi:10.1146/annurev-anchem-062012-092628

Margulies, M., Egholm, M., Altman, W. E., Attiya, S., Bader, J. S., Bemben, L. A., … Rothberg, J. M. (2005). Genome sequencing in microfabricated high-density picolitre reactors. *Nature*, *437*(7057), 376–380. doi:10.1038/nature04726

Martin, M. (2011). Cutadapt removes adapter sequences from high-throughput sequencing reads. *EMBnet.journal*, *17*(1), 10. doi:10.14806/ej.17.1.200

McFarland, K. N., Liu, J., Landrian, I., Godiska, R., Shanker, S., Yu, F., … Ashizawa, T. (2015). SMRT Sequencing of Long Tandem Nucleotide Repeats in SCA10 Reveals Unique Insight of Repeat Expansion Structure. *PLoS ONE*, *10*(8), e0135906. doi:10.1371/journal.pone.0135906

McKenna, A., Hanna, M., Banks, E., Sivachenko, A., Cibulskis, K., Kernytsky, A., … DePristo, M. a. (2010, September). The Genome Analysis Toolkit: a MapReduce framework for analyzing next-generation DNA sequencing data. *Genome Research*. doi:10.1101/gr.107524.110

Meacham, F., Boffelli, D., Dhahbi, J., Martin, D. I., Singer, M., & Pachter, L. (2011). Identification and correction of systematic error in high-throughput sequence data. *BMC Bioinformatics*, *12*(1), 451. doi:10.1186/1471-2105-12-451

Medvedev, P., Stanciu, M., & Brudno, M. (2009). computational methods for discovering structural variation with next-generation sequencing, *6*(11), 13–20. doi:10.1038/NMEtH.1374

Meldrum, C., Doyle, M. a, & Tothill, R. W. (2011). Next-generation sequencing for cancer diagnostics: a practical perspective. *The Clinical Biochemist. Reviews / Australian Association of Clinical Biochemists*, *32*(4), 177–95.

Mertes, F., ElSharawy, A., Sauer, S., van Helvoort, J. M. L. M., van der Zaag, P. J., Franke, A., … Brookes, A. J. (2011). Targeted enrichment of genomic DNA regions for next-generation sequencing. *Briefings in*

Functional Genomics, *10*(6), 374–386. doi:10.1093/bfgp/elr033

Metzker, M. L. (2010). Sequencing technologies - the next generation. *Nature Reviews. Genetics*, *11*(1), 31–46. doi:10.1038/nrg2626

Mills, R. E., Luttig, C. T., Larkins, C. E., Beauchamp, A., Tsui, C., Pittard, W. S., & Devine, S. E. (2006). An initial map of insertion and deletion (INDEL) variation in the human genome. *Genome Research*, *16*(9), 1182–90. doi:10.1101/gr.4565806

Mutarelli, M., Marwah, V., Rispoli, R., Carrella, D., Dharmalingam, G., Oliva, G., & di Bernardo, D. (2014). A community-based resource for automatic exome variant-calling and annotation in Mendelian disorders. *BMC Genomics*, *15*(Suppl 3), S5. doi:10.1186/1471-2164-15-S3-S5

Nakamura, K., Oshima, T., Morimoto, T., Ikeda, S., Yoshikawa, H., Shiwa, Y., … Kanaya, S. (2011). Sequence-specific error profile of Illumina sequencers. *Nucleic Acids Research*, *39*(13), e90. doi:10.1093/nar/gkr344

Need, A. C., Shashi, V., Hitomi, Y., Schoch, K., Shianna, K. V, McDonald, M. T., … Goldstein, D. B. (2012). Clinical application of exome sequencing in undiagnosed genetic conditions. *Journal of Medical Genetics*, *49*(6), 353–61. doi:10.1136/jmedgenet-2012-100819

Needleman, S. B., & 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*(3), 443–453. doi:10.1016/0022-2836(70)90057-4

Ng, S. B., Buckingham, K. J., Lee, C., Bigham, A. W., Tabor, H. K., Dent, K. M., … Bamshad, M. J. (2010). Exome sequencing identifies the cause of a mendelian disorder. *Nature Genetics*, *42*(1), 30–35. doi:10.1038/ng.499

Ng, S. B., Turner, E. H., Robertson, P. D., Flygare, S. D., Bigham, A. W., Lee, C., … Shendure, J. (2009). Targeted capture and massively parallel sequencing of 12 human exomes. *Nature*, *461*(7261), 272–6. doi:10.1038/nature08250

Nielsen, R., Paul, J. S., Albrechtsen, A., & Song, Y. S. (2011). Genotype and SNP calling from next-generation sequencing data. *Nature Reviews. Genetics*, *12*(6), 443–451. doi:10.1038/nrg2986

Ning, B., Su, Z., Mei, N., Hong, H., Deng, H., Shi, L., … Tolleson, W. H. (2014). Toxicogenomics and Cancer Susceptibility: Advances with Next-Generation Sequencing. *Journal of Environmental Science and Health, Part C*, *32*(2), 121–158. doi:10.1080/10590501.2014.907460

Nix, D. A., Courdy, S. J., & Boucher, K. M. (2008). Empirical methods for controlling false positives and estimating confidence in ChIP-Seq peaks. *BMC Bioinformatics*, *9*, 523. doi:10.1186/1471-2105-9-523

O'Rawe, J., Guangqing, S., Wang, W., Hu, J., Bodily, P., Tian, L., … Wu, Y. (2013). Low concordance of multiple variant-calling pipelines: practical implications for exome and genome sequencing. *Genome Medicine*, *5*(3), 28. doi:10.1186/gm432

Ott, M. G., Schmidt, M., Schwarzwaelder, K., Stein, S., Siler, U., Koehl, U., … Grez, M. (2006). *Correction of X-linked chronic granulomatous disease by gene therapy, augmented by insertional activation of MDS1-EVI1, PRDM16 or SETBP1. Nature medicine* (Vol. 12). doi:10.1038/nm1393

Oyola, S. O., Otto, T. D., Gu, Y., Maslen, G., Manske, M., Campino, S., … Quail, M. A. (2012). Optimizing illumina next-generation sequencing library preparation for extremely at-biased genomes. *BMC Genomics*. doi:10.1186/1471-2164-13-1

Pabinger, S., Dander, A., Fischer, M., Snajder, R., Sperk, M., Efremova, M., … Trajanoski, Z. (2013). A survey of tools for variant analysis of next-generation genome sequencing data. *Briefings in Bioinformatics*, *15*(2), 256–78. doi:10.1093/bib/bbs086

Paz-Filho, G., Boguszewski, M., Mastronardi, C., Patel, H., Johar, A., Chuah, A., … Licinio, J. (2014). Whole Exome Sequencing of Extreme Morbid Obesity Patients: Translational Implications for Obesity and Related Disorders. *Genes*, *5*(3), 709–725. doi:10.3390/genes5030709

Pendleton, M., Sebra, R., Pang, A. W. C., Ummat, A., Franzen, O., Rausch, T., … Bashir, A. (2015). Assembly and diploid architecture of an individual human genome via single-molecule technologies. *Nature Methods*, *12*(8), 780–786. doi:10.1038/nmeth.3454

Plagnol, V., Curtis, J., Epstein, M., Mok, K. Y., Stebbings, E., Grigoriadou, S., … Nejentsev, S. (2012). A robust model for read count data in exome sequencing experiments and implications for copy number variant calling. *Bioinformatics*, *28*(21), 2747–2754. doi:10.1093/bioinformatics/bts526

Pleasance, E. D., Cheetham, R. K., Stephens, P. J., McBride, D. J., Humphray, S. J., Greenman, C. D., … Stratton, M. R. (2010). A comprehensive catalogue of somatic mutations from a human cancer genome. *Nature*, *463*(7278), 191–196. doi:10.1038/nature08658

Puckelwartz, M. J., Pesce, L. L., Nelakuditi, V., Dellefave-Castillo, L., Golbus, J. R., Day, S. M., … McNally, E. M. (2014). Supercomputing for the parallelization of whole genome analysis. *Bioinformatics (Oxford, England)*, *30*(11), 1–7. doi:10.1093/bioinformatics/btu071

Quail, M., Smith, M. E., Coupland, P., Otto, T. D., Harris, S. R., Connor, T. R., … Gu, Y. (2012). A tale of three next generation sequencing platforms: comparison of Ion torrent, pacific biosciences and illumina MiSeq sequencers. *BMC Genomics*, *13*(1), 341. doi:10.1186/1471-2164-13-341

Rabbani, B., Mahdieh, N., Hosomichi, K., Nakaoka, H., & Inoue, I. (2012). Next-generation sequencing: impact of exome sequencing in characterizing Mendelian disorders. *Journal of Human Genetics*, *57*(10), 621–632. doi:10.1038/jhg.2012.91

Rabbani, B., Tekin, M., & Mahdieh, N. (2014). The promise of whole-exome sequencing in medical genetics. *Journal of Human Genetics*, *59*(1), 5–15. doi:10.1038/jhg.2013.114

Radó-Trilla, N., & Albà, M. M. (2012). Dissecting the role of low-complexity regions in the evolution of vertebrate proteins. *BMC Evolutionary Biology*, *12*(1), 155. doi:10.1186/1471-2148-12-155

Ramu, A., Noordam, M. J., Schwartz, R. S., Wuster, A., Hurles, M. E., Cartwright, R. a, & Conrad, D. F. (2013). DeNovoGear: de novo indel and point mutation discovery and phasing. *Nature Methods*, *10*(10), 985–7. doi:10.1038/nmeth.2611

Rattei, T., Arnold, R., Tischler, P., Lindner, D., Stümpflen, V., & Mewes, H. W. (2006). SIMAP: the similarity matrix of proteins. *Nucleic Acids Research*, *34*(Database issue), D252–D256. doi:10.1093/nar/gkj106

Rattei, T., Tischler, P., Götz, S., Jehl, M.-A., Hoser, J., Arnold, R., … Mewes, H.-W. (2010). SIMAP—a comprehensive database of pre-calculated protein sequence similarities, domains, annotations and clusters. *Nucleic Acids Research*, *38*(Database issue), D223–D226. doi:10.1093/nar/gkp949

Rehm, H. L. (2013). Disease-targeted sequencing: a cornerstone in the clinic. *Nature Reviews. Genetics*, *14*(April), 295–300. doi:10.1038/nrg3463

Reumers, J., De Rijk, P., Zhao, H., Liekens, A., Smeets, D., Cleary, J., … Del-Favero, J. (2011). Optimized filtering reduces the error rate in detecting genomic variants by short-read sequencing. *Nature*

Biotechnology, *30*(1), 61–68. doi:10.1038/nbt.2053

Rimmer, A., Phan, H., Mathieson, I., Iqbal, Z., Twigg, S. R. F., Wilkie, A. O. M., … Lunter, G. (2014). Integrating mapping-, assembly- and haplotype-based approaches for calling variants in clinical sequencing applications. *Nature Genetics*, *46*(November 2013), 1–9. doi:10.1038/ng.3036

Ritz, A., Bashir, A., Sindi, S., Hsu, D., Hajirasouliha, I., & Raphael, B. J. (2014). Characterization of structural variants with single molecule and hybrid sequencing approaches. *Bioinformatics*, *30*(24), 3458–3466. doi:10.1093/bioinformatics/btu714

Robinson, J. T., Thorvaldsdóttir, H., Winckler, W., Guttman, M., Lander, E. S., Getz, G., & Mesirov, J. P. (2011). Integrative genomics viewer. *Nature Biotechnology*, *29*(1), 24–26. doi:10.1038/nbt.1754

Roe, B. A. (2004). Shotgun library construction for DNA sequencing. *Methods in Molecular Biology (Clifton, N.J.)*, *255*, 171–187. doi:10.1385/1-59259-752-1:171

Ross, M. G., Russ, C., Costello, M., Hollinger, A., Lennon, N. J., Hegarty, R., … Jaffe, D. B. (2013). Characterizing and measuring bias in sequence data. *Genome Biology*, *14*(5), R51. doi:10.1186/gb-2013-14-5-r51

Ruffalo, M., Laframboise, T., & Koyutürk, M. (2011). Comparative analysis of algorithms for next-generation sequencing read alignment. *Bioinformatics*, *27*(20), 2790–2796. doi:10.1093/bioinformatics/btr477

Salomon, D. R., & Ordoukhanian, P. (2015). Library construction for next-generation sequencing: Overviews and challenges, *56*(2). doi:10.2144/000114133.Library

Sanger, F. (1949). The terminal peptides of insulin. *Biochemical Journal*, *45*(5), 563–574. Retrieved from http://www.ncbi.nlm.nih.gov/pmc/articles/PMC1275055/

Sanger, F., Nicklen, S., & Coulson, a R. (1977). DNA sequencing with chain-terminating inhibitors. *Proceedings of the National Academy of Sciences of the United States of America*, *74*(12), 5463–5467. doi:10.1073/pnas.74.12.5463

Sanger, F., & Tuppy, H. (1951). The amino-acid sequence in the phenylalanyl chain of insulin. 2. The investigation of peptides from enzymic hydrolysates. *Biochemical Journal*, *49*(4), 481–490. Retrieved from http://www.ncbi.nlm.nih.gov/pmc/articles/PMC1197536/

Schadt, E. E., Turner, S., & Kasarskis, A. (2010). A window into third-generation sequencing. *Human Molecular Genetics*, *19*(R2), 227–240. doi:10.1093/hmg/ddq416

Schmieder, R., & Edwards, R. (2011). Quality control and preprocessing of metagenomic datasets. *Bioinformatics*, *27*(6), 863–864. doi:10.1093/bioinformatics/btr026

Schneider, T. D., & Stephens, R. M. (1990). Sequence logos: a new way to display consensus sequences. *Nucleic Acids Research*, *18*(20), 6097–6100.

Schubert, J., Siekierska, A., Langlois, M., May, P., Huneau, C., Becker, F., … Lerche, H. (2014). Mutations in STX1B, encoding a presynaptic protein, cause fever-associated epilepsy syndromes. *Nature Genetics*, *2014*(November). doi:10.1038/ng.3130

Sherry, S. T., Ward, M.-H., Kholodov, M., Baker, J., Phan, L., Smigielski, E. M., & Sirotkin, K. (2001). dbSNP: the NCBI database of genetic variation. *Nucleic Acids Research*, *29*(1), 308–311. Retrieved from http://www.ncbi.nlm.nih.gov/pmc/articles/PMC29783/

Shim, Y. J., Kim, J. E., Hwang, S.-K., Choi, B. S., Choi, B. H., Cho, E.-M., … Ko, C. W. (2015). Identification of Candidate Gene Variants in Korean MODY Families by Whole-Exome Sequencing. *Hormone Research in Paediatrics*. doi:10.1159/000368657

Sie, A. S., Prins, J. B., van Zelst-Stams, W. A. G., Veltman, J. A., Feenstra, I., & Hoogerbrugge, N. (2014). Patient experiences with gene panels based on exome sequencing in clinical diagnostics: High acceptance and low distress. *Clinical Genetics*. doi:10.1111/cge.12433

Sikkema-Raddatz, B., Johansson, L. F., de Boer, E. N., Almomani, R., Boven, L. G., van den Berg, M. P., … van Tintelen, J. P. (2013). Targeted Next-Generation Sequencing can Replace Sanger Sequencing in Clinical Diagnostics. *Human Mutation*, 1–30. doi:10.1002/humu.22332

Sims, D., Sudbery, I., Ilott, N. E., Heger, A., & Ponting, C. P. (2014). Sequencing depth and coverage: key considerations in genomic analyses. *Nature Reviews Genetics*, *15*(2), 121–32. doi:10.1038/nrg3642

Smith, L. M., Sanders, J. Z., Kaiser, R. J., Hughes, P., Dodd, C., Connell, C. R., … Hood, L. E. (1986). Fluorescence detection in automated DNA sequence analysis. *Nature*, *321*(6071), 674–679. doi:10.1038/321674a0

Smith, T. F., & Waterman, M. S. (1981). Identification of common molecular subsequences. *Journal of Molecular Biology*, *147*(1), 195–197. doi:10.1016/0022-2836(81)90087-5

Stenson, P. D., Mort, M., Ball, E. V., Shaw, K., Phillips, A. D., & Cooper, D. N. (2014). The Human Gene Mutation Database: Building a comprehensive mutation repository for clinical and molecular genetics, diagnostic testing and personalized genomic medicine. *Human Genetics*. doi:10.1007/s00439-013-1358-4

Synofzik, M., Haack, T. B., Kopajtich, R., Gorza, M., Rapaport, D., Greiner, M., … Prokisch, H. (2014). Absence of BiP Co-chaperone DNAJC3 Causes Diabetes Mellitus and Multisystemic Neurodegeneration. *The American Journal of Human Genetics*, *95*(6), 689–697. doi:10.1016/j.ajhg.2014.10.013

Taub, M. a, Corrada Bravo, H., & Irizarry, R. a. (2010). Overcoming bias and systematic errors in next generation sequencing data. *Genome Medicine*, *2*(12), 87. doi:10.1186/gm208

Thaker, V. V., Esteves, K. M., Towne, M. C., Brownstein, C. A., James, P. M., Crowley, L., … Agrawal, P. B. (2015). Whole Exome Sequencing Identifies RAI1 Mutation in a Morbidly Obese Child Diagnosed with ROHHAD Syndrome. *The Journal of Clinical Endocrinology & Metabolism*, jc.2014–4215. doi:10.1210/jc.2014-4215

The 1000 Genomes Project Consortium. (2010). A map of human genome variation from population-scale sequencing. *Nature*, *467*(7319), 1061–73. doi:10.1038/nature09534

Treangen, T. J., & Salzberg, S. L. (2012). Repetitive DNA and next-generation sequencing: computational challenges and solutions. *Nature Reviews Genetics*, *13*(November 2011), 36–46. doi:10.1038/nrg3164

Trubetskoy, V., Rodriguez, A., Dave, U., Campbell, N., Crawford, E. L., Cook, E. H., … Davis, L. K. (2014). Consensus Genotyper for Exome Sequencing (CGES): improving the quality of exome variant genotypes. *Bioinformatics (Oxford, England)*, 1–7. doi:10.1093/bioinformatics/btu591

Tsongalis, G. J., Peterson, J. D., De Abreu, F. B., Tunkey, C. D., Gallagher, T. L., Strausbaugh, L. D., … Amos, C. I. (2014). Routine use of the Ion Torrent AmpliSeq^{TM} Cancer Hotspot Panel for identification of clinically actionable somatic mutations. *Clinical Chemistry and Laboratory Medicine*, *52*(5), 707–714. doi:10.1515/cclm-2013-0883

Turcatti, G., Romieu, A., Fedurco, M., & Tairi, A. P. (2008). A new class of cleavable fluorescent nucleotides: Synthesis and optimization as reversible terminators for DNA sequencing by synthesis. *Nucleic Acids Research*, *36*(4). doi:10.1093/nar/gkn021

Van der Auwera, G. A., Carneiro, M. O., Hartl, C., Poplin, R., Del Angel, G., Levy-Moonshine, A., … DePristo, M. A. (2013). From fastQ data to high-confidence variant calls: The genome analysis toolkit best practices pipeline. *Current Protocols in Bioinformatics*, *11*(SUPL.43), 11.10.1–11.10.33. doi:10.1002/0471250953.bi1110s43

Veltman, J. a, & Brunner, H. G. (2012). De novo mutations in human genetic disease. *Nature Reviews. Genetics*, *13*(8), 565–75. doi:10.1038/nrg3241

Venter, J. C., Adams, M. D., Myers, E. W., Li, P. W., Mural, R. J., Sutton, G. G., … Zhu, X. (2001). The sequence of the human genome. *Science (New York, N.Y.)*, *291*(5507), 1304–1351. doi:10.1126/science.1058040

Vissers, L. E. L. M., de Ligt, J., Gilissen, C., Janssen, I., Steehouwer, M., de Vries, P., … Veltman, J. a. (2010). A de novo paradigm for mental retardation. *Nature Genetics*, *42*(12), 1109–12. doi:10.1038/ng.712

Watson, I. R., Takahashi, K., Futreal, P. A., & Chin, L. (2013). Emerging patterns of somatic mutations in cancer. *Nature Reviews Genetics*, *14*(10), 703–718. doi:10.1038/nrg3539

Williams, R., Peisajovich, S. G., Miller, O. J., Magdassi, S., Tawfik, D. S., & Griffiths, A. D. (2006). Amplification of complex gene libraries by emulsion PCR. *Nature Methods*, *3*(7), 545–550. doi:10.1038/nmeth896

Wooderchak-Donahue, W. L., O'Fallon, B., Furtado, L. V, Durtschi, J. D., Plant, P., Ridge, P. G., … Bayrak-Toydemir, P. (2012). A direct comparison of next generation sequencing enrichment methods using an aortopathy gene panel- clinical diagnostics perspective. *BMC Medical Genomics*, *5*(1), 50. doi:10.1186/1755-8794-5-50

Yadav, N. K., Shukla, P., Omer, A., Pareek, S., & Singh, R. K. (2014). Next Generation Sequencing: Potential and Application in Drug Discovery. *The Scientific World Journal*, *2014*, 802437. doi:10.1155/2014/802437

Yang, X., Chockalingam, S. P., & Aluru, S. (2013). A survey of error-correction methods for next-generation sequencing. *Briefings in Bioinformatics*, *14*(1), 56–66. doi:10.1093/bib/bbs015

Yeo, G., Burge, C. B., Liebert, M. A., Yeo, G., & Burge, C. B. (2004). Maximum entropy modeling of short sequence motifs with applications to RNA splicing signals. *Journal of Computational Biology : A Journal of Computational Molecular Cell Biology*, *11*(2-3), 377–394. doi:10.1089/1066527041410418

Yi, M., Zhao, Y., Jia, L., He, M., Kebebew, E., & Stephens, R. M. (2014). Performance comparison of SNP detection tools with illumina exome sequencing data - An assessment using both family pedigree information and sample-matched SNP array data. *Nucleic Acids Research*, *42*(12). doi:10.1093/nar/gku392

Zerbino, D. R., Birney, E., Guerra, E., de Lara, J., Malizia, A., & Díaz, P. (2008). Velvet: Algorithms for de novo short read assembly using de Bruijn graphs. *Genome Research*, *18*(5), 821–829. doi:10.1101/gr.074492.107

Zook, J. M., Chapman, B., Wang, J., Mittelman, D., Hofmann, O., Hide, W., & Salit, M. (2014). Integrating human sequence data sets provides a resource of benchmark SNP and indel genotype calls. *Nature Biotechnology*, *32*(3). doi:10.1038/nbt.2835

Zook, J. M., Samarov, D., McDaniel, J., Sen, S. K., & Salit, M. (2012). Synthetic spike-in standards improve run-specific systematic error analysis for DNA and RNA sequencing. *PloS One*, *7*(7), e41356. doi:10.1371/journal.pone.0041356

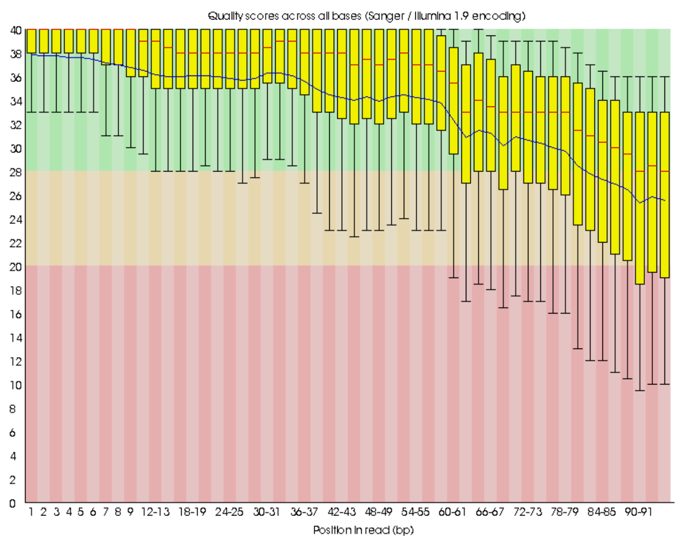
Appendix

Quality control (QC) checks

I performed the following quality checks on our in-house sequenced test data using FastQC tool (cf. Section 2.1):

1. Per base quality score: reports the distribution of base quality scores of all bases at each position in the read (cf. Figure A.1 a).
2. Per sequence quality score: reports the distribution of mean quality score for a subset of the reads (cf. Figure A.3 a).
3. Per base sequence content: shows the percentage of A, T, G, C at each position in the sequence reads (cf. Figure A.2 b).
4. Per base N content: shows the percentage of N at each position in the sequence reads (cf. Figure A.2 a).
5. Adapter content: reports the presence of an adapter sequence in the reads (can also be judged by percentage of overrepresented sequences) (cf. Figure A.1 b).
6. GC distribution: gives an idea of the GC bias in a sample (i.e. GC-poor or GC- rich sequences) (cf. Figure A.3 b).

a)



b)

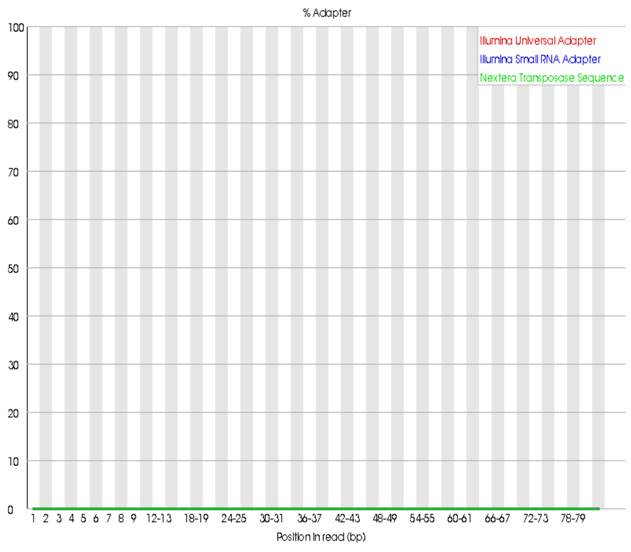
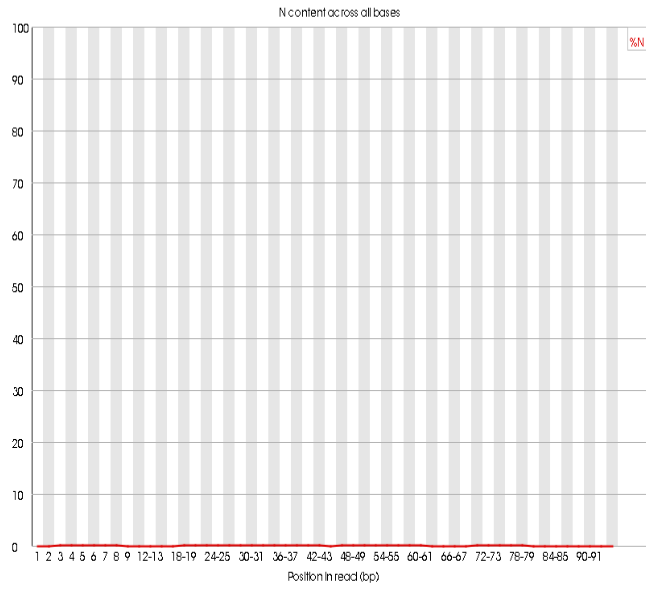


Figure A.1 a) Per base quality score distribution b) Adapter content distribution.

a)



b)

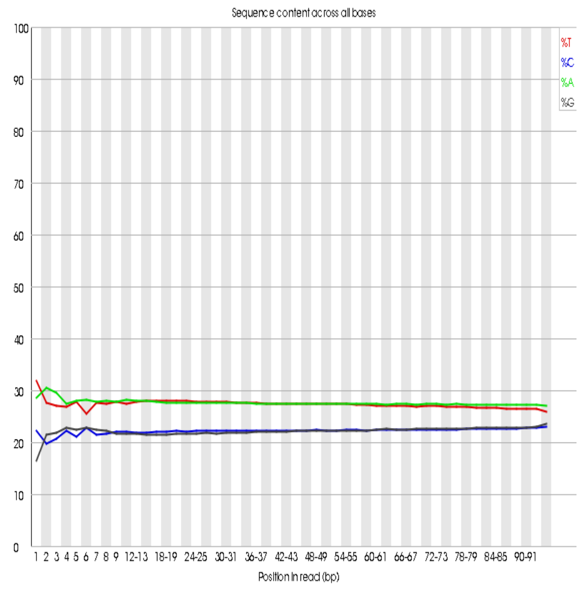
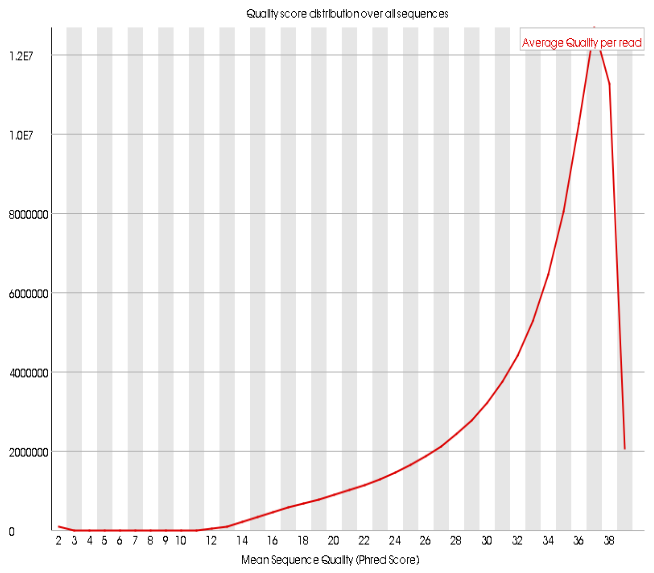


Figure A.2 a) Per sequence N content distribution b) Per base sequence content distribution.

a)



b)

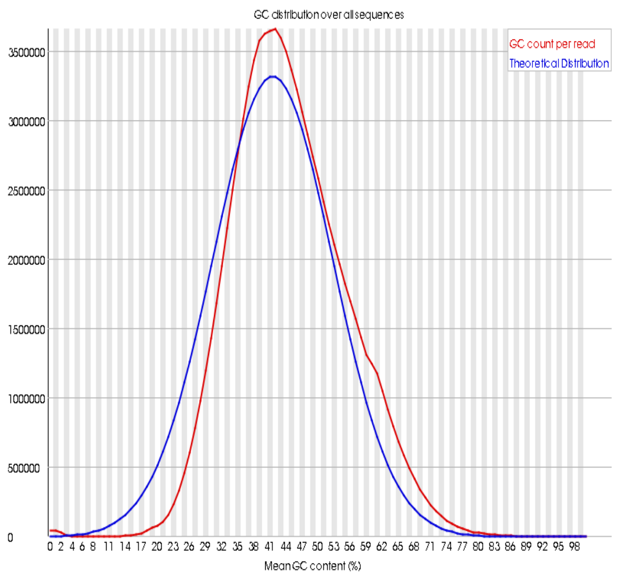


Figure A.3 a) Per sequence quality score distribution b) GC content distribution.

The QC reports containing the above mentioned quality checks were generated for one of the in-house sequenced test samples. All of these quality parameters have certain thresholds to distinguish between a bad and a good quality sample[102]. For a good quality

[102] http://www.bioinformatics.babraham.ac.uk/projects/fastqc/ (This and subsequent URLs are accessed on 28 June 2015).

sample, the majority of both quality scores (bullet points 1&2) should be greater than 20. All four nucleotides (A,T,G,C) should have a uniform distribution throughout the read and the N content should be small. Sometimes the first or last 5-10 bases from a read can show a non-uniform distribution of nucleotides (noise), which could be due to the presence of some random primers. As these can cause ambiguous calls or wrong calls, it is better to trim these bases[103]. There should not be a high GC bias in the sample because it can lead to non-uniform sample coverage or even genomic regions with no coverage (Chen et al., 2013; Dohm et al., 2008). An adapter contamination can lead to misalignments of reads, thus adapter trimming is also required in such situations (cf. Section 2.1.1). Based on these parameters, the sample quality can be judged. If a sample fails to pass all of these quality checks, then it should be either discarded or should undergo certain processing steps to improve the quality.

BWA algorithms and parameters testing

I also tested the effect of the following parameters:

1. -l seed length (default value: Infinite).
2. -k maximum edit distance in the seed (default value: 2).

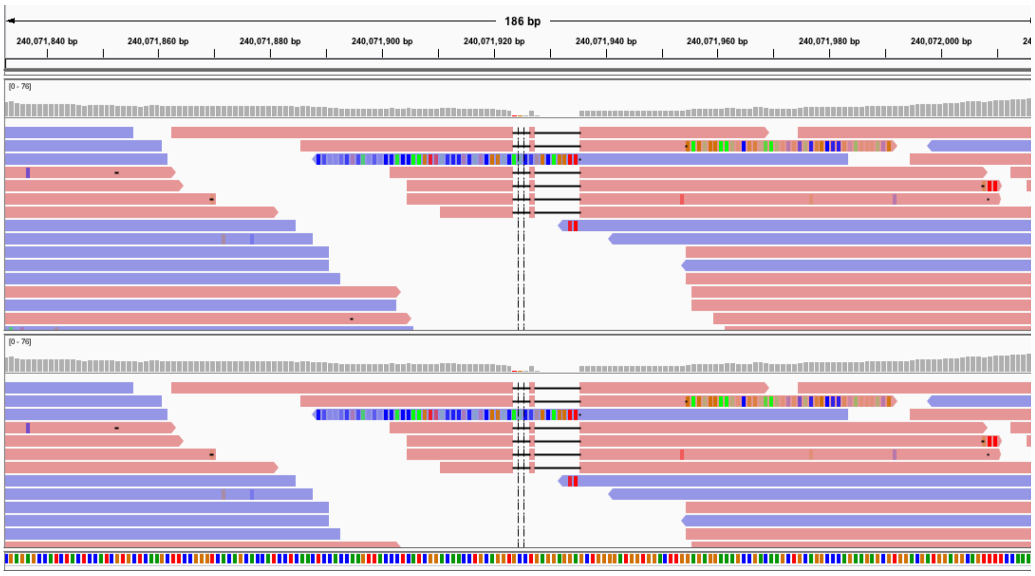


Figure A.4 Differences in alignment at default BWA parameters and at K=30 and L=50.

[103] Bad quality sample:
http://www.bioinformatics.babraham.ac.uk/projects/fastqc/bad_sequence_fastqc.html

The upper part of figure A.4, shows the alignment at k=30 and l=50, whereas the lower part shows alignment at default parameters[104]. Both of the alignments at this position are almost identical. I also compared alignments at other positions, but did not find any significant differences. Moreover, the alignment statistics at k=30 and l=50 are identical to the default values (cf. Table A.1).

BWA-aln	**Total Reads**	**% Aligned Reads**	**% Reads aligned in pair**	**% HQ Aligned Reads**	**Noise Reads**	**% HQ Aligned Bases (Q>20)**
default	175704592	0.988	0.993	0.923	292	0.881
k=30 & l=50	175704592	0.988	0.993	0.923	292	0.881

Table A.1 Alignment statistics for BWA-aln at default and at k=30,l=50.

Indel realignment

The Indel realignment method is a part of the Genome Analysis Tool Kit[105] (GATK) (DePristo et al., 2011; McKenna et al., 2010). It is a two-step process:

1. Check for suspicious intervals (esp. near Indels) in the alignment by using GATK's RealignerTargetCreator. It considers all Indel sites as suspicious which are not present in the known Indels list from Mills Divine (Mills et al., 2006), 1000 Genome (Abecasis et al., 2012) and the dbSNP dataset (Sherry et al., 2001).
2. Perform local realignment (MSA) over those intervals by using GATK's IndelRealigner. This procedure needs sufficient coverage (> 10x) to realign reads correctly. For low coverage data, multi-sample realignment should be performed. In this case, aligned reads from badly covered samples from a single study (or similar studies) are merged resulting in sufficient coverage for realignment of the reads.

[104] http://bio-bwa.sourceforge.net/bwa.shtml

[105] https://www.broadinstitute.org/gatk/guide/topic?name=methods

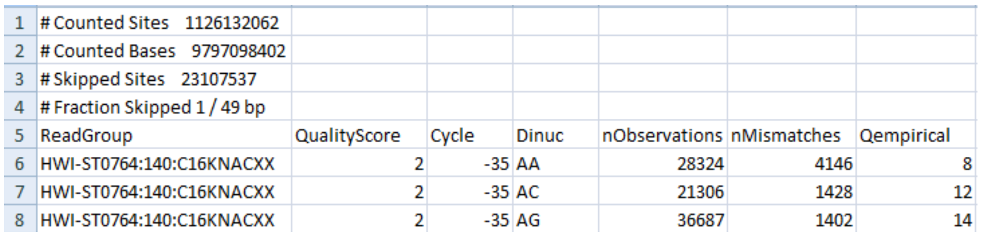
Base quality score recalibration (BQSR)

BQSR corrects the bases quality scores based on the analysis of the following covariates:

1. Read Group: Different lanes may have different error profiles. Thus the reads originating from different lanes are assigned to different read groups.
2. Quality Score: This is the base quality score assigned by the sequencer's base calling algorithm.
3. Machine Cycle: It reports the machine cycle (which corresponds to the position of the base in the read) that produced the analysed base. This information can be used to calibrate base qualities according to systematic errors due to their position in the read.
4. Di-nucleotide: This is the set of 2bp strings where the two bases represent the previous base and current base. It is used to identify sequencing errors dependent on a two-nucleotides context.

GATK's BaseRecalibrator[106] method performs BQSR by walking through all the reads in BAM file and constructing bins based on the above-mentioned covariates. For example, a bin contains all bases belonging to a certain read group, having the assigned quality score X produced by Y machine cycle and having Z nucleotide as the pervious nucleotide in dinucleotide context (cf. Figure A.5) (first 4 columns of row 6 in the table show the parameters defining different bins). Then, for each bin, it counts the number of bases within the bin and how frequent these bases mismatch the reference base in the alignment (excluding loci present in dbSNP). These calculations are stored in a table (cf. Figure A1.2), and used to calculate the empirical probability of error (i.e. p(error) = num mismatches/num observations) as well as new (empirical) quality scores[106]. Figure A.6 shows the improvements in base quality scores of our test sample. The figure at the left side shows the deviation of the reported quality scores (x-axis) from the empirical (observed) quality scores (y-axis) with high root-mean square error (RMSE). However, after BQSR corrections, base quality scores are almost identical to the empirical ones and the RMSE value dropped significantly (cf. right part of figure).

[106] http://gatkforums.broadinstitute.org/discussion/44/base-quality-score-recalibration-bqsr



1	# Counted Sites 1126132062						
2	# Counted Bases 9797098402						
3	# Skipped Sites 23107537						
4	# Fraction Skipped 1 / 49 bp						
5	ReadGroup	QualityScore	Cycle	Dinuc	nObservations	nMismatches	Qempirical
6	HWI-ST0764:140:C16KNACXX	2	-35	AA	28324	4146	8
7	HWI-ST0764:140:C16KNACXX	2	-35	AC	21306	1428	12
8	HWI-ST0764:140:C16KNACXX	2	-35	AG	36687	1402	14

Figure A.5 An example of the BQSR computed table of covariates calculations. First half of the machine cycles are denoted by positive (+) sign, whereas, last cycles are denoted by negative (-) sign.

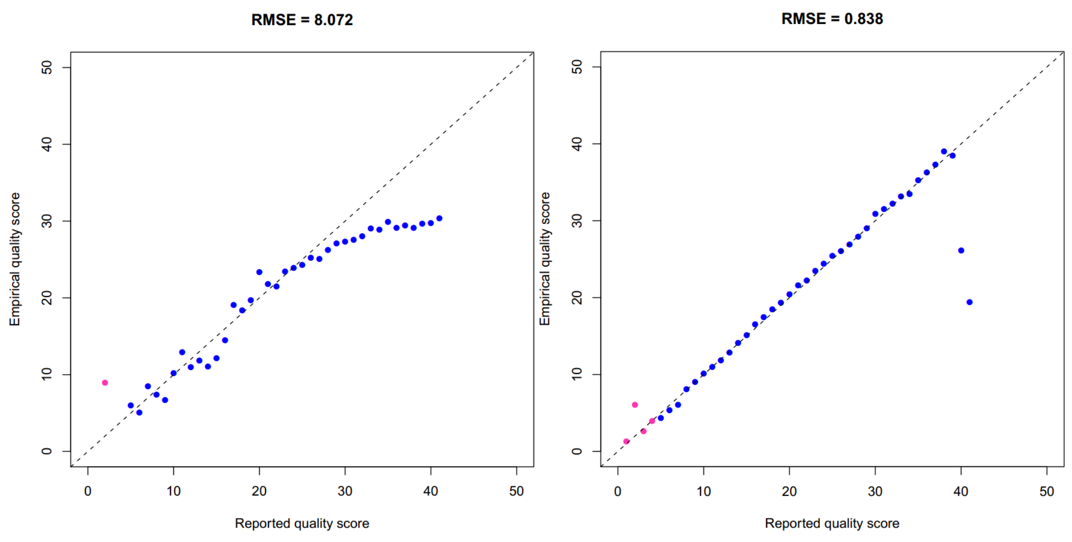


Figure A.6 Improvement in the base quality scores (of in-house sample) after the BQSR.

Variant quality score recalibration (VQSR)

As described in section 2.3.1, the performance of VQSR can be judged by clustering plots of any combination of the two annotations used during VQSR training. Moreover, the tranche plots provide a means for evaluation of the recalibrated variant list in context to both sensitivity and specificity. I tested the performance of VQSR with the following different combination of parameters:

- At default parameters: VQSR does not even run and throws the error: "Clustering with this few variants and these annotations is unsafe".
- At different values of –mG (maxGaussians) parameters.
- With a reference variant set (cf. Section 2.3.1) as additional training data along with a smaller value of mG than the default value (default=8).

After testing these combinations, I found that VQSR with a reference variant set and at –mG=6 works well for most of the data (cf. Figure 2.8).

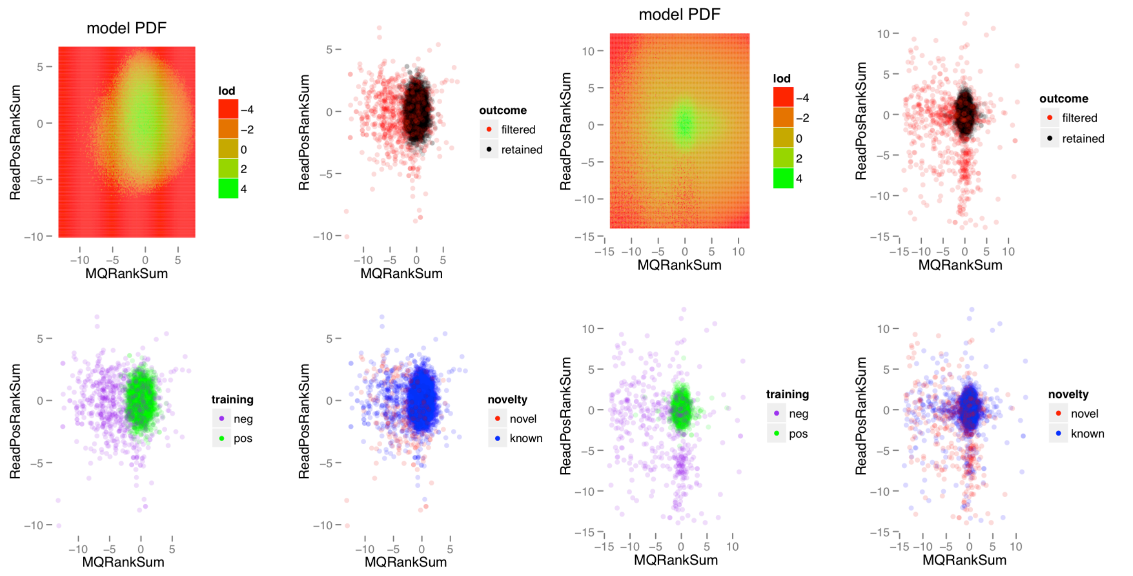


Figure A.7 VQSR clustering based on 2 annotations: ReadPosRankSum and MQRanksum. The left part of the figure shows clustering without usage of the reference variant set during VQSR but with mG 6 parameter. Whereas, the right part shows clustering with usage of the reference variants set but with default value of mG (mG=8).

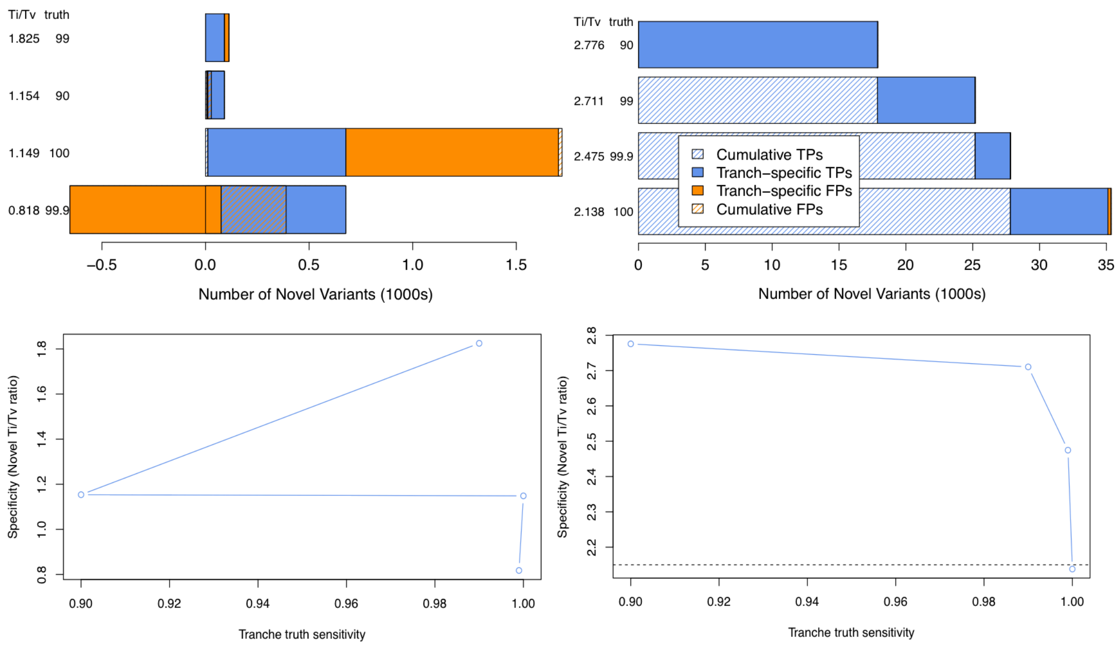


Figure A.8 VQSR's four-default tranches and their correlation with sensitivity and Ti/Tv ratio. The left part of figure shows tranches plots generated by VQSR with mG 6 parameter only. Whereas, the right part shows tranches plots generated by VQSR with usage of reference variant but with default value of mG.

Figure A.7 shows VQSR clustering for the control sample NA12878 based on ReadPosRankSum and MQRanksum annotations. These clusters do not show significant differences between VQSR at only mG 6 and VQSR with only the reference variant set (default mG), except the plot at the right is a bit denser than the first one as it is having more number of variants due to reference variant set. However, the tranche plots (cf. Figure A.8) clearly shows that the usage of a reference variant set is necessary to get a more specific and sensitive variant list. The recalibrated variant list generated from VQSR using the reference set is having a Ti/Tv ratio close to the standard value of 2.8 for the exome sequencing data. On the contrary, the other list generated without using the reference set has lots of FPs, thus a lower value of the Ti/Tv ratio. Overall, VQSR performance is dependent on the training data, thus the additional training data as provided by the reference variant set is required to perform a reliable variant clustering.

Annotations and filters used by FilterRSE

List of Annotations

The following annotations will be appended (by the FilterRSE tool in Annotation mode) to the info field of given VCF file. A description of each of these annotations is added to the VCF header as follows:

1. ##INFO=<ID=NotPA (P), Number=., Type="String", Description="Position is not a pipeline Artefact. "Value: Yes, if variant is called by other pipeline, otherwise Value: NO)">.
2. ###INFO=<ID=NIST (N), Number=., Type="String", Description="Present in the NIST list. Value: Yes, if the RSE location is shared by the control sample NA12878 and present in highly confident variants list (for this sample) provided by GAIB (NIST), otherwise Value: NO">.
3. ##INFO=<ID=BROAD, Number=., Type="String", Description="Present in the Broad list. Value: Yes, if the RSE location is shared by the control sample NA12878 and present in the validated variants list (for this sample) provided by Broad institute, otherwise Value: NO">.

4. ##INFO=<ID=17MER, Number=., Type="String", Description="String of 17consecutive bases around the RSE positions (8 bases upstream, reference base at RSE location, 8 bases downstream">.
5. ##INFO=<ID=CV, Number=., Type="String", Description="Presence of the RSE location in the integrated variant list of common variants (CV) constructed by using 1000 genomes and EXAC VCF files. Value: NO or Yes followed by rsid (DBSNP id) and minor allele frequency (MAF) computed by EXAC if the variant is present in both datasets (1000 genomes and EXAC), otherwise MAF from an individual dataset">.

List of Filters

The following filters can be applied by the FilterRSE tool (in "Filter" mode) either individually or in combination. A description of each of these filters is added to the VCF header as follows:

1. ##INFO=<ID=N, Number=., Type="String", Description="Filter out those RSE locations which are not present in the NIST list">
2. ##INFO=<ID=P, Number=., Type="String", Description="Filter out those RSE locations which are pipeline artefact (PA) (not called by other pipeline)">
3. ##INFO=<ID=C, Number=., Type="String", Description="Filter out those RSE locations which are not in the common variant list (100G+EXAC)">
4. ##INFO=<ID=C_MAF, Number=., Type="String", Description="Filter out those RSE locations which are not in the common variant list (100G+EXAC) only if they have less MAF then the given threshold".

Pseudocode for quality trimming script

The following pseudocode provides the description of the implementation of quality trimming algorithm in Perl programming language. The theoretical description about this algorithm is provided in the section 2.1.1 of Chapter 2.

```
        TrimBases (F, Q, L, B, S)
Input:
        F=Fastq file
        Q=Base quality score threshold for trimming (default = 10)
        L=minimum length of a read to retain after trimming (default = 30)
        B=maximum percentage of bad quality bases in read in order to keep it (default = 20)
        S=file for statistics output (trimmed reads and trimmed bases counts)
        O= Output: Fastq file after trimming action
###
While read each line in the file F
        Store read sequence and corresponding quality score string in r and q respectively
        Screen first 5 bases from 3‘ end of read and select base having quality score < Q as    start
base (at position SB) for trimming action
        Diff = 0  ### Initialize Diff (i.e. difference between quality score of certain base and Q)
        Pos = SB  & MaxPos = Pos
        MinPos =0 #### Starting base of read
        while (Pos>0 and Diff>=0)  ###  screen each base from 3’ end
                        Diffcurr = QPos– Q   ### QPos denotes the quality score of base at position
                                        Pos
                        if (Diff >0)
                                Diff = Diff + Diffcurr
                                MaxPos = Pos     ## Position of current base
                        end
                        Pos-- ## move to next base
        done
        MaxPos = MaxPos -1  ## last base of trimmed read
        Screen first 5 bases from 5‘ end of read and select base having quality score < Q as start base
        (at position SB) for trimming action
        Pos=SB & MinPos=Pos   ## starting base of read
        while (Pos< MaxPos and Diff>=0)  ###  screen each base from 5’ end
                        Diffcurr = Quality score of current base – Q
                        if (Diff >0)
```

```
                    Diff = Diff + Diffcurr
                    MinPos = Pos      ## Position of current base
                end
                Pos++    ### move to next base
        done
        MinPos = MinPos+1   ## first base of trimmed read
        if (MaxPos-MinPos < L)
            Discard entire read
        else
            Trim bases from MinPos til MaxPos
            Count LQbases (bases having less quality score than Q)
            if (LQbases >B/100*length(q))
                Discard entire read
            end
        end
        Count trimmed and discarded bases and reads
        print trimmed read and respective quality score in file O
done
        print trimmed and discarded read and bases counts statistics in file S
```

Pseudocode for expected & observed kmer generation script

The following pseudocode provides the description of the implementation of both the expected and observed kmer generation scripts written in Perl. The theoretical description of the observed and expected kmer generation is provided in the section 4.1.3 of Chapter 4. Few terminologies like OCmid, OCstart, OCend used in pseudocode and formula using these values for coverage correction are also described in the section 4.1.3 of Chapter 4.

```
ExpectedKmerGen(A, S, k, O)
Input:
        A=Base coverage file from one strand (generated by mpileup)
        S=strand of aligned reads
        k=desired length of kmer
        O=Output directory to store outfile containing expected kmer and their counts
###

While read each line in the file A
        Cov=0    ##will be the sum of coverage values of each base in kmer
        Store the reference base in array @bases (as starting base of the kmer)
        Get coverage value of this base and subtract Overcounts (OCstart)
        Add the corrected coverage value to Cov

        for (i from 1 to k-2)                ### Get middle bases of kmer
                Get reference base from ith line and append in array @bases
                Get coverage value of this base and subtract Overcounts (OCmid)
                Add the corrected coverage value to Cov
        Done

        Get the last reference base of kmer $(k-1)_{th}$ line after the starting base and append to array
        @bases
        Get coverage value of this base and subtract Overcounts (OCend)
        Store genomic location of last base as kmer position (Pos)
        Concatenate bases in array @bases and store in kmer
        kmer coverage (kmerCov)=Cov/k
        print kmer, kmerCov, and Pos to kmer-file    ### generated kmer is stored in a tab separated
        file
done

## Add coverage value of from multiple occurrences of each generated kmer in kmer-file
for each generated kmer
        if read strand S is "reverse"
                kmer = Reverse complement of generated kmer
                count = Sum of the coverage value from multiple occurrences of kmer
                print kmer and its count to output file in directory O
done
```

```
ObserevedKmerGen(A, S, k, O)
Input:
        A=SAM file having aligned reads from one strand
        S=strand of aligned reads
        k=desired length of kmer
        O=Output directory to store outfile containing observed kmer and their counts
###

While read each line in the file A


        Extract read sequence from line
        Split read sequence by each base and store each base in array @bases

        for each base i in @bases (i from 1 to length(@bases)-(k+1))
                kmer = i   ### Initialize kmer with base i


                        for each base j in @bases (j from i to i+k-1)
                                kmer = kmer.j   ### Concatenate base j to kmer
                        done
                print kmer to kmer-file    ### generated kmer is stored in a file
        done
done
count number of occurrences of each generated kmer in kmer-file
for each generated kmer
        if read strand S is ”reverse”
                kmer = Reverse complement of generated kmer
                print kmer and its count to output file in directory O
done
```

Pseudocode for FilterRSEs tool

The following pseudocode provides the description of the implementation of the FilterRSEs tool written in Perl. This tool uses the above mentioned filters or annotations (cf. Section “Annotations and filters used by FilterRSE”) to filter out the RSE locations from a given VCF file or to append RSE-associated annotation respectively. The theoretical description of the filter and annotation mode of this tool is provided in section 4.2 of Chapter 4.

```
FilterRSEs (V, M, O, F, A)
Input:
        V=Given VCF file for RSEs filtering
        M=Mode of script („Filter“ or „Annotation“)
        O=Output directory to store output files (RSEfiltered, onlyRSElocs or RSEannotated)
        F=Single filter name or comma separated list of filters (if Filter mode is selected).
        A=MAF value (if DBALL_MAF filter is used)
###
R=List of RSEs
print all annotations or information about the filters (depending on mode) into the Info field of VCF
header
While read each line in the file V
        Store all variant locations (VarLoc) in a Hash %VCF
done
While read each line in the file R
        Store all RSE locations (RSELoc) in a Hash %RSE
done
for each VarLoc in %VCF
        if (VarLoc exists in %RSE & Mode eq "Annotate")
                Append the info field of VarLoc with RSE annoataions
                print this appended line into VCF.RSEannotated file in directory O
        end
        if (VarLoc exists in %RSE & Mode is "Filter")
                Checks for the list of given filters and store in an array @Filter
                for Filter i in @ Filter
                        if (FilterValue eq “NO”) ### the checked value of filter can change
                                    according to filter depending of RSE annotations value
                                FilterOut VarLocs according to given filters
                                print Filtered VarLocs and used filter in additional column in
                        VCF.RSElocs_withFilter file in directory O
                        else
                        print unfiltered VarLocs in VCF. RSEfiltered file in directory O
                        end
                done
        else    print unfiltered VarLocs in VCF. RSEfiltered file in directory O
        end
```

List of RSEs belonging to error class 1

The following table contains list of RSEs (class 1) that resulted from systematic error detection approach presented in Chapter 4. It consists information on chromosomal location (column: “Chr” and “Location”) of RSEs and surrounding 9mer or 17mer (column: “Kmer”) along with their orientation (column: “Strand”).

Chr	Location	Strand	Kmer	InNIST	InBroad	InHC	InDBall(rsid_MAF)	MotifLoc*
1	248813547	Rev	TTCTCTGTCATCTCGGG	NO	NO	Yes	Yes_rs201161767_0.184000625586487	UbD
1	248801633	Rev	AGATGTGGTTCCTCCCA	NO	NO	Yes	Yes_rs74901534_0.015	UbD
1	248801633	For	TGGGAGGAACCACATCT	NO	NO	Yes	Yes_rs74901534_0.015	UbD
1	248801611	Rev	TCAGGGTGG	NO	NO	Yes	NO	Ub
1	248801611	For	CACAGTCGCCACCCTGA	NO	NO	Yes	NO	UbD
1	248801610	Rev	CAGGGTGGC	NO	NO	Yes	Yes_rs150878651_0.00547913266495687	Ub
1	248801610	For	TCACAGTCGCCACCCTG	NO	NO	Yes	Yes_rs150878651_0.00547913266495687	UbD
1	248801602	For	TTCCTGATCACAGTCGC	NO	NO	Yes	Yes_rs370874670_0.00149476831091181	UbD
1	248801592	Rev	GCTAGCAGG	NO	NO	Yes	Yes_rs78622116_0.0206923995224831	bD
1	248801592	For	CCTGCTAGCCCTTCCTG	NO	NO	Yes	Yes_rs78622116_0.0206923995224831	UbD
1	248801566	For	TCTGATACT	NO	NO	Yes	Yes_rs74154510_0.0368394308943089	bD
1	248458689	For	ATGAGGGAAAGTTGGCT	NO	NO	Yes	Yes_NA_0.000147972773009766	UbD
1	248458676	For	CAGCATCATGTCCATGA	NO	NO	Yes	Yes_NA_0.000192980340127849	UbD
1	248112762	Rev	ATGGTGGCACTCAAAAA	Yes	NO	Yes	NO	UbD
1	248112762	For	TTTTTGAGTGCCACCAT	Yes	NO	Yes	NO	UbD
1	247978541	Rev	CACCTTCTC	Yes	Yes_TP	Yes	NO	Ub
1	247978541	For	AGGATAAGGAGAAGGTG	Yes	Yes_TP	Yes	NO	UbD
1	236723177	Rev	TCTGAGTGCCTTCAGCT	Yes	Yes_TP	Yes	NO	UbD
1	236702209	For	CATGAAACC	NO	Yes_TP	Yes	NO	bD
1	235856901	Rev	TGGTCTTGCACCAAATG	Yes	NO	Yes	Yes_rs3754232_0.247603833865815	UbD
1	231064578	For	CTACCTCAC	Yes	NO	Yes	NO	bD
1	230820763	For	GAAGTCTCACATTAGAC	Yes	NO	Yes	Yes_rs397701344_0.233027156549521	UbD
1	230459433	Rev	GAGGGGGGGCCTTTGGG	NO	NO	Yes	NO	UbD
1	228461408	For	GCCCTGAAGGGAGCGGG	Yes	NO	Yes	NO	UbD
1	228430947	Rev	TGGCACTGGCCCCTGCC	NO	NO	Yes	Yes_NA_0.00854445109382273	UbD
1	227222515	Rev	TTGGCTGTTTGTCTGTC	Yes	Yes_TP	Yes	NO	UbD
1	227210888	For	TTATTATTAATGTGACC	Yes	NO	Yes	Yes_rs11578479_0.286142172523962	UbD
1	227172903	For	GGGATGTCGGGAGCTGA	Yes	Yes_TP	Yes	Yes_rs2297415_0.269568690095847	UbD
1	224318151	For	TGATTTTTGTTCTTTTT	NO	Yes_TP	Yes	NO	UbD
1	219383848	For	GTATTCTACTTTTAAGT	NO	Yes_TP	Yes	NO	UbD
1	213062017	Rev	CAACAACAAATAAAACC	NO	NO	Yes	NO	UbD
1	212209367	Rev	CAGAATTGGAGTCGGCC	Yes	NO	Yes	Yes_rs397844077_0.259584664536741	UbD
1	209785011	Rev	AGGGTACACGGAGCTCC	Yes	NO	Yes	NO	UbD
1	206766857	For	TGTGTGTGC	NO	NO	NO	Yes_rs28629842_0.281749201277955	Ub
1	206580084	Rev	ACCTTTAAGTGTAGGGA	NO	NO	Yes	NO	UbD
1	206575141	Rev	TCAATAAGTGGAATGTG	NO	NO	Yes	NO	UbD
1	206567100	Rev	ATAGTATCGGGGAATGT	NO	NO	Yes	NO	UbD
1	205308914	Rev	GCGGCAGGCGGGATGGG	NO	NO	NO	Yes_rs116078922_0.0092151836933968	UbD
1	205275206	For	GGCTTGAGAGTTGTGCT	Yes	NO	Yes	NO	UbD
1	205063949	For	AAGGGCTACACAAAGCA	Yes	Yes_TP	Yes	Yes_rs3738154_0.285543130990415	UbD
1	204413297	Rev	CTTTCTCCGAAACAGGC	Yes	Yes_TP	Yes	NO	UbD
1	202731986	Rev	GATGATGAGGAAGTGGA	NO	Yes_TP	Yes	NO	UbD
1	201179984	Rev	TCAGAACCC	NO	NO	NO	Yes_NA_0.0802245563201738	bD
1	201179984	For	GGGTTCTGA	NO	NO	NO	Yes_NA_0.0802245563201738	Ub
1	201178926	Rev	TAACCTGCCTTATTCAC	NO	NO	Yes	Yes_rs6702960_0.1689453125	UbD
1	201178926	For	GTGAATAAGGCAGGTTA	NO	NO	Yes	Yes_rs6702960_0.1689453125	UbD
1	201178924	Rev	ACCTGCCTTATTCACTG	NO	NO	Yes	Yes_rs6678538_0.135758998435055	UbD
1	201178924	For	CAGTGAATAAGGCAGGT	NO	NO	Yes	Yes_rs6678538_0.135758998435055	UbD
1	201178904	For	GGGTTCTGAAGAAATGG	NO	NO	Yes	Yes_rs201227267_0.136434349719975	UbD
1	201178904	Rev	CCATTTCTTCAGAACCC	NO	NO	Yes	Yes_rs201227267_0.136434349719975	UbD
1	181706552	For	TTCCTTTGGCAAGGGCC	Yes	NO	Yes	Yes_rs2280866_0.275359424920128	UbD
1	179783095	For	GGTGGCCGCGAGCCACT	NO	NO	NO	Yes_rs202244671_0.0230830196156132	UbD
1	173921301	Rev	TTTTTTTTGTTTTTTTT	NO	NO	NO	Yes_rs199928760_0.066831343488404	UbD
1	173921292	Rev,For	TTTTTTTTC	NO	NO	NO	Yes_NA_0.0531843724913032	Ub
1	173921292	Rev	CTTTCTCCA	NO	NO	NO	Yes_NA_0.0531843724913032	bD
1	170993444	For	TATATGGGGTGTGTGTG	NO	NO	Yes	NO	UbD
1	169580972	Rev	TTGCTCTCTGTGTGGCT	Yes	NO	Yes	NO	UbD
1	169272451	Rev	CTATTGCCTTTTTGATT	Yes	Yes_TP	Yes	NO	UbD
1	165376228	Rev	CTACAGAGAGTGGCCAG	NO	NO	NO	Yes_rs111545739_0.180311501597444	UbD
1	161594525	Rev	TGTTGGCACTTGTCAAA	NO	NO	Yes	NO	UbD
1	160990875	Rev	GTGTCGGGAGCCTGATC	NO	NO	NO	NO	UbD
1	160990875	For	GATCAGGCT	NO	NO	NO	NO	Ub
1	160090681	For	CTCTCCCTTCCTCCCTC	NO	Yes_TP	Yes	NO	UbD
1	157062364	For	GCTAACCCA	Yes	Yes_TP	Yes	Yes_rs1176536_0.208067092651757	bD
1	156565049	For	CACCTAAAAGTTTCCTC	Yes	Yes_TP	Yes	NO	UbD
1	156532250	For	GATCTTTCTCCGAGGCA	Yes	NO	Yes	NO	UbD
1	155785578	Rev	TGAATGTTAGTTGGTAA	NO	Yes_TP	Yes	NO	UbD
1	155785578	For	TTACCAACTAACATTCA	NO	Yes_TP	Yes	NO	UbD
1	155733257	Rev	TCTGAGCCAGAAGAAGA	NO	Yes_FP	NO	Yes_rs607834_0.060207336523126	UbD
1	155733257	For	TCTTCTTCTGGCTCAGA	NO	Yes_FP	NO	Yes_rs607834_0.060207336523126	UbD
1	154227239	For	TAAAGCTGCATCAGTGC	NO	NO	Yes	NO	UbD
1	153636472	For	ACTACTACTAGTAAATA	NO	NO	NO	Yes_rs372540759_0.220740023486269	UbD
1	153636471	For	ACTACTACTAGTAAATA	NO	NO	NO	NO	UbD
1	152681680	Rev	CCCAGAGCTGGAGCCAC	Yes	Yes_TP	Yes	NO	UbD
1	152681680	For	GTGGCTCCAGCTCTGGG	Yes	Yes_TP	Yes	NO	UbD
1	152195728	For	TTTTTTTTT	NO	Yes	Yes	Yes_rs561299511_0.211165481308319	bD
1	152195728	Rev	AAAAAAAAA	NO	Yes	Yes	Yes_rs561299511_0.211165481308319	Ub
1	152195728	Rev	ATGCCTAAA	NO	Yes_TP	Yes	Yes_rs561299511_0.211165481308319	bD
1	151789609	For	TGTGTGTGCGCGCGCGC	NO	NO	Yes	NO	UbD
1	151700189	Rev	CAGACTCTGGGAGAATC	Yes	NO	Yes	NO	UbD
1	151634803	Rev	TATTTAATCTATCTTCC	Yes	NO	Yes	Yes_rs11808761_0.269968051118211	UbD
1	151212399	For	CGGGCCTCCTAACTCTA	Yes	Yes_TP	Yes	NO	UbD
1	145507826	Rev	TCCTTACGA	NO	NO	Yes	Yes_rs872786_0.288738019169329	bD
1	145282093	Rev	TGATCACCCACTGTTAA	NO	NO	Yes	NO	UbD
1	145281247	For	TAGCCTTTTGTAAAATG	NO	NO	Yes	NO	UbD
1	145273520	Rev	GAGCCAGACACCATGAT	NO	NO	Yes	NO	UbD
1	145115602	For	GTTAGCCTCAAAATAAA	NO	NO	Yes	NO	UbD
1	145112313	For	CTTTTAATTTCCTACCT	NO	NO	Yes	NO	UbD
1	145112285	For	TGAGCCACCGTGCCTGG	NO	NO	NO	NO	UbD
1	145109809	Rev	GAAAAGGAAGCAGAAGC	NO	NO	Yes	NO	UbD
1	145109456	For	CAGAGATTCTGCTTTAA	NO	NO	Yes	NO	UbD
1	145103844	For	ATAATATGGTGCTTCTC	NO	NO	Yes	NO	UbD
1	145021024	For	CATAGTGAGAGAGTGAG	NO	NO	Yes	NO	UbD
1	145020936	For	AGCTACTCAGGAGGCTG	NO	NO	Yes	NO	UbD
1	144994861	Rev	ATGTCGCCGTAGCCCCA	NO	NO	Yes	NO	UbD
10	135439305	Rev	TTATTGACGAGTAAATT	NO	NO	Yes	NO	UbD
10	135439305	For	AATTTACTCGTCAATAA	NO	NO	Yes	NO	UbD
10	135439287	For	GAGTGCTAAGATCTCAA	NO	NO	Yes	NO	UbD
10	135439220	For	CCATCTCAG	NO	NO	Yes	NO	bD
10	135438929	Rev	CACCGTCAATTCGAAAA	NO	NO	Yes	Yes_rs201901202_0.215164525905614	UbD
10	135438929	For	TTTTCGAATTGACGGTG	NO	NO	Yes	Yes_rs201901202_0.215164525905614	UbD
10	135350490	For	GATGGGTGG	NO	NO	Yes	NO	bD
10	135105932	For	CAGCATGCACACTCCGT	NO	NO	NO	NO	UbD
10	135103301	For	CCCAAGTCC	NO	Yes_TP	NO	NO	bD
10	135084232	For	GCGGTCTCAGCTCACCT	Yes	Yes_TP	Yes	NO	UbD
10	135077121	For	CGTCAGCCC	NO	NO	Yes	NO	bD
10	134902396	For	CACGCTTCC	Yes	Yes_TP	Yes	NO	Ub
10	124329672	Rev	AAGCTGCGAGGTGGAAA	NO	Yes_TP	Yes	NO	UbD
10	124329672	For	TTTCCACCTCGCAGCTT	NO	Yes_TP	Yes	NO	UbD
10	123970021	Rev	ATCGGCTTGTCTTCATC	NO	Yes_FP	NO	Yes_rs201905665_0.0119893589883751	UbD
10	123970021	For	GATGAAGACAAGCCGAT	NO	Yes_FP	NO	Yes_rs201905665_0.0119893589883751	UbD
10	120933134	For	AAAAAAAAAAAAAAAAA	Yes	NO	Yes	NO	UbD
10	120922017	Rev	TCCAGTGCATGCTGGTG	Yes	NO	Yes	Yes_rs12413842_0.279552715654952	UbD
10	113921450	For	GGAGCAGGCAAGCCACA	NO	Yes	NO	Yes_rs199856746_0.0325124081171075	UbD
10	113921449	For	GGAGCAGGCAAGCCACA	NO	NO	NO	NO	UbD
10	113921447	For	GGAGCAGGCAAGCCACA	NO	NO	NO	NO	UbD
10	112266822	Rev	GTGGGTCGCGCAGGCCT	NO	Yes_TP	Yes	NO	UbD
10	112266822	For	AGGCCTGCGCGACCCAC	NO	Yes_TP	Yes	NO	UbD
10	108432828	Rev	AGCTCAGGGTGAACGCC	NO	NO	NO	Yes_rs11193010_0.295527156549521	UbD
10	105799550	For	ACACACACGCACACGCA	NO	NO	Yes	Yes_rs11191904_0.277156549520767	UbD
10	105363565	Rev	ACTGGCTTGTCTGCACT	NO	NO	Yes	Yes_NA_0.23893907806347	UbD
10	102265056	For	CTTCACAAACACACACA	NO	NO	Yes	NO	UbD
10	95405665	For	ATGAACCTATGTGTAAT	Yes	Yes_TP	Yes	NO	UbD
10	77818594	Rev	GGGTGTAAGCACTGCCA	Yes	NO	Yes	Yes_rs7924288_0.260583067092652	UbD
10	75620757	Rev	CTTCAGCAAATCATCAC	Yes	NO	Yes	Yes_rs2664282_0.284944089456869	UbD
10	75559662	For	GAGACTCTATCTCAAAA	NO	Yes_TP	Yes	NO	UbD
10	73571581	Rev	GGCTCCGGT	Yes	NO	Yes	NO	bD
10	73115941	Rev	AGGAAGACAGTGGCCGT	Yes	Yes_TP	Yes	NO	UbD
10	73115941	For	ACGGCCACTGTCTTCCT	Yes	Yes_TP	Yes	NO	UbD
10	71391719	Rev	TCTTCCCCAGCTCTCTG	Yes	NO	Yes	NO	UbD
10	65225244	For	GGGGCGCTGACTCTCTT	Yes	Yes_TP	Yes	NO	UbD
10	62551889	Rev	CAGGAAATTAAAAATTT	Yes	Yes_TP	Yes	NO	UbD
10	62551889	For	AAATTTTTAATTTCCTG	Yes	Yes_TP	Yes	NO	UbD
10	61802374	For	TAGTGAAAAATATAAGT	Yes	NO	Yes	NO	UbD
10	51623004	For	CTTGTAAATGCCAGCAA	NO	NO	Yes	NO	UbD
10	51620499	Rev	CGCCTTCAG	NO	NO	Yes	NO	bD
10	46999189	For	ATGGGCGGCAGTGACCT	NO	NO	Yes	NO	UbD
10	33093858	For	ACACACACA	NO	NO	Yes	NO	bD
10	29783908	Rev	ACCAGATATGCAGTTAG	NO	Yes_FP	Yes	Yes_rs78773460_0.177077479629109	UbD
10	29783908	For	CTAACTGCATATCTGGT	NO	Yes_FP	Yes	Yes_rs78773460_0.177077479629109	UbD
10	17032559	Rev	TTTCCATGTGCTGTTGC	Yes	Yes_TP	NO	Yes_NA_1.65612268556855e-05	UbD
10	13541724	For	TTAGAAATAGAACAGTA	Yes	Yes_TP	Yes	NO	UbD
10	11356093	For	GTTTTCCTTTTTTCAGT	Yes	Yes_TP	Yes	NO	UbD
10	5926079	Rev	TTGAGGTATTGAATTTT	NO	Yes_TP	Yes	NO	UbD
10	5682855	Rev	TGTGTTCCATTGCAGGT	Yes	NO	Yes	NO	UbD
10	1061625	For	CGCTGAGCACTGAGCCT	NO	NO	NO	NO	UbD
11	133800773	For	CCGCAGAGT	Yes	NO	Yes	NO	bD
11	130064747	Rev	ACACACACATGCACACA	NO	NO	Yes	NO	UbD
11	124747333	Rev	TTCAGGACA	NO	NO	NO	Yes_rs10790713_0.236022364217252	bD
11	119548052	Rev	CCCACCCACACTTCCCT	Yes	NO	Yes	Yes_rs10892429_0.204672523961661	UbD
11	119205735	For	ATTTTCCTTCCAAATTA	NO	Yes_TP	Yes	Yes_rs12797083_0.295127795527157	UbD
11	116703671	Rev	TACAGGGGCAGCCCTGG	NO	NO	Yes	Yes_rs4225_0.291333865814697	UbD
11	113848184	For	CCACCTGCTCCGACCTT	Yes	Yes_TP	Yes	NO	UbD
11	103086399	For	ACTGTGTGTGTAAAATT	NO	NO	Yes	NO	UbD
11	94704092	For	TTTTAGAATAGCATGGT	Yes	Yes_TP	Yes	NO	UbD
11	93494878	Rev	AATTATTTCTTTTTCTA	NO	NO	NO	NO	UbD
11	71740378	Rev	CAAACCATGTTGCTTCT	Yes	Yes_TP	Yes	NO	UbD
11	71714526	Rev	ATGCTGACCTCACCTGG	Yes	Yes_TP	Yes	NO	UbD
11	71714526	For	CCAGGTGAGGTCAGCAT	Yes	Yes_TP	Yes	NO	UbD
11	71640096	For	TGAGGGGGCGAGCCGTG	NO	NO	NO	Yes_rs376190587_0.00250391236306729	UbD
11	71640090	For	CTGCGGTGAGGGGGCGA	NO	Yes_FP	NO	Yes_rs201521523_0.00108837614279495	UbD
11	70544937	Rev	CAGCCTCATGTGCTGTG	Yes	NO	Yes	Yes_rs7117514_0.296924920127796	UbD
11	70507605	For	AGGCGTATACACACACA	NO	NO	Yes	NO	UbD
11	68566651	For	TACGGAAAAAATTTCATC	NO	NO	Yes	Yes_rs61887064_8.28129917021382e-06	UbD
11	68566650	For	ATACGGAAAAATTTCAT	NO	NO	NO	Yes_rs61887063_5.20616409829238e-05	UbD
11	68566649	For	TACGGAAAAATTTCATC	NO	NO	NO	NO	UbD
11	68566648	For	CATACGGAAAAATTTCA	NO	NO	NO	Yes_rs61887062_0.000998668442077223	UbD
11	67262289	For	GCGAGTGGGCGAGGGGG	NO	NO	Yes	Yes_rs7932071_0.0106792499689557	UbD
11	67223920	Rev	GGCTCCCTGACTGACTG	NO	Yes_TP	Yes	NO	UbD
11	66109779	Rev	GTTCAGCATCCTGCTGC	NO	NO	Yes	NO	UbD
11	65631879	For	ACAGGGTCCCGGCACCA	Yes	NO	Yes	NO	UbD
11	65629932	Rev	CAGGGTCGAGCACTCCC	Yes	Yes_TP	Yes	NO	UbD
11	65629932	For	GGGAGTGCTCGACCCTG	Yes	Yes_TP	Yes	NO	UbD
11	64606910	For	TGGCCCGCCTTGTATCC	Yes	Yes_TP	Yes	NO	UbD
11	64603844	For	CTCACCCAC	NO	NO	Yes	Yes_rs71471960_0.282947284345048	bD
11	61725599	For	CTACCACATCCTCCTCC	NO	NO	Yes	NO	UbD
11	57137538	Rev	CCATGCCCCTTGCTCAG	Yes	Yes_TP	Yes	NO	UbD
11	56468835	Rev	GAAATACACAACACTCC	NO	NO	Yes	NO	UbD
11	56468817	Rev	TCTAAGGTCGTCTTTGT	NO	NO	Yes	NO	UbD
11	56468198	Rev	CCAGCAGGTAGCACTCA	NO	Yes_FP	Yes	NO	UbD
11	56468198	For	TGAGTGCTACCTGCTGG	NO	Yes_FP	Yes	NO	UbD
11	56468047	Rev	ATTTCCAGTGAAAAAAT	NO	NO	Yes	NO	UbD
11	56468047	For	ATTTTTTCACTGGAAAT	NO	NO	Yes	NO	UbD
11	56468020	Rev	GTGGAGGCAGGAGTCAT	NO	Yes_TP	Yes	NO	UbD
11	56468020	For	ATGACTCCTGCCTCCAC	NO	Yes_TP	Yes	NO	UbD
11	56467810	For	TTCATGACAGTAATGCA	NO	NO	Yes	NO	UbD
11	56467787	For	CTTATGTAACATAATTC	NO	NO	Yes	NO	UbD
11	56143559	Rev	TGCCATTAGGAAGCTAT	NO	NO	Yes	NO	UbD
11	56143559	For	ATAGCTTCCTAATGGCA	NO	NO	Yes	NO	UbD
11	48373833	Rev	CATGTACTCTTTTCCGT	NO	NO	Yes	NO	UbD

1	144946876	Rev	GTCAAAAGG	NO	NO	Yes	NO	bD
1	144923535	For	GAATAAAGAGAAGAACT	NO	NO	Yes	NO	UbD
1	144922523	Rev	CAGGGAACGTGAGGTAA	NO	NO	NO	Yes_rs2455994_0.291196094385679	UbD
1	144922523	For	TTACCTCACGTTCCCTG	NO	NO	NO	Yes_rs2455994_0.291196094385679	UbD
1	144922109	Rev	TTTATTTTTAGTCTTTG	NO	NO	Yes	NO	UbD
1	144922109	For	CAAAGACTAAAAATAAA	NO	NO	Yes	NO	UbD
1	144919094	Rev	CACCCTCTGCTCCTCTT	NO	NO	Yes	NO	UbD
1	144919008	Rev	TTAAAAAGGTAATATGT	NO	Yes_TP	NO	Yes_rs11296953_0.295079561758184	UbD
1	144917827	For	TCCAGAGCAACAGCTTT	NO	NO	NO	Yes_rs375854543_0.295409035409035	UbD
1	144916676	Rev	CCTCCAGTGGCTGAAAG	NO	NO	NO	Yes_rs1698683_0.294959405151234	UbD
1	144916676	For	CTTTCAGCCACTGGAGG	NO	NO	NO	Yes_rs1698683_0.294959405151234	UbD
1	144916493	For	GTCCTCCCAATCTGAAG	NO	NO	Yes	NO	UbD
1	144916486	For	GGTCACAGTCCTCCCAA	NO	NO	Yes	NO	UbD
1	144892149	Rev	AACTGAAACGCACCACA	NO	NO	Yes	NO	UbD
1	144892149	For	TGTGGTGCGTTTCAGTT	NO	NO	Yes	NO	UbD
1	144892089	For	CCAAGCAGGTCCTCCGC	NO	NO	Yes	NO	UbD
1	144886409	Rev	TTCTCATTTAAAACTGC	NO	NO	Yes	NO	UbD
1	144877390	Rev	CAGCTTGGG	NO	NO	Yes	NO	bD
1	144877366	Rev	CTTAGAGATGTTCTCAT	NO	NO	Yes	NO	UbD
1	144874088	Rev	ACATAAGGCCAGGATTC	NO	NO	Yes	NO	UbD
1	144873818	For	GAAGAAAAATCAAACAG	NO	NO	Yes	NO	UbD
1	144873794	For	CTAACAGTCCAAACCTC	NO	NO	Yes	NO	UbD
1	144866509	For	TAAGGAGCGAGTGAAGT	NO	NO	Yes	NO	UbD
1	144864454	Rev	AGCATGCTCTTCCCACC	NO	NO	Yes	NO	UbD
1	144854380	Rev	ACCAGGAGC	NO	NO	Yes	NO	bD
1	144828404	Rev	TTTTTTTTT	NO	NO	Yes	NO	bD
1	120381687	For	TTATATGGAGCACCTAG	NO	NO	Yes	NO	UbD
1	120381686	For	ATTATATGGAGCACCTA	NO	NO	Yes	NO	UbD
1	118501941	Rev	TAATCAAGACCAGCCAT	NO	NO	NO	Yes_NA_0.291746083023744	UbD
1	118501941	For	ATGGCTGGT	NO	NO	NO	Yes_NA_0.291746083023744	Ub
1	117487710	Rev	TCACCTGATGGCTGGAT	Yes	Yes_TP	Yes	NO	UbD
1	111970431	Rev	GTGTGTGTA	NO	NO	Yes	NO	Ub
1	111772328	For	AAAAAAAAG	NO	NO	Yes	Yes_NA_2.63047138047138e-05	Ub
1	111436857	For	CCCTGTACTCGTGCAAA	Yes	NO	Yes	NO	UbD
1	110716511	For	GGCGCTGCCGACGTGCT	NO	NO	Yes	Yes_rs12747833_0.25379392971246	UbD
1	110211835	For	TCTCCTCTTTGCCCTTG	Yes	NO	Yes	NO	UbD
1	100598866	Rev	CACCTCCGCGGCTCCAG	Yes	Yes_TP	Yes	NO	UbD
1	89449483	Rev	CCAGGAAAGCTCTTCAT	NO	NO	Yes	Yes_rs74100106_0.245906267645398	UbD
1	89449483	For	ATGAAGAGCTTTCCTGG	NO	NO	Yes	Yes_rs74100106_0.245906267645398	UbD
1	89449434	Rev	CTCTTGAAA	NO	Yes_FP	Yes	Yes_rs2893084_0.283000495715601	Ub
1	89449434	For	AAATACTGTTTCAAGAG	NO	Yes_FP	Yes	Yes_rs2893084_0.283000495715601	UbD
1	87045902	Rev	GGAGTAGGTGTAGGATC	NO	Yes	NO	Yes_87045902_rs1932809_0.001746053918	UbD
1	87045902	For	GATCCTACACCTACTCC	NO	Yes	NO	Yes_87045902_rs1932809_0.001746053918	UbD
1	87045896	Rev	GGAGTAGGTGTAGGATC	Yes	Yes	Yes	NO	UbD
1	87045896	For	GATCCTACACCTACTCC	Yes	Yes	Yes	NO	UbD
1	85331808	Rev	TGTTCACTTCTCCACAA	Yes	Yes_TP	Yes	NO	UbD
1	66731886	Rev	GAGACATACTTTTGAGA	Yes	Yes_TP	Yes	Yes_rs6588193_0.274161341853035	UbD
1	62594677	Rev	GACGCTCAATCTTTTTT	Yes	NO	Yes	Yes_rs2481665_0.186501597444089	UbD
1	62061245	For	TTTTTTTTC	NO	NO	NO	NO	Ub
1	62061245	Rev	GAAAAAAAA	NO	NO	NO	NO	bD
1	60228310	Rev	CAAATGACAAGTCTTTA	Yes	Yes_TP	Yes	NO	UbD
1	57221553	Rev	TTATCTTTCAAAAGAAA	Yes	Yes_TP	Yes	NO	UbD
1	57221553	For	TTTCTTTTGAAAGATAA	Yes	Yes_TP	Yes	NO	UbD
1	55614036	For	AATAAACATAAAGATAG	Yes	NO	Yes	NO	UbD
1	53153432	Rev	GCTACACAAAGAACAGC	NO	NO	NO	Yes_rs443751_0.293063875504623	UbD
1	53153432	For	GCTGTTCTTTGTGTAGC	NO	NO	NO	Yes_rs443751_0.293063875504623	UbD
1	50956212	Rev	GTTGTATTT	Yes	Yes_TP	Yes	NO	Ub
1	50956212	For	TCAGTATCAAATACAAC	Yes	Yes_TP	Yes	NO	UbD
1	47153855	For	TCACACAGC	NO	NO	NO	Yes_rs11211340_0.272164536741214	bD
1	47014821	Rev	AAAAAAAAG	Yes	Yes	Yes	Yes_NA_2.63047138047138e-05	Ub
1	47014821	For	GAAAAAAGA	Yes	Yes	Yes	Yes_NA_2.63047138047138e-05	bD
1	43786721	Rev	AAAAAAAAA	NO	NO	Yes	NO	Ub
1	43393245	For	TGCGTGCGGGTGAGTAT	NO	Yes_TP	Yes	NO	UbD
1	37319270	For	TCCGTCCGCAAGCCACT	NO	NO	NO	Yes_NA_0.0483945165148588	UbD
1	36282417	For	AAAAAAAAA	NO	NO	NO	NO	Ub
1	34286189	For	TTTTTTTTT	NO	Yes	NO	Yes_rs3843294_0.103686594202899	bD
1	34286189	Rev	GTGTGCTGT	NO	Yes_TP	NO	Yes_rs3843294_0.103686594202899	Ub
1	33478920	Rev	ACTCAAACCACCCCACT	NO	NO	Yes	Yes_rs111261425_0.130386740331492	UbD
1	33478920	For	AGTGGGGTGGTTTGAGT	NO	NO	Yes	Yes_rs111261425_0.130386740331492	UbD
1	33478900	Rev	AGAGTACTACAGGAAAC	NO	Yes_FP	Yes	Yes_rs113711467_0.00964100173171706	UbD
1	33478900	For	GTTTCCTGTAGTACTCT	NO	Yes_FP	Yes	Yes_rs113711467_0.00964100173171706	UbD
1	33476404	Rev	AGACTGCAACACTGCTC	NO	NO	NO	Yes_rs76230073_0.171189160603242	UbD
1	33476404	For	GAGCAGTGTTGCAGTCT	NO	NO	NO	Yes_rs76230073_0.171189160603242	UbD
1	33476396	Rev	ACACTGCTCATCACCCC	NO	NO	NO	Yes_rs79824855_0.175618413961009	UbD
1	33476396	For	GGGGTGATGAGCAGTGT	NO	NO	NO	Yes_rs79824855_0.175618413961009	UbD
1	33476387	Rev	ATCACCCCGCGGCGTGA	NO	NO	NO	Yes_rs201842558_0.209405622333483	UbD
1	33476387	For	TCACGCCGCGGGGTGAT	NO	NO	NO	Yes_rs201842558_0.209405622333483	UbD
1	33476385	Rev	CACCCCGCGGCGTGATC	NO	Yes_TP	NO	Yes_rs76881767_0.216441804269029	UbD
1	33476385	For	GATCACGCCGCGGGGTG	NO	Yes_TP	NO	Yes_rs76881767_0.216441804269029	UbD
1	32658062	Rev	CTCCTCCCACCCCCACC	NO	Yes_TP	Yes	NO	UbD
1	32044641	For	CAGTTATAGTTTAATTT	NO	NO	NO	NO	UbD
1	31478638	For	GGAAAAAAAGAGAGAGA	Yes	NO	Yes	NO	UbD
1	31406286	Rev	GTGTGTGTATGTGTGTG	NO	NO	Yes	NO	UbD
1	29320169	For	AAAAAAAAG	NO	NO	NO	NO	Ub
1	29320169	Rev	GAAAAAAGA	NO	NO	NO	NO	bD
1	27995605	Rev	AGGTAAGGGTGCAGGTA	NO	NO	NO	NO	UbD
1	27995605	For	TACCTGCACCCTTACCT	NO	NO	NO	NO	UbD
1	27995593	Rev	AGGTAAGGATGCGGGTA	NO	NO	NO	NO	UbD
1	27995593	For	TACCCGCATCCTTACCT	NO	NO	NO	NO	UbD
1	27995589	Rev	AAGGATGCGGGTAAGGA	NO	NO	NO	NO	UbD
1	27995589	For	TCCTTACCCGCATCCTT	NO	NO	NO	NO	UbD
1	27995566	For	CCTTACCCGCATCCTTA	NO	NO	NO	NO	UbD
1	27995565	For	TCCTTACCCGCATCCTT	NO	NO	NO	NO	UbD
1	27995565	Rev	AAGGATGCGGGTAAGGA	NO	NO	NO	NO	UbD
1	27995545	For	TCCTTACCT	NO	NO	NO	NO	bD
1	27995545	Rev	AGGTAAGGA	NO	NO	NO	NO	Ub
11	48373833	For	ACGGAAAAGAGTACATG	NO	NO	Yes	NO	UbD
11	48373815	Rev	CTATTTGTTCATTATAG	NO	NO	Yes	NO	UbD
11	48373815	For	CTATAATGAACAAATAG	NO	NO	Yes	NO	UbD
11	48367469	Rev	CTATGACCACTATGTGG	NO	NO	Yes	NO	UbD
11	48366971	For	ATTTTTGAGGGTGTAGA	NO	NO	Yes	NO	UbD
11	48347539	Rev	CAAAAATAA	NO	NO	Yes	NO	bD
11	48347140	Rev	ACATACGTATTGGTGCA	NO	Yes_FP	Yes	NO	UbD
11	48347140	For	TGCACCAATACGTATGT	NO	Yes_FP	Yes	NO	UbD
11	48346961	Rev	GGTAGTATTGTGCAGGG	NO	Yes_FP	Yes	NO	UbD
11	48346961	For	CCCTGCACAATACTACC	NO	Yes_FP	Yes	NO	UbD
11	48346551	Rev	CTGAGAATGCAGGTGCA	NO	Yes_FP	Yes	Yes_rs77470587_0.243449801619951	UbD
11	48346551	For	TGCACCTGCATTCTCAG	NO	Yes_FP	Yes	Yes_rs77470587_0.243449801619951	UbD
11	48346547	Rev	GAATGCAGG	NO	Yes_FP	Yes	Yes_rs79042268_0.232394493587308	Ub
11	48346547	For	ATACTGCACCTGCATTC	NO	Yes_FP	Yes	Yes_rs79042268_0.232394493587308	UbD
11	48346541	For	TAGGCAATACTGCACCT	NO	Yes_FP	Yes	Yes_rs75498992_0.222422008450957	UbD
11	48346535	For	TCTTTATAGGCAATACT	NO	Yes_FP	Yes	Yes_rs76964780_0.198116104327722	UbD
11	48346523	For	TGTTTCTCCTTGTCTTT	NO	Yes_FP	Yes	Yes_rs72911451_0.248489074848907	UbD
11	48346513	For	CTATTACTTATGTTTCT	NO	Yes_FP	Yes	Yes_rs78206553_0.268059367693402	UbD
11	45937968	Rev	TAGTCTGTG	Yes	NO	Yes	NO	bD
11	34982146	Rev	AAGACTGAATAAGCTCT	NO	NO	Yes	NO	UbD
11	34188981	Rev	TTAACTCTGGTTCACCA	NO	NO	NO	Yes_rs7120870_0.212659744408946	UbD
11	26718732	Rev	TGTGGCTTG	NO	Yes_FP	NO	Yes_rs72883299_0.256771275091886	Ub
11	26718732	For	TGAAGGCACAAGCCACA	NO	Yes_FP	NO	Yes_rs72883299_0.256771275091886	UbD
11	22239663	For	ACACACACA	NO	NO	NO	NO	bD
11	18524208	Rev	TATGTCAAGCTTAACTT	NO	NO	NO	Yes_rs2658552_0.270966453674121	UbD
11	18267373	For	AACTTCTCCACAAGGAG	NO	NO	Yes	NO	UbD
11	17131918	For	ACGGCAGGTACACTTTA	NO	NO	Yes	NO	UbD
11	10831340	Rev	CAAAAAAAA	Yes	NO	Yes	Yes_rs72430557_0.277005712336943	bD
11	10823582	For	AAATATTAC	NO	NO	NO	Yes_rs71034786_0.29174319951019	bD
11	6567895	Rev	AGTAGGGCAGCCTCCTC	NO	Yes_TP	Yes	NO	UbD
11	6567895	For	GAGGAGGCTGCCCTACT	NO	Yes_TP	Yes	NO	UbD
11	6499215	For	AATGCCCAAATTTCCAC	NO	NO	Yes	Yes_rs2303493_0.294329073482428	UbD
11	5618298	Rev	ATGTCTCTA	NO	NO	Yes	NO	bD
11	5269586	For	GGGCAGTGAGCTCAGTG	NO	Yes_TP	Yes	Yes_rs62755960_8.31020326757192e-06	UbD
11	5269583	For	CATGGGCAGTGAGCTCA	NO	NO	NO	Yes_rs5009539_0.000191644308164048	UbD
11	1651613	For	TGTAAGCCTTACTGCTG	NO	Yes_FP	NO	Yes_rs76143925_0.132516417438088	UbD
11	1093569	For	ACCACCACAACTACGGT	NO	Yes_TP	Yes	Yes_rs72842460_0.0584891234647068	UbD
11	1093452	Rev	GTGGGTGTCGGGGTTGG	NO	NO	Yes	Yes_rs34136803_0.298812175204157	UbD
11	1093452	For	CCAACCCCGACACCCAC	NO	NO	Yes	Yes_rs34136803_0.298812175204157	UbD
11	1093412	Rev	TGGGTGTCGTGGTTGGG	NO	NO	Yes	Yes_rs113330607_0.2575	UbD
11	1093412	For	CCCAACCACGACACCCA	NO	NO	Yes	Yes_rs113330607_0.2575	UbD
11	1093411	Rev	TGGGTGTCGTGGTTGGG	NO	NO	NO	NO	UbD
11	1093411	For	CCCAACCACGACACCCA	NO	NO	NO	NO	UbD
11	1093354	Rev	GGTGGTGGAGATGGGTG	NO	NO	NO	Yes_rs56290335_0.00385811355522285	UbD
11	1093354	For	CACCCATCTCCACCACC	NO	NO	NO	Yes_rs56290335_0.00385811355522285	UbD
11	1093158	Rev	GTCACCGTAGTGGTGGT	NO	NO	NO	Yes_rs202045556_0.172145328719723	UbD
11	1093158	For	ACCACCACTACGGTGAC	NO	NO	NO	Yes_rs202045556_0.172145328719723	UbD
11	1093136	Rev	TGGGTGTCGTGGTTGGG	NO	NO	NO	Yes_rs374924328_0.00923295454545455	UbD
11	1093136	For	CCCAACCACGACACCCA	NO	NO	NO	Yes_rs374924328_0.00923295454545455	UbD
11	1093135	Rev	TGGGTGTCTGGTTGGGG	NO	NO	NO	Yes_rs374924328_0.00923295454545455	UbD
11	1093135	For	CCCCAACCCGACACCCA	NO	NO	NO	Yes_rs56080332_0.00662251655629139	UbD
11	1092807	Rev	GTGGTGGTG	NO	NO	NO	Yes_NA_0.000203334688897926	Ub
11	1092807	For	CACCACCAC	NO	NO	NO	Yes_NA_0.000203334688897926	bD
11	1092804	Rev	GTGGTGGTG	NO	NO	NO	NO	Ub
11	1092804	For	CACCACCAC	NO	NO	NO	NO	bD
11	1092492	Rev	GGGCTGGGGGTGGTGGT	NO	NO	NO	Yes_rs111844333_0.00719564371839751	UbD
11	1092459	Rev	GTGGTGGTGGTTGGAGG	NO	Yes	NO	Yes_rs374506032_0.000996346728661574	UbD
11	1092459	For	CACCACCAC	NO	Yes	NO	Yes_rs374506032_0.000996346728661574	bD
11	1092457	Rev	GGTGGTGGTTGGAGGGC	NO	NO	NO	Yes_rs145610697_5.2565180824222e-05	UbD
11	1092457	For	CACCACCAC	NO	NO	NO	Yes_rs374506032_0.000996346728661574	bD
11	1092444	Rev	GGGCTGGGGGTGGTGGT	NO	NO	NO	Yes_rs79619079_0.000854052002277472	UbD
11	1092420	Rev	AGGGTGGTCGTGCTGGT	NO	NO	NO	NO	UbD
11	1018385	Rev	ATGGCCACATCTGCCTC	NO	NO	Yes	Yes_rs7396380_0.191608986035216	UbD
11	1018385	For	GAGGCAGATGTGGCCAT	NO	NO	Yes	Yes_rs7396380_0.191608986035216	UbD
11	1018366	Rev	ACCACTCAGCGCCAACA	NO	Yes_FP	Yes	Yes_rs76741048_0.142628009553863	UbD
11	1018366	For	TGTTGGCGCTGAGTGGT	NO	Yes_FP	Yes	Yes_rs76741048_0.142628009553863	UbD
11	1018355	Rev	CCAACAGGTACCATTCC	NO	NO	Yes	Yes_rs7396697_0.241086881273797	UbD
11	1018355	For	GGAATGGTACCTGTTGG	NO	NO	Yes	Yes_rs7396697_0.241086881273797	UbD
11	1018263	Rev	CCAAGCCCACAGCTCAT	NO	Yes_FP	Yes	Yes_rs79680044_0.241363580323018	UbD
11	1018263	For	ATGAGCTGTGGGCTTGG	NO	Yes_FP	Yes	Yes_rs79680044_0.241363580323018	UbD
11	1018262	Rev	CAAGCCCACAGCTCATT	NO	Yes_FP	Yes	NO	UbD
11	1018262	For	AATGAGCTGTGGGCTTG	NO	Yes_FP	Yes	NO	UbD
11	1018245	Rev	CAGCACAAACAAAACAC	NO	Yes_FP	Yes	NO	UbD
11	1018245	For	GTGTTTTGTTTGTGCTG	NO	Yes_FP	Yes	NO	UbD
11	1018175	Rev	ACTTCTACTACCACGAT	NO	NO	Yes	Yes_rs12418172_0.196287462407964	UbD
11	1018175	For	ATCGTGGTAGTAGAAGT	NO	NO	Yes	Yes_rs12418172_0.196287462407964	UbD
11	1018144	Rev	CTAGTACACGCACCAGA	NO	Yes_FP	Yes	Yes_rs12807084_0.282015140837065	UbD
11	1018144	For	TCTGGTGCGTGTACTAG	NO	Yes_FP	Yes	Yes_rs12807084_0.282015140837065	UbD
11	1018138	Rev	CACGCACCAGAACCCCT	NO	Yes_FP	Yes	Yes_rs71454075_0.29656933953128	UbD
11	1018138	For	AGGGGTTCTGGTGCGTG	NO	Yes_FP	Yes	Yes_rs71454075_0.29656933953128	UbD
11	1018012	Rev	CATCCTTCAAGACCACC	NO	Yes_FP	NO	Yes_rs10751676_0.267247185605696	UbD
11	1018012	For	GGTGGTCTTGAAGGATG	NO	Yes_FP	NO	Yes_rs10751676_0.267247185605696	UbD
11	1017883	Rev	CACAGATGGCCACTTCT	NO	NO	Yes	Yes_rs79644784_0.112084203760244	UbD
11	1017883	For	AGAAGTGGCCATCTGTG	NO	NO	Yes	Yes_rs79644784_0.112084203760244	UbD
11	1017797	Rev	CACACAGCGCCAACAAT	NO	NO	NO	Yes_rs76800954_0.0271230019598324	UbD
11	1017797	For	ATTGTTGGCGCTGTGTG	NO	NO	NO	Yes_rs76800954_0.0271230019598324	UbD
11	1017793	Rev	CAGCGCCAACAATGACG	NO	Yes_FP	Yes	Yes_rs77940304_0.0263719300679895	UbD
11	1017793	For	CGTCATTGTTGGCGCTG	NO	Yes_FP	Yes	Yes_rs77940304_0.0263719300679895	UbD
11	1017684	Rev	TGCTGAAGCCACCTCAA	NO	NO	NO	Yes_rs112886536_0.000108158477128642	UbD
11	1017684	For	TTGAGGTGGCTTCAGCA	NO	NO	NO	Yes_rs112886536_0.000108158477128642	UbD
11	1017337	Rev	CCTTCCAGACCACCACT	NO	NO	Yes	NO	UbD
11	1017337	For	AGTGGTGGTCTGGAAGG	NO	NO	Yes	NO	UbD
11	1017239	Rev	ACTCACACGGTCATCAT	NO	NO	Yes	NO	UbD
11	1017239	For	ATGATGACCGTGTGAGT	NO	NO	Yes	NO	UbD
11	1017186	Rev	CCACTCAATGCCAACAG	NO	NO	Yes	Yes_rs74579726_0.0257486313567707	UbD
11	1017186	For	CTGTTGGCATTGAGTGG	NO	NO	Yes	Yes_rs74579726_0.0257486313567707	UbD

1	27995530	For	CCTTACCCG	NO	NO	NO	Yes_rs145092707_0.289536741214057	Ub
1	24671485	Rev	AGGTGTGGAGGGGGCAC	NO	NO	NO	NO	UbD
1	24671482	Rev	TGTGGAGGGGGCACAGG	NO	NO	NO	NO	UbD
1	24106510	Rev	ATGATACTA	Yes	NO	Yes	NO	bD
1	22310411	Rev	ACAAGGTCTGGTCTTGG	NO	NO	Yes	NO	UbD
1	19447752	Rev	GTTCTGGACAAGCCACA	NO	Yes_FP	NO	Yes_rs78347051_0.296629900177527	UbD
1	19447752	For	TGTGGCTTGTCCAGAAC	NO	Yes_FP	NO	Yes_rs78347051_0.296629900177527	UbD
1	19208145	For	GTGAGCAGGTGAACGTC	Yes	Yes_TP	Yes	NO	UbD
1	16916702	Rev	TGCCTTGAA	NO	NO	Yes	NO	Ub
1	16383581	Rev	CCCAGGATG	NO	NO	NO	NO	bD
1	16382827	For	CGCCCCTCTTCCTGGCT	NO	NO	NO	NO	UbD
1	16376238	For	AAGGCATTCCCCCCAAG	NO	NO	Yes	NO	UbD
1	16376237	For	AAAAGGCATTCCCCCCA	NO	NO	NO	NO	UbD
1	16376236	For	CAAAAGGCATTCCCCCC	NO	NO	NO	NO	UbD
1	16376235	For	CCAAAAGGCATTCCCCC	NO	NO	NO	NO	UbD
1	16376233	For	GTCCAAAAGGCATTCCC	NO	NO	NO	Yes_NA_0.000213447171824973	UbD
1	16376231	For	AAAAGGCATTCCCCCCA	NO	NO	NO	Yes_rs45470508_0.000798722044728434	UbD
1	16376230	For	GAGTCCAAAAGGCATTC	NO	Yes	Yes	NO	UbD
1	16375063	Rev	AGGGAGCTGCAAAGACT	NO	Yes_TP	Yes	NO	UbD
1	16375063	For	AGTCTTTGCAGCTCCCT	NO	Yes_TP	Yes	NO	UbD
1	16373199	Rev	GATCTGAGAAGGTTTTG	NO	NO	NO	NO	UbD
1	16359051	Rev	CGGGGGCTGCGTGTGTC	NO	NO	Yes	Yes_NA_3.28320966576926e-05	UbD
1	15873410	Rev	GAAAAGAAA	NO	NO	NO	Yes_NA_6.90099374309901e-05	bD
1	15873410	For	AAAAAAAAG	NO	NO	NO	Yes_NA_6.90099374309901e-05	Ub
1	15873408	Rev	GAAAAGAAA	NO	NO	NO	NO	bD
1	15873408	Rev	AAAAAAAAG	NO	NO	NO	NO	Ub
1	15873405	Rev	GAAAAGAAA	NO	NO	NO	Yes_NA_6.4648405031493e-05	bD
1	15438825	For	TGTGTGTGT	NO	NO	Yes	NO	bD
1	13183966	Rev	CCGTAGAAC	NO	NO	Yes	Yes_rs80025711_0.00798722044728434	bD
1	13183532	Rev	GGACTATGGCTTTCAAC	NO	NO	Yes	NO	UbD
1	13183532	For	GTTGAAAGCCATAGTCC	NO	NO	Yes	NO	UbD
1	13183248	Rev	AGGAACAGAGCAAACAA	NO	NO	Yes	NO	UbD
1	13183248	For	TTGTTTGCTCTGTTCCT	NO	NO	Yes	NO	UbD
1	13183237	Rev	AAACAAGAGGTAGAGGT	NO	NO	Yes	NO	UbD
1	13183237	For	ACCTCTACCTCTTGTTT	NO	NO	Yes	NO	UbD
1	13182899	For	CTGAAGATACTAAGGGA	NO	NO	Yes	Yes_rs28537666_0.167931309904153	UbD
1	12908243	Rev	GCAGTCAGTTTCTCCCC	NO	NO	Yes	NO	UbD
1	12908236	Rev	GTTTCTCCCCGTAGAAC	NO	NO	Yes	NO	UbD
1	12907802	Rev	GGACTATGGCTTTCAAC	NO	Yes	Yes	NO	UbD
1	12907802	For	GTTGAAAGCCATAGTCC	NO	Yes	Yes	NO	UbD
1	12907282	Rev	GACAGCACCAATGGCCA	NO	NO	Yes	NO	UbD
1	12907282	For	TGGCCATTGGTGCTGTC	NO	NO	Yes	NO	UbD
1	12907212	For	AGACAAGCTCCTAGGTA	NO	Yes_FP	Yes	Yes_rs5027838_0.296232775445029	UbD
1	12907199	For	AAAATTTGTTGTTAGAC	NO	NO	Yes	NO	UbD
1	12907169	For	CTGAAGATACTAAGGGA	NO	NO	Yes	NO	UbD
1	11917620	For	AATACATTAAAAAAATG	NO	Yes_TP	Yes	Yes_rs35458601_0.293730031948882	UbD
1	11133956	Rev	AAAAAAAAA	NO	NO	NO	Yes_rs10082000_0.00135554417567853	Ub
1	11133956	For	AAGTCTCTT	NO	NO	NO	Yes_rs10082000_0.00135554417567853	bD
1	9009214	For	GGTGCCTCCGTGGGTCC	Yes	NO	Yes	Yes_rs2274326_0.270567092651757	UbD
1	3418565	Rev	ACCAGGAGC	NO	NO	Yes	NO	bD
1	3395973	For	CCCCACCACCCCCATCT	NO	NO	Yes	NO	UbD
1	1853615	For	ATTGGAATCTGGCAGAG	Yes	NO	Yes	NO	UbD
1	1844046	Rev	GTTATTCTC	NO	NO	NO	Yes_rs2474460_0.296924920127796	bD
1	1684347	For	GACCTAGCCCTCCTCCT	NO	Yes_FP	NO	Yes_NA_0.0270626357582879	UbD
1	1670570	Rev	GATGGGAGACCGGGGGT	NO	NO	Yes	NO	UbD
1	1654038	For	GGAAGTACA	NO	Yes_TP	Yes	Yes_rs61777494_0.218100010777023	bD
1	1651003	Rev	TTCAGCTTCGTTCCTAT	NO	NO	Yes	Yes_rs76941027_0.000399361022364217	UbD
1	1650996	Rev	TCGTTCCTATCTGAATC	NO	NO	Yes	Yes_rs74223875_0.000199680511182109	UbD
1	1650967	Rev	GGGTTGAAAAAACCTCA	NO	NO	Yes	NO	UbD
1	1650960	Rev	AAAAACCTCATAATAGC	NO	NO	Yes	NO	UbD
1	1650942	Rev	TGATGCTTTTCCGCACT	NO	NO	Yes	Yes_1650942_rs75972011_1.773364071643	UbD
1	1650940	Rev	ATGCTTTTCCGCACTTT	NO	NO	NO	Yes_rs74345479_0.000135291368410695	UbD
1	1650939	Rev	TGCTTTTCCGCACTTTC	NO	NO	NO	Yes_rs201741924_0.00819672131147541	UbD
1	1650657	For	TTCAGCTACAGTTTGCT	NO	NO	NO	NO	UbD
1	1650642	For	TGACACCATTATGCTTT	NO	NO	NO	NO_1650642	UbD
1	1650641	For	GTGACACCATTATGCTT	NO	NO	NO	NO	UbD
1	1650640	For	TGTGACACCATTATGCT	NO	NO	NO	NO	UbD
1	1648058	Rev	CTGGTTTTG	NO	NO	Yes	NO	bD
1	1648054	Rev	CAGGCTGGTTTTGAACT	NO	NO	Yes	NO_1648054	UbD
1	1648053	Rev	AGGCTGGTTTTGAACTC	NO	NO	Yes	NO	UbD
1	1648018	Rev	CCCGCCTCGGCCTCCCA	NO	NO	Yes	NO	UbD
1	1648002	Rev	AAAGTGCTGGGATTACA	NO	NO	Yes	NO	UbD
1	1647990	Rev	TTACAGGCGTAAGCCAC	NO	NO	Yes	NO	UbD
1	1647983	Rev	CGTAAGCCACCGTGCCC	NO	NO	Yes	NO	UbD
1	1647971	Rev	TGCCCGGCCTCGTGAAA	NO	NO	Yes	NO	UbD
1	1647730	For	GCTGTGACAGGACACAC	NO	NO	Yes	NO_1647730	UbD
1	1647729	For	TGCTGTGACAGGACACA	NO	NO	Yes	NO	UbD
1	1647726	For	CATTGCTGTGACAGGAC	NO	NO	Yes	NO_1647726	UbD
1	1647725	For	TCATTGCTGTGACAGGA	NO	NO	Yes	Yes_rs368680882_0.00319488817891374	UbD
1	1647722	For	TCTTCATTGCTGTGACA	NO	NO	Yes	NO	UbD
1	1647689	For	CTGCAACAAATGTGACT	NO	NO	NO	NO	UbD
1	1581926	Rev	TCCTGTAACTATCTCTC	NO	NO	Yes	NO	UbD
1	1581926	For	GAGAGATAGTTACAGGA	NO	NO	Yes	NO	UbD
1	1581759	Rev	GAGGCATACGGTGATCC	NO	NO	Yes	NO	UbD
1	1581759	For	GGATCACCGTATGCCTC	NO	NO	Yes	NO	UbD
1	1574076	Rev	TCCACCCCCGGCCCAGT	NO	NO	Yes	NO	UbD
1	1574019	Rev	CGTGCCTGTGGACGCAG	NO	NO	Yes	Yes_rs9442412_0.21685303514377	UbD
1	900285	For	TTGTGGTGCAGCCCCTC	NO	NO	Yes	NO	UbD
1	889158	For	TGGGCCACGAACCTTGA	Yes	Yes_TP	Yes	NO	UbD
1	884091	Rev	GCAGCCAGG	NO	NO	Yes	NO	bD
2	242121959	For	ACCTGCTCGCCATCTTC	Yes	NO	Yes	Yes_rs34186191_0.253394568690096	UbD
2	240098010	For	CGGCCTTCT	NO	NO	NO	Yes_rs573094758_0.262779552715655	Ub
2	240098002	For	CCTCTGCCCGGCCTTCT	NO	NO	NO	Yes_rs138604304_0.262779552715655	UbD
2	238261112	For	CCTGCTTGGCAATGTGC	Yes	NO	Yes	NO	UbD
2	234590974	Rev	TAATTTTCGGTCATTAA	NO	Yes_TP	Yes	NO	UbD
2	234590974	For	TTAATGACCGAAAATTA	NO	Yes_TP	Yes	NO	UbD
11	1017045	Rev	ATCCCTACCTTACCACA	NO	NO	Yes	Yes_rs77885750_0.161663601331231	UbD
11	1017045	For	TGTGGTAAGGTAGGGAT	NO	NO	Yes	Yes_rs77885750_0.161663601331231	UbD
11	1016976	Rev	CCCCAAACACACCAGTA	NO	NO	Yes	Yes_rs78728217_0.119136928611618	UbD
11	1016976	For	TACTGGTGTGTTTGGGG	NO	NO	Yes	Yes_rs78728217_0.119136928611618	UbD
11	1016963	Rev	AGTACAGGCACCAGAAC	NO	NO	Yes	Yes_rs76273000_0.0777383887279336	UbD
11	1016963	For	GTTCTGGTGCCTGTACT	NO	NO	Yes	Yes_rs76273000_0.0777383887279336	UbD
11	1016933	Rev	ACCACCTCGGCCAGCAG	NO	NO	Yes	Yes_rs199579978_0.11494528453753	UbD
11	1016933	For	CTGCTGGCCGAGGTGGT	NO	NO	Yes	Yes_rs199579978_0.11494528453753	UbD
11	1016722	Rev	TCATCATCACTACCCAC	NO	Yes_FP	Yes	Yes_rs144616203_0.100191754554171	UbD
11	1016722	For	GTGGGTAGTGATGATGA	NO	Yes_FP	Yes	Yes_rs144616203_0.100191754554171	UbD
11	1016618	Rev	CACACAGCCCCACCAAT	NO	NO	NO	Yes_rs151061093_0.000505250643946899	UbD
11	1016618	For	ATTGGTGGGGCTGTGTG	NO	NO	NO	Yes_rs151061093_0.000505250643946899	UbD
11	1016614	Rev	CAGCCCCACCAATGACA	NO	Yes_FP	NO	Yes_rs148793744_0.000383063009193512	UbD
11	1016614	For	TGTCATTGGTGGGGCTG	NO	Yes_FP	NO	Yes_rs148793744_0.000383063009193512	UbD
11	1016606	Rev	CCAATGACAGTGACCAC	NO	NO	NO	Yes_rs148905794_0.00164996333414813	UbD
11	1016606	For	GTGGTCACTGTCATTGG	NO	NO	NO	Yes_rs148905794_0.00164996333414813	UbD
11	1016605	Rev	CAATGACAGTGACCACC	NO	NO	NO	Yes_rs147913342_0.00159659393294305	UbD
11	1016605	For	GGTGGTCACTGTCATTG	NO	NO	NO	Yes_rs147913342_0.00159659393294305	UbD
11	1016604	Rev	AATGACAGTGACCACCA	NO	NO	NO	Yes_rs148590913_0.00152597204419215	UbD
11	1016604	For	TGGTGGTCACTGTCATT	NO	NO	NO	Yes_rs148590913_0.00152597204419215	UbD
11	1016581	Rev	CCAGCCAAACCCACAGC	NO	NO	NO	Yes_rs115593063_0.00177705977382876	UbD
11	1016581	For	GCTGTGGGTTTGGCTGG	NO	NO	NO	Yes_rs115593063_0.00177705977382876	UbD
11	1016565	Rev	CTCATTCAGCACAGCTA	NO	NO	NO	Yes_rs369173915_4.99633602025182e-05	UbD
11	1016565	For	TAGCTGTGCTGAATGAG	NO	NO	NO	Yes_rs369173915_4.99633602025182e-05	UbD
11	1016558	Rev	AGCACAGCTACAGCCTC	NO	Yes	NO	Yes_rs114928002_0.000354693781035706	UbD
11	1016558	For	GAGGCTGTAGCTGTGCT	NO	Yes	NO	Yes_rs114928002_0.000354693781035706	UbD
11	1016556	Rev	CACAGCTACAGCCTCTT	NO	NO	NO	Yes_rs114271589_0.000276344878408254	UbD
11	1016556	For	AAGAGGCTGTAGCTGTG	NO	NO	NO	Yes_rs114271589_0.000276344878408254	UbD
11	1016554	Rev	CAGCTACAGCCTCTTCT	NO	NO	NO	Yes_rs376688775_0.000257510729613734	UbD
11	1016554	For	AGAAGAGGCTGTAGCTG	NO	NO	NO	Yes_rs376688775_0.000257510729613734	UbD
11	1016548	Rev	CAGCCTCTTCTTCCTTC	NO	NO	NO	Yes_rs140561174_0.000162654521795706	UbD
11	1016548	For	GAAGGAAGAAGAGGCTG	NO	NO	NO	Yes_rs140561174_0.000162654521795706	UbD
11	1016546	Rev	GCCTCTTCTTCCTTCAT	NO	NO	NO	Yes_rs145319819_0.000159235668789809	UbD
11	1016546	For	ATGAAGGAAGAAGAGGC	NO	NO	NO	Yes_rs145319819_0.000159235668789809	UbD
11	280816	For	CTACAAGTATTTCCGGG	Yes	Yes_TP	Yes	NO	UbD
11	244106	Rev	ACTCATTTTCAAAGTGA	Yes	Yes_TP	Yes	NO	UbD
11	244106	For	TCACTTTGAAAATGAGT	Yes	Yes_TP	Yes	NO	UbD
11	193863	Rev	CCATCTATACAGGACTT	NO	Yes_TP	Yes	NO	UbD
11	193863	For	AAGTCCTGTATAGATGG	NO	Yes_TP	Yes	NO	UbD
12	133291679	Rev	ATATGCACC	NO	NO	Yes	NO	bD
12	124181820	Rev	ATGCTGTCTTTGATATT	Yes	NO	Yes	Yes_rs7979528_0.272164536741214	UbD
12	124171658	Rev	ACCGCGTTGCAGGGCCT	Yes	NO	Yes	Yes_rs7309528_0.272164536741214	UbD
12	124158450	Rev	ATCAACCTC	Yes	NO	Yes	Yes_rs7953596_0.27176517571885	bD
12	124158080	For	TGATGTGTACTTTTATT	Yes	NO	Yes	Yes_rs7978447_0.271565495207668	UbD
12	123816007	Rev	ACATAACATATTCCTAC	Yes	NO	Yes	NO	UbD
12	122394921	For	TAGGTATTATATAGATG	Yes	NO	Yes	Yes_rs11043271_0.166333865814696	UbD
12	117718473	For	AGTCTGTGGTCTGGGGA	Yes	NO	Yes	NO	UbD
12	116445450	Rev	TTTTTTTTGTCTTTTAG	NO	NO	NO	Yes_rs200367922_0.0553402239448751	UbD
12	116445448	Rev	TTTTTTGTCTTTTAGAC	NO	NO	NO	Yes_rs199749418_0.0497326203208556	UbD
12	116412921	For	TAATGACAATAGCTGAG	Yes	NO	Yes	NO	UbD
12	109723079	For	TGTGTGTGT	NO	NO	NO	Yes_109723079_rs10774644_0.154293700	Ub
12	109723049	For	TGTGTGTGT	NO	NO	Yes	NO	Ub
12	80762163	Rev	TGGGGGGGGACATTTTC	Yes	NO	Yes	NO	UbD
12	66826527	Rev	TTCATCTTACTTCAGTT	Yes	NO	Yes	NO	UbD
12	66522654	For	AAGAGAAAGAAAAAGGA	NO	NO	NO	Yes_NA_0.0417669435602826	UbD
12	64712546	Rev	CAGATCACAACTCCATA	Yes	Yes_TP	Yes	NO	UbD
12	64712546	For	TATGGAGTTGTGATCTG	Yes	Yes_TP	Yes	NO	UbD
12	57703935	For	AGGGTGCAGTGGGCCCT	Yes	NO	Yes	Yes_rs7139302_0.297723642172524	UbD
12	57548168	Rev	AGAACCCCGCCTGCCAC	NO	NO	NO	Yes_rs199886321_0.128128664767968	UbD
12	57548168	For	GTGGCAGGCGGGGTTCT	NO	NO	NO	Yes_rs199886321_0.128128664767968	UbD
12	57493541	For	GGCTGGGGTGGCCTCAC	NO	NO	Yes	Yes_rs3024972_0.232859174964438	UbD
12	56295544	For	GGTGGAAGTAAGCCAGT	Yes	Yes_TP	Yes	NO	UbD
12	53693532	For	GGGACACCAATTCCTGC	NO	Yes_TP	Yes	Yes_NA_8.41665825000842e-06	UbD
12	53421746	Rev	AAAAAAAAAGGCAAAGT	Yes	Yes_TP	Yes	Yes_rs77925779_0.00116689053055742	UbD
12	53421746	For	ACTTTGCCTTTTTTTTT	Yes	Yes_TP	Yes	Yes_rs77925779_0.00116689053055742	UbD
12	53207606	Rev	GCTGGAGGCTTTGGCAC	NO	NO	NO	Yes_rs79164931_0.000475031173920789	UbD
12	53207606	For	GTGCCAAAGCCTCCAGC	NO	NO	NO	Yes_rs79164931_0.000475031173920789	UbD
12	53207603	Rev	GGAGGCTTTGGCACTGG	NO	NO	NO	Yes_rs7135148_0.00067710537452391	UbD
12	53207603	For	CCAGTGCCAAAGCCTCC	NO	NO	NO	Yes_rs7135148_0.00067710537452391	UbD
12	50829263	For	ATTTTGTTTTTTCAGAG	Yes	Yes_TP	Yes	NO	UbD
12	50745863	Rev	CAGCAGGCGCAGGAACT	NO	NO	Yes	Yes_rs12317337_0.0564566437928883	UbD
12	50745863	For	AGTTCCTGCGCCTGCTG	NO	NO	Yes	Yes_rs12317337_0.0564566437928883	UbD
12	50745791	Rev	CAGCAGGCGCAGGCTCA	NO	NO	Yes	Yes_rs144970699_0.00173778721760958	UbD
12	50745791	For	TGAGCCTGCGCCTGCTG	NO	NO	Yes	Yes_rs144970699_0.00173778721760958	UbD
12	50745785	Rev	GCGCAGGCTCAGGGGAT	NO	NO	Yes	Yes_NA_0.269412028725314	UbD
12	50745785	For	ATCCCCTGAGCCTGCGC	NO	NO	Yes	Yes_NA_0.269412028725314	UbD
12	50745783	Rev	GCAGGCTCAGGGGATCC	NO	NO	Yes	Yes_rs373687267_0.275175379045033	UbD
12	50745783	For	GGATCCCCTGAGCCTGC	NO	NO	Yes	Yes_rs373687267_0.275175379045033	UbD
12	50745779	Rev	GCTCAGGGGATCCCTCT	NO	Yes_TP	Yes	Yes_NA_0.275592311130979	UbD
12	50745779	For	AGAGGGATCCCCTGAGC	NO	Yes_TP	Yes	Yes_NA_0.275592311130979	UbD
12	32948971	For	CACATTCACAACCGGAT	Yes	NO	Yes	Yes_rs61927769_0.249400958466454	UbD
12	32948969	For	GCACATTCACAACCGGA	Yes	NO	Yes	Yes_rs71447623_0.249400958466454	UbD
12	31298250	For	CTGAGTGTAAGTTTCCC	NO	NO	Yes	NO	UbD
12	31145010	Rev	TGGGCACAGAAGCTATG	NO	Yes_TP	Yes	NO	UbD
12	27849795	Rev	TGCTTGTTGGATCCCCA	NO	Yes_TP	Yes	NO	UbD
12	27800576	For	ATCATTCCTTTTTTTGT	NO	NO	Yes	NO	UbD
12	26834804	Rev	AAAGGAGGTAACTCTTT	Yes	Yes_TP	Yes	NO	UbD
12	26834804	For	AAAGAGTTACCTCCTTT	Yes	Yes_TP	Yes	NO	UbD
12	11506785	Rev	CCACAAGGGGACAAGTC	NO	NO	NO	Yes_rs201935302_0.0166045031480975	UbD
12	11506785	For	GACTTGTCCCCTTGTGG	NO	NO	NO	Yes_rs201935302_0.0166045031480975	UbD
12	11506774	Rev	CAAGTCCCGAAGTCCCC	NO	NO	NO	Yes_rs202083397_0.0207294990083642	UbD
12	11506774	For	GGGGACTTCGGGACTTG	NO	NO	NO	Yes_rs202083397_0.0207294990083642	UbD
12	11506749	Rev	CCAGGAAAACCACAAGG	NO	NO	Yes	Yes_NA_0.0434024274223773	UbD
12	11506749	For	CCTTGTGGTTTTCCTGG	NO	NO	Yes	Yes_NA_0.0434024274223773	UbD
12	11506591	Rev	CAAGTCCCG	NO	NO	NO	Yes_rs112485268_0.011360829834527	Ub
12	11506591	For	CGGGACTTG	NO	NO	NO	Yes_rs112485268_0.011360829834527	bD

2	234590527	For	TCTTCCACTTACTATAT	Yes	NO	Yes	Yes_rs7586110_0.297723642172524	UbD
2	233408220	Rev	TGCCCCCCCTCAGGCCT	NO	NO	Yes	NO	UbD
2	233408220	For	AGGCCTGAGGGGGGGCA	NO	NO	Yes	NO	UbD
2	232087474	For	TGAAGTTGATAAAGTTT	Yes	Yes_TP	Yes	NO	UbD
2	231943505	Rev,For	AAAAAAAAG	NO	NO	NO	NO	Ub
2	231943505	Rev	GAAAAAGAT	NO	NO	NO	NO	bD
2	225422600	Rev	TTTTTTTTGTCTCTTCT	NO	NO	NO	NO	UbD
2	211456637	Rev	AGGGAGGGTGTTGTCCA	NO	Yes_TP	Yes	NO	UbD
2	211456637	For	TGGACAACACCCTCCCT	NO	Yes_TP	Yes	NO	UbD
2	211421452	For	ACAAATCATCAAAATGA	NO	Yes_TP	Yes	NO	UbD
2	206630073	For	GTGCAATAACACACACA	NO	NO	Yes	NO	UbD
2	187490389	Rev	ACACACACCCACACACA	NO	NO	Yes	Yes_rs5836987_0.170527156549521	UbD
2	171850207	For	AAGAAAAAG	NO	NO	NO	NO	Ub
2	171850201	For	AAAAAAAAG	NO	NO	NO	NO	Ub
2	171508490	For	CTTTGATCAAAAAAAAT	NO	NO	Yes	NO	UbD
2	170425523	For	AAAAAAAAA	NO	NO	Yes	NO	Ub
2	162876828	For	TTTTTTTTG	NO	Yes	NO	Yes_rs61406317_0.051092245574076	Ub
2	162876828	Rev	GAAAAAAAA	NO	Yes	NO	Yes_rs61406317_0.051092245574076	bD
2	160964059	For	GCTGTTGGAGTCATGGC	NO	NO	NO	Yes_rs13008664_0.265974440894569	UbD
2	160964057	For	AAGCTGTTGGAGTCATG	NO	NO	NO	Yes_rs13009231_0.264976038338658	UbD
2	160086653	For	ATCGCTGCCACTCCTGC	Yes	Yes_TP	Yes	NO	UbD
2	158283710	For	AAGGTAATT	Yes	NO	Yes	NO	bD
2	158152046	For	GAATTCAGAAAAAAAAA	NO	NO	Yes	NO	UbD
2	157425502	For	AAAAAAAAG	NO	NO	NO	Yes_rs200733092_0.258985623003195	Ub
2	157425502	Rev	GAAAAGAAA	NO	NO	NO	Yes_rs200733092_0.258985623003195	bD
2	130899804	Rev	AGGCAAGCATGGGCGTC	NO	Yes_TP	Yes	Yes_rs138331089_0.214447250386341	UbD
2	130899804	For	GACGCCCATGCTTGCCT	NO	Yes_TP	Yes	Yes_rs138331089_0.214447250386341	UbD
2	130737578	For	GCAGGGGCGCAGAGAGC	NO	NO	Yes	NO	UbD
2	128939598	For	GGAGCAGCC	Yes	NO	Yes	Yes_rs13000972_0.254193290734824	bD
2	128872595	For	GCATAATCCGAGGTAAT	Yes	NO	Yes	Yes_rs56397917_0.250399361022364	UbD
2	128394877	Rev	ACAGGGGACGGGGGGGT	NO	Yes_TP	Yes	Yes_rs3217355_0.297087378640777	UbD
2	128394877	For	ACCCCCCCGTCCCCTGT	NO	Yes_TP	Yes	Yes_rs3217355_0.297087378640777	UbD
2	121036385	Rev	AATGCCAAAGGAAGGTG	Yes	Yes_TP	Yes	NO	UbD
2	119748152	Rev	GATCAAAACAAAAAAGG	NO	NO	Yes	Yes_rs199736481_0.172068076328004	UbD
2	119748152	For	CCTTTTTTG	NO	NO	Yes	Yes_rs199736481_0.172068076328004	Ub
2	118753148	Rev	TATAAGACATATCTGAG	Yes	NO	Yes	NO	UbD
2	98205924	For	TCAGCCTGCAAAGGGGT	NO	NO	Yes	NO	UbD
2	98196849	For	TAATATTTGTGACTTGA	NO	NO	Yes	Yes_rs34131105_0.297923322683706	UbD
2	97833583	Rev	TAAAAGAAGTAAGAATC	NO	NO	Yes	NO	UbD
2	97833569	Rev	ATCAGAAGG	NO	NO	Yes	NO	Ub
2	97833567	Rev	CAGAAGGCTATGAACAT	NO	NO	Yes	NO	UbD
2	97833399	Rev	ATTAATATGTCATGTAT	NO	NO	NO	Yes_rs201829271_0.259242384795626	UbD
2	97833394	Rev	TATGTCATGTATATATC	NO	NO	Yes	Yes_rs1636143_0.158119151960204	UbD
2	97833217	For	GTACTTTCCTGTGTTAG	NO	NO	Yes	NO	UbD
2	97833215	For	AAGTACTTTCCTGTGTT	NO	NO	Yes	NO	UbD
2	97833193	For	GTTAAGAGGATCACATT	NO	NO	Yes	NO	UbD
2	97833159	For	GTCCTTTCTTTTGACAT	NO	NO	Yes	NO	UbD
2	97830053	For	CACGGTAAACATATTTT	NO	NO	Yes	Yes_rs112213180_0.241864139020537	UbD
2	97830012	For	TCCTTTTGCTTTGTAGT	NO	NO	Yes	Yes_rs200393754_0.179917229361795	UbD
2	97829998	For	AGTCAATTATGTGTTCC	NO	NO	Yes	Yes_rs201970613_0.151098901098901	UbD
2	97829984	For	TCATATTTACCTGTAGT	NO	NO	Yes	Yes_rs201356360_0.15708604483008	UbD
2	97820478	Rev	TACTCTACCTCAGATTC	NO	NO	NO	Yes_rs79756591_0.261011681491802	UbD
2	97820434	Rev	CTGTGCAAAACGGTCCA	NO	NO	NO	Yes_rs200122064_0.0361216730038023	UbD
2	97820434	For	TGGACCGTTTTGCACAG	NO	NO	NO	Yes_rs200122064_0.0361216730038023	UbD
2	97820431	Rev	TGCAAAACGGTCCAGTA	NO	NO	NO	Yes_97820431_NA_0.0458970792767733	UbD
2	97820431	For	TACTGGACCGTTTTGCA	NO	NO	NO	Yes_97820431_NA_0.0458970792767733	UbD
2	97820430	Rev	GCAAAACGGTCCAGTAG	NO	NO	NO	Yes_NA_0.045689312673818	UbD
2	97820430	For	CTACTGGACCGTTTTGC	NO	NO	NO	Yes_NA_0.045689312673818	UbD
2	97820417	Rev	GTAGGTAGAGACACCTA	NO	Yes_FP	NO	Yes_rs201499984_0.0739617114760958	UbD
2	97820417	For	TAGGTGTCTCTACCTAC	NO	Yes_FP	NO	Yes_rs201499984_0.0739617114760958	UbD
2	97820392	For	ATTTTTGTAATATCTTT	NO	NO	NO	Yes_rs76352034_0.217239829175096	UbD
2	97820349	For	TAGACTGACGGTTTTTG	NO	NO	NO	NO	UbD
2	97820332	For	GAGGGGACCTGCATGTA	NO	NO	NO	NO	UbD
2	97818283	Rev	AAAAATCTCTGAAATCA	NO	NO	Yes	Yes_rs112336445_0.0268336314847943	UbD
2	97818264	Rev	CCTGTCAACAGCCTCTT	NO	Yes	Yes	NO	UbD
2	97818262	Rev	TGTCAACAGCCTCTTCA	NO	NO	NO	NO	UbD
2	97818262	For	TGAAGAGGC	NO	NO	NO	NO	Ub
2	97818261	Rev	GTCAACAGCCTCTTCAA	NO	NO	NO	NO	UbD
2	97818261	For	TTGAAGAGG	NO	NO	NO	NO	Ub
2	97818236	Rev	AGCACTATTAAAGAAAA	NO	NO	Yes	Yes_rs112647905_0.0329607519369999	UbD
2	97818236	For	TTTTCTTTAATAGTGCT	NO	NO	Yes	Yes_rs112647905_0.0329607519369999	UbD
2	97818197	Rev	CAATAGAAC	NO	NO	Yes	Yes_rs200340181_0.000538599640933573	Ub
2	97818197	For	TCTATAGTGTTCTATTG	NO	NO	Yes	Yes_rs200340181_0.000538599640933573	UbD
2	97818192	For	AGTATTCTATAGTGTTC	NO	NO	Yes	Yes_rs202154783_0.00053248136315229	UbD
2	97818188	For	TTTTAGTATTCTATAGT	NO	NO	Yes	NO	UbD
2	97818169	For	CCACTGTGCTCTTTTAT	NO	NO	Yes	NO	UbD
2	97818161	For	GGAACAAGCCACTGTGC	NO	NO	Yes	NO	UbD
2	97818146	For	GTAGCTGGGATTATAGG	NO	NO	Yes	NO	UbD
2	97818114	For	TCTCAAGTGATACTTCT	NO	NO	NO	NO	UbD
2	97817564	For	GTATTATCCTATTTTAA	NO	NO	Yes	NO	UbD
2	97817563	For	GTATTATCCTATTTTAA	NO	NO	Yes	NO	UbD
2	97817545	For	GATCATGTAGTAATGGT	NO	NO	Yes	NO	UbD
2	97817511	For	ATGAGTGATTTCAACGT	NO	NO	Yes	NO	UbD
2	97817469	For	CTGTAGATTAGTAAATG	NO	NO	Yes	NO	UbD
2	97817450	For	ACCATCACCCCATGTGA	NO	NO	Yes	NO	UbD
2	97817449	For	AACCATCACCCCATGTG	NO	NO	Yes	NO	UbD
2	97749417	For	CACACCTGCCACTGGGC	NO	NO	Yes	NO	UbD
2	85571228	Rev	GTGGCAGGC	NO	NO	NO	Yes_rs200882656_0.018703038222803	Ub
2	85571228	For	TCAGCTCCGCCTGCCAC	NO	NO	NO	Yes_rs200882656_0.018703038222803	UbD
2	74649921	For,Rev	TTTTTTTTCAAAAAAAA	NO	NO	NO	NO	UbD
2	74649921	For	CCGTCCCCC	NO	NO	NO	NO	Ub
2	74649916	For	CCGTCCCCC	NO	NO	NO	NO	Ub
2	74649916	For,Rev	TTTTTTTTCAAAAAAAA	NO	NO	NO	NO	UbD
2	71776345	For	CCTCGGGCCACATCTGT	Yes	NO	Yes	NO	UbD
2	62063093	For	AAAGCTGTAAAAAAAAA	NO	NO	Yes	NO	UbD
2	61456624	For	AAAAAAAACAAAAACAC	NO	NO	NO	NO	UbD
12	11461706	Rev	CCCAAGGTCCCCCACCT	NO	Yes_FP	Yes	Yes_rs12308381_0.215901732232475	UbD
12	11461706	For	AGGTGGGGGACCTTGGG	NO	Yes_FP	Yes	Yes_rs12308381_0.215901732232475	UbD
12	11461596	Rev	CAAGGAGGAAACCAGTC	NO	NO	NO	Yes_rs59021567_0.0188633809600257	UbD
12	11461596	For	GACTGGTTTCCTCCTTG	NO	NO	NO	Yes_rs59021567_0.0188633809600257	UbD
12	11461580	Rev	CCCAAGGTACCCCACCT	NO	NO	NO	Yes_rs12303607_0.199891852919971	UbD
12	11461580	For	AGGTGGGGTACCTTGGG	NO	NO	NO	Yes_rs12303607_0.199891852919971	UbD
12	11461570	Rev	CCCACCTCCTCCAGGAA	NO	NO	NO	Yes_rs59189129_0.282818880614941	UbD
12	11461570	For	TTCCTGGAGGAGGTGGG	NO	NO	NO	Yes_rs59189129_0.282818880614941	UbD
12	11461553	Rev	AGCCAGAAAGACCACCC	NO	NO	NO	Yes_rs11054244_0.000131285282919785	UbD
12	11461553	For	GGGTGGTCTTTCTGGCT	NO	NO	NO	Yes_rs11054244_0.000131285282919785	UbD
12	11461549	Rev	AGAAAGACCACCCCCAC	NO	NO	NO	Yes_rs11054243_6.45911380958533e-05	UbD
12	11461549	For	GTGGGGGTGGTCTTTCT	NO	NO	NO	Yes_rs11054243_6.45911380958533e-05	UbD
12	11338613	For	AGGTAAAGCATTTGGTG	NO	Yes_TP	Yes	NO	UbD
12	11244470	Rev	ACTTAAAGA	NO	NO	NO	Yes_rs201460452_0.207429709784036	bD
12	11244190	Rev	CTCCATGGTAAAGGATC	NO	NO	NO	Yes_rs202034865_0.252492389649924	UbD
12	11244190	For	GATCCTTTACCATGGAG	NO	NO	NO	Yes_rs202034865_0.252492389649924	UbD
12	11183052	Rev	CAGTTTTGCGGCAAGTG	NO	NO	Yes	Yes_rs199894662_0.0129403070859676	UbD
12	11183052	For	CACTTGCCGCAAAACTG	NO	NO	Yes	Yes_rs199894662_0.0129403070859676	UbD
12	11183046	Rev	TGCGGCAAGTGAGGTAC	NO	NO	Yes	Yes_rs201730548_0.0128598784997359	UbD
12	11183046	For	GTACCTCACTTGCCGCA	NO	NO	Yes	Yes_rs201730548_0.0128598784997359	UbD
12	11182998	Rev	AGATTCATGAGAGGGGC	NO	NO	Yes	NO	UbD
12	11182998	For	GCCCCTCTCATGAATCT	NO	NO	Yes	NO	UbD
12	11174416	Rev	TTGGAATCTTAGGACAC	NO	Yes_TP	Yes	Yes_rs112900131_0.00762564707424153	UbD
12	11174416	For	GTGTCCTAAGATTCCAA	NO	Yes_TP	Yes	Yes_rs112900131_0.00762564707424153	UbD
12	11174397	Rev	CAGAGCAAACTTGTACT	NO	Yes_FP	Yes	Yes_rs76455106_0.0586554593930132	UbD
12	11174397	For	AGTACAAGTTTGCTCTG	NO	Yes_FP	Yes	Yes_rs76455106_0.0586554593930132	UbD
12	11174390	Rev	AACTTGTACTCCTGCTT	NO	Yes_FP	Yes	Yes_rs74992161_0.0606151288445553	UbD
12	11174390	For	AAGCAGGAGTACAAGTT	NO	Yes_FP	Yes	Yes_rs74992161_0.0606151288445553	UbD
12	11174380	Rev	CCTGCTTTGCCAAACTG	NO	NO	Yes	Yes_rs76970958_0.0838959272170734	UbD
12	11174380	For	CAGTTTGGCAAAGCAGG	NO	NO	Yes	Yes_rs76970958_0.0838959272170734	UbD
12	11174372	Rev	GCCAAACTGTTGCAATC	NO	NO	Yes	Yes_rs74772077_0.0977704459108178	UbD
12	11174372	For	GATTGCAACAGTTTGGC	NO	NO	Yes	Yes_rs74772077_0.0977704459108178	UbD
12	11174327	Rev	TGATTATGGGAAGTAGG	NO	NO	Yes	Yes_rs72475481_0.211818535825545	UbD
12	11174327	For	CCTACTTCCCATAATCA	NO	NO	Yes	Yes_rs72475481_0.211818535825545	UbD
12	11035274	Rev	AGCAGTTCCTAGATGAG	NO	NO	NO	Yes_rs28607516_0.299252741954923	UbD
12	11035274	For	CTCATCTAGGAACTGCT	NO	NO	NO	Yes_rs28607516_0.299252741954923	UbD
12	10588647	Rev	ATACACCATTTATAGCT	NO	NO	Yes	NO	UbD
12	10586961	For	TAAATATTT	NO	NO	Yes	NO	bD
12	10586956	For	CTTTATAAA	NO	NO	Yes	NO	bD
12	10573211	Rev	ATACACCATTTATAGCT	NO	NO	Yes	NO	UbD
12	10573094	Rev	CCCAAAGTGGCAGCAAA	NO	Yes_TP	Yes	NO	UbD
12	10573094	For	TTTGCTGCC	NO	Yes_TP	Yes	NO	Ub
12	9590505	For	ACAGCAACGGCAGGGGA	NO	NO	Yes	NO	UbD
12	9590464	For	AGCACCATCGACCCAAG	NO	NO	Yes	NO	UbD
12	9583324	Rev	TGGTGCTGCCCTATCAG	NO	NO	NO	NO	UbD
12	9583221	Rev	CGACACCAC	NO	NO	Yes	NO	Ub
12	9582244	Rev	CCTTAGCCCTCGGCTGC	NO	NO	Yes	NO	UbD
12	9581948	For	GCTGGCCACGGCCCCAG	NO	NO	NO	NO	UbD
12	9580395	Rev	CCTCAGAGGGCCCAGAG	NO	NO	Yes	NO	UbD
12	9580203	Rev	TCCACATTCCACATCCA	NO	NO	Yes	NO	UbD
12	9580203	For	TGGATGTGGAATGTGGA	NO	NO	Yes	NO	UbD
12	9578197	Rev	TGTGCAGCTCTTGCAGC	NO	NO	Yes	NO	UbD
12	9578197	For	GCTGCAAGAGCTGCACA	NO	NO	Yes	NO	UbD
12	9577992	For	AAAGAATCCTTTCATGC	NO	NO	NO	NO	UbD
12	9574480	Rev	AGGTGTCTAACTTCCGG	NO	NO	Yes	NO	UbD
12	9574480	For	CCGGAAGTTAGACACCT	NO	NO	Yes	NO	UbD
12	9573309	Rev	CATGGGGGCTCCCTCCC	NO	NO	NO	NO	UbD
12	9573307	Rev	ATGGGGGCCCCTCCCTC	NO	NO	NO	NO	UbD
12	9573304	Rev	GGGGCTCCCCCTCCCTC	NO	NO	NO	NO	UbD
12	9573287	Rev	ATGGGGGCTCCCTCCCT	NO	NO	Yes	NO	UbD
12	9573224	Rev	ACCTGTGCGGTGTGGTT	NO	NO	Yes	NO	UbD
12	9573224	For	AACCACACCGCACAGGT	NO	NO	Yes	NO	UbD
12	9573130	For	CAGCCAGACGGCCCAGC	NO	NO	NO	NO	UbD
12	9572871	Rev	TCTGGGAAATGTCCTCT	NO	NO	Yes	NO	UbD
12	9572819	Rev	CACCAGGTGGAGCAGGT	NO	NO	NO	NO	UbD
12	8693245	For	TCCCCCCACTTTTTTTT	NO	NO	Yes	NO	UbD
12	8203273	Rev	GTCAAAGGAAAGCAAAG	Yes	NO	Yes	Yes_rs17801353_0.218250798722045	UbD
12	7362204	For	GAGTTTGAGCTGAGTCG	NO	NO	Yes	NO	UbD
12	7290391	For	TGTGTGTGTGAGAGAGA	NO	NO	NO	NO	UbD
12	7290389	For	TGTGTGTGTGAGAGAGA	NO	NO	NO	NO	UbD
12	6928586	Rev	CAGAGGCTGGGGATCTG	Yes	Yes_TP	Yes	NO	UbD
12	6161963	Rev	GCCTTACTTCTCCTCTC	Yes	Yes_TP	Yes	Yes_NA_2.83527076835838e-05	UbD
12	2791130	For	GCACGTTCCGATGTGTG	NO	Yes_TP	Yes	NO	UbD
12	2016550	For	TGCCCCTATTATTTCCA	Yes	NO	Yes	NO	UbD
12	1041996	For	TACTAGGTAAACGTACT	NO	NO	NO	Yes_rs113084521_0.00139776357827476	UbD
13	113333878	Rev	GTGGCTGGTCCCATGAC	NO	NO	NO	Yes_rs200954527_0.0398414802696528	UbD
13	113333878	For	GTCATGGGACCAGCCAC	NO	NO	NO	Yes_rs200954527_0.0398414802696528	UbD
13	111109618	For	GTGGGGCTGATGCCCTG	NO	NO	NO	Yes_rs9515216_0.00710357470210816	UbD
13	111109576	For	GTGGGGCTGATGCCCTG	NO	NO	NO	Yes_rs9521780_0.00207373271889401	UbD
13	99112754	Rev	GTTTATTAAATTATTGC	Yes	NO	Yes	NO	UbD
13	95858704	For	ATTACACTGGAACATAG	Yes	NO	Yes	NO	UbD
13	95715148	Rev	ATTTTTTTC	NO	NO	Yes	NO	Ub
13	77644862	Rev	TTTTTTTTCTTTTTCTA	NO	NO	NO	Yes_rs371232979_0.00483745475457528	UbD
13	75915215	For	AACAAAATTATTAAACA	Yes	Yes_TP	Yes	NO	UbD
13	52707751	For	CACACACAC	NO	NO	Yes	NO	bD
13	52585591	Rev	CCGAGCCGCGCCGATGC	Yes	Yes_TP	Yes	NO	UbD
13	50100637	Rev	TGATAAATATTTCTCAA	NO	NO	NO	Yes_rs371713260_0.25227716330514	UbD
13	46108853	Rev	TAGAAACCAACAGCAGA	Yes	Yes_TP	Yes	NO	UbD
13	46108853	For	TCTGCTGTTGGTTTCTA	Yes	Yes_TP	Yes	NO	UbD
13	43491778	Rev	TTTTTTTTCTCTTTGTA	NO	Yes	NO	NO	UbD
13	26153950	For	GCAGGCTTG	NO	NO	NO	Yes_rs201822155_0.0423402879411865	Ub
13	26128128	Rev	TTTTGAGTGCTCAATTA	Yes	NO	Yes	Yes_rs7331675_0.0818690095846645	UbD
13	25671080	Rev	ATCTCATCTACAGCTTT	NO	NO	NO	Yes_rs76241879_0.111983284391236	UbD
13	25671080	For	AAAGCTGTAGATGAGAT	NO	NO	NO	Yes_rs76241879_0.111983284391236	UbD
13	25671062	Rev	TGTGCATCTTCATGCCT	NO	NO	NO	Yes_rs78534836_0.0663115968877362	UbD
13	25671062	For	AGGCATGAAGATGCACA	NO	NO	NO	Yes_rs78534836_0.0663115968877362	UbD

2	61456611	Rev	CAAAAAAAA	NO	NO	NO	NO	bD
2	61456611	For	AAAAAAAAC	NO	NO	NO	NO	Ub
2	58362161	Rev	AAAAAAAAG	NO	NO	Yes	Yes_rs11678320_0.294329073482428	Ub
2	58362161	For	GCATTGAAA	NO	NO	Yes	Yes_rs11678320_0.294329073482428	bD
2	39536669	For	GAAAAAAAA	NO	NO	NO	Yes_rs369443408_0.0814145974416855	bD
2	39536669	Rev	TTTTTTTTCTTTTTTAG	NO	NO	NO	Yes_rs369443408_0.0814145974416855	UbD
2	38963046	For	AAGAAAAAG	NO	NO	NO	NO	Ub
2	33623387	For	CTCACAGAATAATATCC	Yes	Yes_TP	Yes	NO	UbD
2	33586479	For,Rev	TTTTTTTTCAAAAAAAA	NO	NO	NO	Yes_rs77418198_0.172530795898798	UbD
2	32961691	For	ATTAGGTACACCTGCAA	NO	NO	NO	Yes_rs17497801_0.291134185303514	UbD
2	31572482	For	TCCTATATGGGGTCAGC	Yes	NO	Yes	NO	UbD
2	27324340	Rev	GATGCCCCCGGGCCCAG	NO	NO	NO	Yes_rs11893427_8.36446774103608e-05	UbD
2	15491994	For	TCTGCATGCACACACAC	NO	NO	Yes	NO	UbD
2	15378822	Rev	CTCACTATTCTTTTTTT	NO	Yes_TP	Yes	NO	UbD
2	8822912	For	AAAAAAAAAAAAAACCC	NO	NO	Yes	NO	UbD
2	1642537	For	GAATAAAATGCTATACC	Yes	NO	Yes	NO	UbD
2	1133188	For	CACACACAC	NO	NO	NO	NO	bD
2	1079130	For	AGCAGGAGGCGTGCGTC	Yes	NO	Yes	NO	UbD
2	905736	Rev	CGGGAGGATCTGAAACA	NO	NO	NO	Yes_rs11891241_0.101102941176471	UbD
2	905442	Rev	CGGGAGGATCTGAAACA	NO	NO	NO	Yes_rs62103942_0.0446957640605424	UbD
2	242732	For	ACAACAATAACAACAAC	Yes	NO	Yes	NO	UbD
3	197544211	Rev	ACACACACA	NO	NO	NO	Yes_rs367618377_0.220143240823635	bD
3	196674972	Rev	CTGGGGCAGCCCTCACA	Yes	Yes_TP	Yes	NO	UbD
3	196674972	For	TGTGAGGGCTGCCCCAG	Yes	Yes_TP	Yes	NO	UbD
3	195515387	For	AGGGGTGGTGTGACCTG	NO	NO	NO	Yes_rs13065584_0.00913176515586154	UbD
3	195511959	For	GAAGTGTCGGTGACAGG	NO	NO	NO	Yes_rs369770584_0.280307510348906	UbD
3	195510614	For	TGGTGACAG	NO	NO	NO	NO	bD
3	195510613	For	CTGGTGACA	NO	NO	NO	NO	bD
3	195510611	For	GGCTGGTGA	NO	NO	NO	Yes_NA_0.00252525252525253	bD
3	195507827	For	GGCTGGTGA	NO	NO	Yes	Yes_rs373469782_0.177501055297594	bD
3	195506775	For	GAAGTGTCGGTGACAGG	NO	NO	NO	Yes_rs79609066_0.00314465408805031	UbD
3	195506245	For	AGGAAGGGCTGGTGACA	NO	NO	NO	Yes_rs150394536_0.000304460344040189	UbD
3	195456708	Rev	CTTCCAAGCGCGGTAGC	NO	NO	Yes	NO	UbD
3	195451880	Rev	GGCTTCCCTGGGATCAC	NO	NO	Yes	NO	UbD
3	195451880	For	GTGATCCCAGGGAAGCC	NO	NO	Yes	NO	UbD
3	195426490	Rev	GGCTGACTTATTGAAAA	NO	NO	Yes	NO	UbD
3	195426207	For	TGTCCTGGAACCCTGGG	NO	NO	NO	NO	UbD
3	194373832	Rev	ACACAACCAAGTTTGGA	Yes	Yes_TP	Yes	NO	UbD
3	189712089	Rev	TTTTTTCCCGTCTGTTT	Yes	Yes_TP	Yes	NO	UbD
3	186953808	Rev	GATGTCCTGCAGTATGT	NO	NO	NO	Yes_rs850312_0.2909912109375	UbD
3	186953808	For	ACATACTGCAGGACATC	NO	NO	NO	Yes_rs850312_0.2909912109375	UbD
3	186386664	For	AAAAAAAAA	NO	NO	Yes	NO	bD
3	185906246	Rev	CACACACAC	NO	NO	NO	NO	Ub
3	183756837	Rev	CAAGGCTGACTGAGAGC	NO	NO	Yes	NO	UbD
3	183756836	Rev	AAGGCTGACTGAGAGCC	NO	Yes_TP	Yes	NO	UbD
3	182810144	For	GAAAATACTGACACAGT	Yes	NO	Yes	NO	UbD
3	179481971	Rev	ACACACACAGTCTCTCA	NO	NO	Yes	NO	UbD
3	179469995	For	CCTCAGACAAAGTTGAT	Yes	NO	Yes	Yes_rs2239639_0.262579872204473	UbD
3	154018690	For	TTAAGTACGTACAGATT	Yes	NO	Yes	Yes_rs3732652_0.243809904153355	UbD
3	153839959	Rev	CGCTGCGAGGAGCGTCC	Yes	Yes_TP	Yes	NO	UbD
3	142215178	For	ATGATGACA	NO	Yes_TP	NO	NO	bD
3	136287541	For	GTGGCAGGT	NO	NO	NO	Yes_rs549823681_0.194688498402556	Ub
3	133901733	For	AAAAATTAAGCTATTCC	NO	NO	NO	Yes_rs9829505_0.235423322683706	UbD
3	129147418	Rev	CGGTTTATCTGATACCC	NO	NO	NO	Yes_rs3796391_0.178314696485623	UbD
3	129147418	For	GGGTATCAGATAAACCG	NO	NO	NO	Yes_rs3796391_0.178314696485623	UbD
3	128859202	Rev	CGTAAGTAAGTTTGTGA	NO	Yes_TP	Yes	NO	UbD
3	128859202	For	TCACAAACTTACTTACG	NO	Yes_TP	Yes	NO	UbD
3	123066555	For	CCACTTGAAAGGAAGTG	Yes	NO	Yes	NO	UbD
3	122821671	Rev	CTCCACCATTGCTGAGA	Yes	Yes_TP	Yes	NO	UbD
3	122821671	For	TCTCAGCAATGGTGGAG	Yes	Yes_TP	Yes	NO	UbD
3	121195302	Rev	ACACACACA	NO	NO	Yes	NO	bD
3	121195302	For	AGGTAAATA	NO	NO	Yes	NO	Ub
3	116163634	Rev	ACACACACA	NO	NO	Yes	NO	bD
3	116163634	For	GAGATCAGA	NO	NO	Yes	NO	Ub
3	113557787	For	AGGTAAGCCGCGGGCCT	Yes	Yes_TP	Yes	NO	UbD
3	113524266	Rev	CAGCTCTACGAGCCATA	NO	NO	Yes	Yes_NA_0.184776047325848	UbD
3	112301458	For	GGTTTTCCCTTTAAAAA	NO	NO	Yes	NO	UbD
3	108355575	Rev	AAAAAAAAAGGACAAAA	NO	NO	Yes	Yes_NA_0.205806156659555	UbD
3	108355575	For	TTTTGTCCTTTTTTTTT	NO	NO	Yes	Yes_NA_0.205806156659555	UbD
3	108211869	For	AAGAAGAAAAGGAAGGA	Yes	NO	Yes	NO	UbD
3	108135606	Rev	TTTTTTTTT	NO	NO	Yes	Yes_rs529118404_0.00479233226837061	Ub
3	108135606	For	TATTTTTTAAAAAAAAA	NO	NO	Yes	Yes_rs529118404_0.00479233226837061	UbD
3	108102586	Rev	ATTTTTATA	NO	NO	NO	Yes_rs7426463_0.211550950525446	bD
3	101399910	For	AAAAAAAAA	NO	NO	Yes	NO	bD
3	101399910	Rev	TTTTTTTTT	NO	NO	Yes	NO	Ub
3	100064902	For	TCTGTCCTCAGGTCCTG	Yes	NO	Yes	Yes_rs2289506_0.229432907348243	UbD
3	100064829	For	CTCCGCTGA	NO	NO	NO	Yes_rs2289505_0.230031948881789	bD
3	98518160	For	AAAAAAAAA	NO	NO	Yes	NO	bD
3	75790860	Rev	AGGTAGCTGTGCACTTC	NO	NO	Yes	Yes_rs73117241_0.0148767356191556	UbD
3	75790860	For	GAAGTGCACAGCTACCT	NO	NO	Yes	Yes_rs73117241_0.0148767356191556	UbD
3	75790838	For	GGTCCTGCCACTCCTCC	NO	NO	Yes	Yes_rs113991634_0.0295826326249809	UbD
3	75790814	For	ACAGGGTCCTCTGAGCA	NO	NO	Yes	Yes_rs200578600_0.0720533111203665	UbD
3	75790811	For	TGTACAGGGTCCTCTGA	NO	NO	Yes	Yes_NA_0.0729240345169048	UbD
3	75790810	For	TGTACAGGGTCCTCTGA	NO	NO	Yes	NO	UbD
3	75790797	For	CAGCATCACGTCCCTGT	NO	NO	Yes	Yes_NA_0.000674114021571649	UbD
3	75680133	Rev	AGCCCTGGCTCTGCAAA	NO	NO	Yes	NO	UbD
3	75679820	For	CATTGCAGGTGTGCAGC	NO	NO	Yes	NO	UbD
3	73440108	Rev	TTTTTTTTT	NO	NO	Yes	NO	Ub
3	73440108	For	TGAACATGAAAAAAAAA	NO	NO	Yes	NO	UbD
3	62518392	For	TGTGTGTGT	NO	NO	Yes	NO	bD
3	57582718	For	GGCGGAGGAAAAAAACA	NO	NO	Yes	NO	UbD
3	57488279	Rev	AAAAAAAAAAAGATCTA	NO	NO	Yes	NO	UbD
3	57438710	Rev	TGTGGCTTG	NO	NO	NO	Yes_rs201687411_0.2325	Ub
3	56591278	Rev	CTTACCCCTGCTTACCC	NO	NO	Yes	NO	UbD
3	56591278	For	GGGTAAGCAGGGGTAAG	NO	NO	Yes	NO	UbD
3	53220473	Rev	GAAAGCATG	NO	NO	NO	NO	bD

13	25671027	Rev	AAATCCTTTGGATTTTC	NO	Yes_FP	NO	Yes_rs78826513_0.00198957528322673	UbD
13	25671027	For	GAAAATCCAAAGGATTT	NO	Yes_FP	NO	Yes_rs78826513_0.00198957528322673	UbD
13	25670907	Rev	AACATTGGGGAACTCTT	NO	NO	NO	Yes_rs76264750_0.000392208131781932	UbD
13	25670907	For	AAGAGTTCCCCAATGTT	NO	NO	NO	Yes_rs76264750_0.000392208131781932	UbD
13	25670877	Rev	AAGTTCAGCTTCTCGTT	NO	Yes_FP	NO	Yes_rs112107735_0.00130524873146139	UbD
13	25670877	For	AACGAGAAGCTGAACTT	NO	Yes_FP	NO	Yes_rs112107735_0.00130524873146139	UbD
13	21442945	Rev	CCCCCCAAA	Yes	NO	Yes	Yes_rs9509410_0.262979233226837	bD
13	20279731	For	TATATTAAGTGTGAAGT	NO	NO	NO	NO	UbD
14	107178748	For	TCTGCAGAAGTGGGGAG	NO	NO	Yes	NO	UbD
14	107131394	Rev	AGTGAACACGAGTGAGA	NO	NO	Yes	NO	UbD
14	107131394	For	TCTCACTCGTGTTCACT	NO	NO	Yes	NO	UbD
14	107130977	For	TTCCCCCAACGTTCCTG	NO	NO	Yes	NO	UbD
14	106361561	Rev	AGGCCTGGCGGTAGGTT	NO	NO	Yes	NO	UbD
14	106361561	For	AACCTACCGCCAGGCCT	NO	NO	Yes	NO	UbD
14	106359328	For	CAGGTGAGCTGGAGGCC	NO	NO	NO	Yes_rs2983757_0.231829073482428	UbD
14	106359327	For	GCAGGTGAGCTGGAGGC	NO	NO	NO	Yes_rs3021114_0.232228434504792	UbD
14	106330320	For	CCCCCAGGATGCTCCGG	NO	NO	Yes	NO	UbD
14	106330319	For	CCCCCAGGATGCTCCGG	NO	NO	Yes	NO	UbD
14	106329452	Rev	CTACTACTACATGGACG	NO	NO	NO	Yes_rs2338627_0.0194026040336993	UbD
14	106329452	For	CGTCCATGTAGTAGTAG	NO	NO	NO	Yes_rs2338627_0.0194026040336993	UbD
14	106329451	Rev	TACTACTACATGGACGT	NO	NO	NO	Yes_rs2338628_0.0380165289256198	UbD
14	106329451	For	ACGTCCATGTAGTAGTA	NO	NO	NO	Yes_rs2338628_0.0380165289256198	UbD
14	105418166	For	GTCAGTGGTCTTGAGGT	NO	NO	Yes	Yes_rs78116894_0.129344251724805	UbD
14	105412541	Rev	AGATAGATGTCAAGGGC	NO	NO	Yes	NO	UbD
14	105412541	For	GCCCTTGACATCTATCT	NO	NO	Yes	NO	UbD
14	105258892	For	ACTACCCCCATCTCTCC	Yes	Yes_TP	Yes	NO	UbD
14	104167564	Rev	CAGTCCCATCCTGGAGC	Yes	Yes_TP	Yes	Yes_rs861536_0.298923679060665	UbD
14	104167564	For	GCTCCAGGA	Yes	Yes_TP	Yes	Yes_rs861536_0.298923679060665	Ub
14	103440282	For	CCATCATCGGCATGAAC	Yes	NO	Yes	NO	UbD
14	102551795	Rev	TGAATGTATTAAGTCAA	NO	Yes_TP	Yes	NO	UbD
14	95566105	For	CACACACACAAACTTAC	NO	NO	Yes	Yes_rs368080354_0.112291968162084	UbD
14	95566068	For	CACACACACAAACTTAC	NO	NO	NO	NO	UbD
14	94582130	For	GCAGTTGTGGCTGTGCC	NO	Yes_TP	Yes	NO	UbD
14	77941865	For	AGGAAAGTT	Yes	Yes_TP	Yes	NO	bD
14	75016482	For	CTCTGACATCCTGGAGG	NO	NO	Yes	NO	UbD
14	70924507	Rev	TCCAGGAGTGCACGGTC	NO	NO	Yes	Yes_rs3751524_0.29379264192211	UbD
14	70924507	For	GACCGTGCACTCCTGGA	NO	NO	Yes	Yes_rs3751524_0.29379264192211	UbD
14	66082573	For	TTGGTACTACTTTTGTG	NO	NO	NO	Yes_rs8012278_0.254992012779553	UbD
14	52509483	Rev	CTGGGGCCAGGAACTCC	Yes	Yes_TP	Yes	NO	UbD
14	52509483	For	GGAGTTCCTGGCCCCAG	Yes	Yes_TP	Yes	NO	UbD
14	51492135	Rev	GTCTCGCTGGCCTTGTT	Yes	NO	Yes	NO	UbD
14	51208236	For	ACCAAGTTGTCCTAAGA	Yes	NO	Yes	NO	UbD
14	50298153	Rev	AAAAAAAAA	Yes	NO	Yes	NO	Ub
14	24910973	Rev	AAGAGGTAATCGGCAGC	NO	Yes_TP	Yes	NO	UbD
14	24910973	For	GCTGCCGAT	NO	Yes_TP	Yes	NO	Ub
14	24795231	Rev	TCATATGGTGGAGAAGG	Yes	NO	Yes	NO	UbD
14	24795231	For	CCTTCTCCACCATATGA	Yes	NO	Yes	NO	UbD
14	22999202	For	ATGAGGGAGATAGCTGC	Yes	NO	Yes	NO	UbD
14	22957488	For	GCTGAATCGTACCTGAG	Yes	NO	Yes	Yes_rs1483971_0.270367412140575	UbD
15	93616283	Rev	TTTTCTTCTTTTTTTCT	Yes	Yes_TP	Yes	NO	UbD
15	91544558	For	AAAAAAAAGAAAAGTAC	NO	NO	NO	NO	UbD
15	90328774	Rev	TCAAGGCTGTTAGGGAC	Yes	Yes_TP	Yes	NO	UbD
15	90195765	For	AAGAGGACAAGGCAGAA	Yes	NO	Yes	Yes_rs11630003_0.0115814696485623	UbD
15	84257351	For	GGCAGAGAGGATGTGTG	Yes	NO	Yes	Yes_rs7182482_0.192292332268371	UbD
15	75664570	Rev	TTTTTTTTT	Yes	Yes_TP	Yes	NO	bD
15	75651022	For	CACCATCCCCACATGCT	Yes	Yes_TP	Yes	NO	UbD
15	65114505	Rev	GGGGTGGCAGCCTGCCA	NO	NO	NO	Yes_NA_0.0203073332036569	UbD
15	65114504	Rev	GGGTGGCAGCCTGCCAC	NO	NO	NO	Yes_NA_0.0321244477172312	UbD
15	45454720	Rev	CACCACCCA	NO	NO	Yes	Yes_rs1706811_0.244808306709265	bD
15	45445499	For	CTGGCCAGGGTCCTCGA	Yes	NO	Yes	Yes_rs1648304_0.24241214057508	UbD
15	45250755	Rev	GATTAGCTGGATTTGGC	NO	NO	Yes	NO	UbD
15	43170690	For	ATATGTATACACACACA	NO	NO	Yes	NO	UbD
15	41810439	Rev	CCACACACCTTATGTAT	Yes	NO	Yes	Yes_rs932959_0.299321086261981	UbD
15	41224280	For	CCAGGAGAGCTAGGGGA	Yes	NO	Yes	Yes_rs3212279_0.157547923322684	UbD
15	40282701	Rev	AGGTTTGTATAATCTCC	NO	NO	NO	Yes_rs2307104_0.268570287539936	UbD
15	35230934	Rev	TTTCTGGTATCTTCTTT	NO	Yes_TP	Yes	NO	UbD
15	35230934	For	AAAGAAGATACCAGAAA	NO	Yes_TP	Yes	NO	UbD
15	33200798	Rev	TTGGTTTTGTTTTCTAT	NO	NO	NO	Yes_NA_0.0508373205741627	UbD
15	33200793	Rev	TTTGTTTTCTATTTTGA	NO	NO	NO	Yes_NA_0.00916104146576664	UbD
15	33200791	Rev	TGTTTTCTATTTTGATA	NO	NO	NO	Yes_NA_0.00686116236162362	UbD
15	20588458	For	CTGGCCATGACTGCAGT	NO	NO	NO	NO	UbD
16	89973777	Rev	AGCTGGGAACTGGGAGG	NO	NO	Yes	NO	UbD
16	89753031	For	TAAGGAGAGGAAGCCCG	Yes	Yes_TP	Yes	Yes_rs397891_0.195287539936102	UbD
16	89703519	For	TCCAGCCACGAAGGATG	NO	NO	NO	NO	UbD
16	89703424	Rev	GTCCCACGTGAGGAGAT	NO	NO	NO	Yes_rs445537_0.258386581469649	UbD
16	89178474	For	CATCTTCCTACCGAGTG	NO	Yes_TP	NO	Yes_rs9921293_0.0540639573617588	UbD
16	88779952	Rev	ACCATCCGAGGCCGCAG	Yes	NO	Yes	NO	UbD
16	88776499	Rev	AGGCCAGGCTGCCTGTG	NO	NO	NO	Yes_rs61744010_0.184105431309904	UbD
16	88776499	For	CACAGGCAGCCTGGCCT	NO	NO	NO	Yes_rs61744010_0.184105431309904	UbD
16	87436764	Rev	TGGTAACATCCCTTTTT	NO	NO	Yes	NO	UbD
16	85814899	Rev	TGGTAGCCTATTTCTTT	Yes	Yes_TP	Yes	NO	UbD
16	85814899	For	AAAGAAATAGGCTACCA	Yes	Yes_TP	Yes	NO	UbD
16	84808969	Rev	TCACGTGACTCCCCGAC	Yes	NO	Yes	Yes_rs2303232_0.189496805111821	UbD
16	84132628	Rev	CAAATTTCTCTCTGCTG	Yes	Yes_TP	Yes	NO	UbD
16	84132628	For	CAGCAGAGAGAAATTTG	Yes	Yes_TP	Yes	NO	UbD
16	81892043	Rev	TTTTTTTTTCCCCCCTG	Yes	NO	Yes	Yes_rs559329422_0.00119808306709265	UbD
16	81892041	Rev	TTTTTTTTTCCCCCCTG	NO	NO	NO	Yes_rs559329422_0.00119808306709265	UbD
16	81201774	Rev	TGTCCCTTGAATGACAC	NO	NO	NO	Yes_rs34172014_0.280950479233227	UbD
16	81201440	For	GGGGGGGGA	NO	NO	Yes	NO	bD
16	77401684	Rev	GTTGTTGAAAACCAACC	Yes	NO	Yes	Yes_rs2434895_0.284544728434505	UbD
16	74443733	Rev	AGCTGTTAACCCAGGAA	NO	NO	Yes	Yes_rs2549216_0.0283448865254962	UbD
16	74443401	For	AGCAGCACTCCCAGGGT	NO	NO	Yes	Yes_rs62055231_0.0641193927633832	UbD
16	71418447	Rev	TGTGTGTGT	NO	NO	NO	NO	Ub
16	71418423	Rev	TGTGTGTGTTGTGTGTG	NO	NO	NO	NO	UbD
16	71007657	For	TGATGAGCGCCAAGACC	NO	NO	Yes	NO	UbD
16	70732483	For	TCACACAGGGTGTGCCC	Yes	NO	Yes	Yes_rs7201952_0.273761980830671	UbD

3	53220473	For	CATGCTTTCCCCCCTCA	NO	NO	NO	NO	UbD	16	70187550	Rev	CACATCAGTTCTTCTTC	NO	NO	Yes	NO	UbD
3	52878746	For	GTTAAAACAAGTTTGGT	Yes	NO	Yes	NO	UbD	16	70177648	Rev	ATGGTTCCCGGAGAAGC	NO	NO	Yes	NO	UbD
3	52877649	For	GGGCTACAGGTGAGCTT	Yes	NO	Yes	NO	UbD	16	70164248	For	GTTTCTTCTGGGTGGAG	NO	NO	NO	NO	UbD
3	52877647	For	AGGGCTACAGGTGAGCT	Yes	NO	Yes	Yes_rs55829187_0.00559105431309904	UbD	16	69951847	Rev	GAACACGTGGCTGCACA	Yes	NO	Yes	NO	UbD
3	52584431	For	GAAGGAATCTTACCTGC	NO	NO	NO	NO	UbD	16	67970188	Rev	CCGATACGCGAGCCACT	NO	UNKNOW	NO	Yes_NA_0.00123780551767544	UbD
3	51471302	For	AAAAAAAAGAAAGAAAG	NO	NO	Yes	NO	UbD	16	67970188	For	AGTGGCTCG	NO	UNKNOW	NO	Yes_NA_0.00123780551767544	Ub
3	50230913	Rev	GTGGCTGGT	NO	NO	NO	Yes_rs201243059_0.202618345760361	Ub	16	57071113	For	AGCCTCAGTAACAACGG	Yes	Yes_TP	Yes	NO	UbD
3	49725948	For	TGGCTCCCCGACTTTTT	NO	NO	Yes	NO	UbD	16	56601720	Rev	TCTTCCAATATGTTTTA	Yes	Yes_TP	Yes	NO	UbD
3	48602523	For	CCCCGCTGGCAGCCCCC	Yes	Yes_TP	Yes	NO	UbD	16	56601720	For	TAAAACATATTGGAAGA	Yes	Yes_TP	Yes	NO	UbD
3	46700271	Rev	TTTTTTTTT	NO	NO	Yes	NO	Ub	16	55857396	For	GGCATGACAAGAGCTGG	NO	NO	Yes	NO	UbD
3	46700271	For	AAAAAAAAA	NO	NO	Yes	NO	bD	16	33965627	Rev	ATGCACCCC	NO	NO	NO	Yes_NA_0.0278174687623507	bD
3	46501284	Rev	CCGTAGGAGGAGTGTTC	Yes	Yes_TP	Yes	NO	UbD	16	33965582	Rev	ACTGATCGGCAAGCGAC	NO	NO	Yes	Yes_rs199951204_0.23672396479414	UbD
3	45677637	Rev	GCTCCTTGCAGGACAAG	NO	NO	NO	NO	UbD	16	33965560	Rev	GACAGGCAAAGCCCTGG	NO	NO	Yes	Yes_rs75603223_0.222564631144955	UbD
3	45677637	For	CTTGTCCTGCAAGGAGC	NO	NO	NO	NO	UbD	16	33965538	Rev	ACCCGGGCTGCAATTGC	NO	NO	Yes	Yes_rs146300192_0.110999522711002	UbD
3	37512646	Rev	TTTCCACCACAGCAGTA	NO	NO	NO	NO	UbD	16	33965533	Rev	GGCTGCAATTGCGTTCG	NO	NO	Yes	Yes_rs78817399_0.118027210884354	UbD
3	31663515	For	GTTTTATAGTTTTAAGT	Yes	NO	Yes	NO	UbD	16	33965533	For	CGAACGCAA	NO	NO	Yes	Yes_rs78817399_0.118027210884354	Ub
3	30842346	For	CACACAGAAAGAGAGAG	NO	NO	Yes	Yes_rs111777308_0.00623791754709347	UbD	16	33965507	For	GCAGGACAGATTGATCA	NO	NO	Yes	Yes_rs74614199_0.0208877519770856	UbD
3	30842342	For	CACACAGACAGAAAGAG	NO	NO	NO	NO	UbD	16	33965505	For	TTGCAGGACAGATTGAT	NO	NO	Yes	Yes_rs375920374_0.0954734004931853	UbD
3	30842342	For,Rev	CACACACAC	NO	NO	NO	NO	Ub	16	33965456	For	GGCTTCTGTGTCGATGA	NO	NO	Yes	NO	UbD
4	190878485	For	TTAATTCAGTCTAAACA	NO	NO	Yes	NO	UbD	16	33965449	For	ATCACTGGGCTTCTGTG	NO	NO	Yes	NO	UbD
4	190878463	For	TTTTTAAATCAGGGAGG	NO	NO	Yes	NO	UbD	16	31388640	Rev	TTTTTTTTCTTTTTGAG	NO	NO	Yes	Yes_rs11574643_0.140375399361022	UbD
4	186444704	Rev	ATAAGTTAGTGAGGTTT	NO	NO	Yes	NO	UbD	16	31151521	For	AAAAAAAAA	Yes	NO	Yes	NO	bD
4	183650026	For	AAAAAAAAA	NO	NO	Yes	NO	bD	16	24357998	For	CCCAGGCTGGCCCCCAG	Yes	NO	Yes	NO	UbD
4	183601595	Rev	CCCCCCCCC	NO	NO	NO	NO	Ub	16	22128052	For	CCTGGGATTTCCACTTC	NO	Yes_TP	Yes	NO	UbD
4	183601594	Rev	CCCCCCCCC	NO	NO	NO	NO	Ub	16	19455318	For	ATGATTGAATGGATGGA	NO	NO	Yes	NO	UbD
4	183601593	Rev	CCCCCCCCC	NO	NO	NO	NO	Ub	16	16253283	For	ACCCATTGCCCCCCCCC	NO	NO	Yes	NO	UbD
4	183601590	Rev	CCCCCCCCC	NO	NO	Yes	[Yes_rs34738241_0.00159744408945687	Ub	16	15120411	For	TAGAGCATTTCTAAGCG	NO	NO	Yes	NO	UbD
4	151161470	For	TTCCTAACAAATAAAGG	NO	NO	Yes	Yes_rs11931842_0.167132587859425	UbD	16	8992868	For	AGAAATGGACTGTCAGG	Yes	NO	Yes	NO	UbD
4	145040962	Rev	ATTCTTGTGCCCTTTCT	NO	NO	Yes	Yes_NA_8.21611673458657e-06	UbD	16	3078616	For	GGGATCATTATAACCCG	Yes	NO	Yes	NO	UbD
4	142640462	For	AATTCCCTCCCTTTCTT	NO	NO	Yes	NO	UbD	16	1907701	For	CCCCTGCCAGATTGCAC	NO	Yes_TP	Yes	NO	UbD
4	140811111	Rev	CAGCAGCAGCAGCAGCA	NO	NO	NO	Yes_rs62344937_0.00261629990681672	UbD	16	1306971	Rev	TGTGGATGTGGCTGGAG	NO	Yes_TP	Yes	NO	UbD
4	140811111	For	TGCTGCTGCTGCTGCTG	NO	NO	NO	Yes_rs62344937_0.00261629990681672	UbD	16	1306971	For	CTCCAGCCACATCCACA	NO	Yes_TP	Yes	NO	UbD
4	140811096	For	TGCTGCTGCTGCTGCTG	NO	NO	NO	Yes_rs79090010_0.251597444089457	UbD	16	1279732	Rev	AGCCCCAGGCCAGGCCC	NO	Yes_TP	Yes	Yes_rs201836020_0.280053137134682	UbD
4	140811096	Rev	CAGCAGCAGCAGCAGCA	NO	NO	NO	Yes_rs79090010_0.251597444089457	UbD	16	539093	Rev	AGCAAGTTCTCTCACCA	NO	NO	Yes	Yes_rs74390230_0.159944089456869	UbD
4	129869520	For	GAAAAGAAA	NO	NO	NO	NO	bD	16	456291	For	GGCAGATGCCCCGGGAG	NO	NO	Yes	NO	UbD
4	95206252	Rev	GTCATATAGATAAATAT	Yes	Yes_TP	Yes	NO	UbD	17	79577127	For	AAGTTCTTTGAACCACT	NO	NO	NO	Yes_rs8077038_0.23961661341853	UbD
4	90756550	For	TCACCTACCTACACATA	NO	NO	NO	NO	UbD	17	78081526	Rev	GCCCGCCGCTGTACCGT	Yes	Yes_TP	Yes	NO	UbD
4	88759690	For	AATACAACTCAGTGAGA	Yes	NO	Yes	NO	UbD	17	78081526	For	ACGGTACAGCGGCGGGC	Yes	Yes_TP	Yes	NO	UbD
4	88036182	For	GCTGCCGCCAAGCCGTG	NO	Yes_FP	NO	Yes_rs201122599_0.0386490474052216	UbD	17	76491127	Rev	TCATGCTCATGGCCGAG	Yes	Yes_TP	Yes	NO	UbD
4	85710824	For	ACACACACCCCCCAAAC	NO	NO	Yes	NO	UbD	17	74017594	Rev	AACCTGTGTATCTGCCA	NO	NO	NO	Yes_rs193119748_0.0313133934047271	UbD
4	84234190	For	AAAAAAAAA	NO	NO	Yes	NO	Ub	17	74017573	Rev	ACCCAGCTG	NO	NO	NO	Yes_rs78569674_0.122151469591109	Ub
4	69181899	For	AAACAAACAATTATTCC	NO	NO	Yes	NO	UbD	17	73258085	Rev	GCCCCCCCGCCGCTGCT	Yes	NO	Yes	Yes_rs10539706_0.0527101096224117	UbD
4	62775483	Rev	CACACACACACATAATT	NO	NO	NO	Yes_rs12648576_0.106351550960118	UbD	17	72959000	For	ACTGGCCCGGCCCCAAC	Yes	NO	Yes	Yes_rs7503218_0.255191693290735	UbD
4	57176966	Rev	AGGGGGGGG	NO	NO	Yes	NO	bD	17	72889676	Rev	ACGGAGCCCATGGAACC	NO	Yes_FP	Yes	NO	UbD
4	57176966	For	AAAAAAAAA	NO	NO	Yes	NO	Ub	17	72888541	For	CCCCCCGTG	Yes	NO	Yes	NO	bD
4	57176965	For	AAAAAAAAA	NO	NO	NO	NO	Ub	17	71468192	For	GTGGCTGGTGGGTGCCA	NO	NO	NO	NO	UbD
4	57176965	Rev	AGGGGGGGG	NO	NO	NO	NO	bD	17	66425995	For	CCCAAGGTTCCGGGATT	Yes	NO	Yes	NO	UbD
4	57176964	Rev	AGGGGGGGG	NO	NO	Yes	NO	bD	17	63014254	For	AAAAAAAAA	NO	NO	NO	NO	Ub
4	57176964	For	AAAAAAAAA	NO	NO	Yes	NO	Ub	17	58288545	Rev	CTAGGCATTTTTGCGCC	NO	NO	NO	NO	UbD
4	54280705	For	AAAAAAAAA	NO	NO	Yes	NO	bD	17	58288540	Rev	CATTTTTGCGCCCACAT	NO	NO	NO	NO	UbD
4	7066052	Rev	AGTAAATAG	Yes	NO	Yes	NO	bD	17	56569842	Rev	TAGTATGACAAAAAACA	Yes	Yes_TP	Yes	NO	UbD
4	7064003	For	CAGCTGGACCCCCCACC	NO	NO	Yes	NO	UbD	17	56569842	For	TGTTTTTTGTCATACTA	Yes	Yes_TP	Yes	NO	UbD
4	6698500	For	TGACAGCAA	Yes	NO	Yes	NO	bD	17	51902492	Rev	TTGGACAGT	NO	Yes_TP	Yes	NO	bD
4	4425422	Rev	GTGCCCCCTGTCTCTCC	NO	NO	Yes	NO	UbD	17	48157827	Rev	CACACACACCCATGCTA	Yes	NO	Yes	NO	UbD
4	4239539	For	AAAAACCACTCCAGATA	NO	NO	Yes	Yes_rs62286914_0.112890355417529	UbD	17	48148413	Rev	GGCCAATCCGTGTGTGC	Yes	NO	Yes	Yes_rs2301628_0.166533546325879	UbD
4	2502299	For	AGCCTATAACATTAGCA	Yes	NO	Yes	NO	UbD	17	47010795	Rev	TAAATTTTCTGTCAGTA	NO	NO	NO	NO	UbD
4	343795	For	CACAGGGCTCAGGAGCA	Yes	NO	Yes	NO	UbD	17	46673723	For	TGACGCCCTCGGCTCCC	NO	Yes_TP	Yes	NO	UbD
5	180046426	Rev	GTGGCGGGG	NO	NO	Yes	Yes_NA_0.000281531531531532	bD	17	45234657	Rev	AGTACCTAA	NO	NO	NO	Yes_rs78108688_0.0453615950292973	Ub
5	177679654	Rev	GATGGGTAGCGAAGGGA	Yes	NO	Yes	Yes_rs2033200_0.298123003194888	UbD	17	45234657	For	AACTATGATTAGGTACT	NO	NO	NO	Yes_rs78108688_0.0453615950292973	UbD
5	177679447	For	AGGACACCCGGAGGGAA	NO	NO	Yes	Yes_rs1549612_0.295926517571885	UbD	17	45234645	For	GTCTGTGAGATAAACTA	NO	Yes_FP	NO	Yes_rs75990396_0.0278173179093406	UbD
5	177676986	For	TGCACACGCACACACAC	NO	NO	Yes	Yes_rs5873601_0.104832268370607	UbD	17	45214604	Rev	TTAGGGCATGAGTTTGT	NO	NO	Yes	NO	UbD
5	176895817	Rev	CCTCGCCGCCTTCTCCA	Yes	Yes_TP	Yes	NO	UbD	17	45214604	For	ACAAACTCATGCCCTAA	NO	NO	Yes	NO	UbD
5	176895817	For	TGGAGAAGGCGGCGAGG	Yes	Yes_TP	Yes	NO	UbD	17	43333785	Rev	CTTCTTCTGTCTTAGAG	NO	NO	NO	NO	UbD
5	176858049	Rev	TCTGCTCCGGGGCAAGG	NO	NO	NO	Yes_rs867755_0.288738019169329	UbD	17	43333785	For	CTCTAAGACAGAAGAAG	NO	NO	NO	NO	UbD
5	175923431	For	AGTCTAAGGGTTGATCT	Yes	NO	Yes	NO	UbD	17	41960633	Rev	TGGCGCACGTGGCTGAG	Yes	Yes_TP	Yes	NO	UbD
5	175915835	For	GAACTGTACTTTAGAGG	Yes	NO	Yes	NO	UbD	17	41960633	For	CTCAGCCACGTGCGCCA	Yes	Yes_TP	Yes	NO	UbD
5	175528527	Rev	CTATGAGGTTCAGACCA	NO	NO	Yes	Yes_rs116633144_0.272416077738516	UbD	17	41121210	Rev	TCCCCCCCGCCCCCAGG	NO	Yes	NO	NO	UbD
5	175528527	For	TGGTCTGAACCTCATAG	NO	NO	Yes	Yes_rs116633144_0.272416077738516	UbD	17	41121209	Rev	TCCCCCCCGCCCCCAGG	NO	NO	NO	NO	UbD
5	175387139	Rev	TGTTTTCTGTAGGGGAC	NO	NO	NO	Yes_NA_2.68039026482256e-05	UbD	17	39742898	Rev	GGAGCCTGCGGGCTGGG	NO	NO	NO	NO	UbD
5	172659511	For	TTCTCCCCCCGAGAGTC	NO	NO	NO	Yes_rs703752_0.255591054313099	UbD	17	39619093	Rev	CTGGAGAAC	Yes	Yes_TP	Yes	NO	Ub
5	160763821	Rev	TTTTTTTTG	NO	NO	NO	Yes_NA_0.00605577111360917	Ub	17	39619093	For	CAGTCCTCGTTCTCCAG	Yes	Yes_TP	Yes	NO	UbD
5	153709043	For	GAAGTGGGGTTTGCAGT	NO	NO	Yes	NO	UbD	17	39340843	Rev	CCACTCTGCTGTCAGAC	NO	NO	NO	Yes_rs2320229_0.108966376089664	UbD
5	150518358	Rev	TCCCCACCTGCCAGGAC	Yes	Yes_TP	Yes	NO	UbD	17	39254142	For	CTGGGGCGACAGCAGGT	NO	NO	NO	Yes_rs375280428_0.00623700623700624	UbD
5	149006518	For	GGGCCCTTCAGTGGCAG	Yes	NO	Yes	Yes_rs66982663_0.2935303514377	UbD	17	38153714	Rev	AGAGCAGCTCAGCCCTT	Yes	Yes_TP	Yes	NO	UbD
5	148586451	For	CAGTGTCAGGTGGGAGC	Yes	NO	Yes	NO	UbD	17	38153714	For	AAGGGCTGAGCTGCTCT	Yes	Yes_TP	Yes	NO	UbD
5	139931629	Rev	TCTGAGGCTGTGATGGA	Yes	Yes_TP	Yes	Yes_rs5871740_0.262013729977117	UbD	17	37829570	Rev	GGGGTGGCC	Yes	NO	Yes	NO	bD
5	139931629	For	TCCATCACAGCCTCAGA	Yes	Yes_TP	Yes	Yes_rs5871740_0.262013729977117	UbD	17	36734706	For	CGCAGCCGGAGCCGCAG	NO	NO	Yes	NO	UbD
5	139931628	Rev	CTGAGGCTGTGATGGAG	Yes	Yes_TP	Yes	Yes_NA_1.96436639361974e-05	UbD	17	36070592	Rev	GCCATGGTGACCAGCCA	NO	NO	NO	Yes_NA_0.000138127906441365	UbD
5	139931628	For	CTCCATCACAGCCTCAG	Yes	Yes_TP	Yes	Yes_NA_1.96436639361974e-05	UbD	17	36070591	Rev	CCATGGTGACCAGCCAG	NO	NO	NO	Yes_NA_0.00202336848104873	UbD
5	137503739	Rev	TGTGGCTTC	NO	NO	NO	Yes_rs201078605_0.267831596157776	Ub	17	34951327	For	ACGGGGGTCGGGGTCTG	Yes	NO	Yes	Yes_rs2285639_0.275958466453674	UbD
5	133537479	For	TAAGCACCTGAACACTA	Yes	NO	Yes	NO	UbD	17	34867415	Rev	AGATGGGCACTTGGGCC	NO	NO	Yes	NO	UbD
5	131322666	Rev	GGGTATGTG	Yes	NO	Yes	NO	bD	17	31323728	For	AGGTGGGAGGCACCTGC	NO	NO	Yes	NO	UbD
5	118513246	Rev	AAACTAAAAGTCTTTCT	Yes	NO	Yes	NO	UbD	17	26708304	Rev	GATGCCTGACCCACTCC	NO	Yes	NO	Yes_rs71373647_0.0833333333333333	UbD
5	108679870	For	ATTATTCTA	Yes	Yes_TP	Yes	NO	bD	17	26708304	For	GGAGTGGGTCAGGCATC	NO	Yes	NO	Yes_rs71373647_0.0833333333333333	UbD
5	96237114	For	TCCCCAAGCCTTGTTTC	Yes	NO	Yes	NO	UbD	17	26708302	Rev	TGCCTGACCCACTCCGC	NO	Yes_TP	Yes	Yes_rs71373646_9.07951842234288e-06	UbD
5	86667856	For	TGAATTTGAAAAAAAAA	NO	NO	Yes	NO	UbD	17	26708302	For	GCGGAGTGGGTCAGGCA	NO	Yes_TP	Yes	Yes_rs71373646_9.07951842234288e-06	UbD
5	70357189	Rev	TTTTTTTTGTGGGGGGG	NO	NO	NO	NO	UbD	17	26708300	Rev	CCTGACCCACTCCGCGC	NO	Yes_FP	NO	Yes_rs71373645_0.0769230769230769	UbD
5	54706325	For	TCTGACATTTTATTCTT	NO	Yes_TP	Yes	NO	UbD	17	26708300	For	GCGCGGAGTGGGTCAGG	NO	Yes_FP	NO	Yes_rs71373645_0.0769230769230769	UbD
5	54579211	Rev	GTGGCAGGTGAAACAGG	NO	NO	Yes	Yes_NA_0.0270407792396598	UbD	17	21319767	Rev	GAGTTGGCGCTGGGCAG	NO	NO	Yes	Yes_rs1612176_0.253943956503555	UbD
5	54579211	For	CCTGTTTCACCTGCCAC	NO	NO	Yes	Yes_NA_0.0270407792396598	UbD	17	21319767	For	CTGCCCAGCGCCAACTC	NO	NO	Yes	Yes_rs1612176_0.253943956503555	UbD
5	43294357	Rev	TTCTACTTTTACTTATA	Yes	NO	Yes	NO	UbD	17	21319523	Rev	AGTCGTCCGTCTCCAGG	NO	NO	Yes	Yes_rs77987694_0.298576883026727	UbD
5	41070840	For	TGACGCGCCACTCCCAG	Yes	NO	Yes	Yes_rs1423391_0.281948881789137	UbD	17	21319523	For	CCTGGAGACGGACGACT	NO	NO	Yes	Yes_rs77987694_0.298576883026727	UbD
5	41069734	For	AATATATACGAAATGTT	Yes	NO	Yes	Yes_rs10512757_0.193690095846645	UbD	17	21319519	Rev	GTCCGTCTCCAGGTCCT	NO	NO	Yes	Yes_rs78113532_0.289852512585004	UbD
5	36301362	For	TTCTTCCTG	Yes	NO	Yes	Yes_rs10941287_0.251996805111821	bD	17	21319519	For	AGGACCTGGAGACGGAC	NO	NO	Yes	Yes_rs78113532_0.289852512585004	UbD
5	36217764	For	AAAACAAGTAAATTATT	Yes	NO	Yes	NO	UbD	17	21319488	Rev	AAGAGCGGGCTGGCCTC	NO	NO	Yes	Yes_rs77176173_0.166103653658103	UbD
5	35763916	Rev	AGGTACAATTTTCTACA	Yes	NO	Yes	NO	UbD	17	21319488	For	GAGGCCAGCCCGCTCTT	NO	NO	Yes	Yes_rs77176173_0.166103653658103	UbD

5	34863220	Rev	GAGATGACCCATCTGGC	Yes	NO	Yes	Yes_rs3846632_0.191892971246006	UbD	17	21319470	Rev	TCAATCTCATGCAAGAT	NO	NO	Yes	Yes_NA_0.176899415017514	UbD
5	13885023	For	CACACGTGCACACACAC	NO	NO	Yes	NO	UbD	17	21319470	For	ATCTTGCATGAGATTGA	NO	NO	Yes	Yes_NA_0.176899415017514	UbD
5	1337982	For	AGTGTCTCG	NO	NO	NO	Yes_NA_0.078436018957346	Ub	17	21319465	Rev	CTCATGCAAGATGGTGA	NO	NO	Yes	Yes_NA_0.183828376794609	UbD
5	1225687	Rev	CAAACCCAGTGCAGCTC	NO	Yes_TP	Yes	NO	UbD	17	21319465	For	TCACCATCTTGCATGAG	NO	NO	Yes	Yes_NA_0.183828376794609	UbD
5	1225687	For	GAGCTGCACTGGGTTTG	NO	Yes_TP	Yes	NO	UbD	17	21319452	Rev	GTGATGGGCGACACCAG	NO	NO	Yes	Yes_rs73313929_0.189061693815877	UbD
5	834075	Rev	CGTGCACACGTAAGTGG	NO	NO	Yes	NO	UbD	17	21319452	For	CTGGTGTCGCCCATCAC	NO	NO	Yes	Yes_rs73313929_0.189061693815877	UbD
5	174240	Rev	GAGCCTGGCAGGCCCTG	NO	NO	Yes	NO	UbD	17	21319439	Rev	CCAGAAAGATGCGGTCC	NO	Yes_FP	Yes	Yes_rs76684759_0.197978702425557	UbD
6	170871040	Rev	TGTTGCTGTTGCTGCTG	NO	NO	NO	Yes_rs55736770_0.00836804522463552	UbD	17	21319439	For	GGACCGCATCTTTCTGG	NO	Yes_FP	Yes	Yes_rs76684759_0.197978702425557	UbD
6	170871040	For	CAGCAGCAACAGCAACA	NO	NO	NO	Yes_rs55736770_0.00836804522463552	UbD	17	21319436	Rev	GAAAGATGCGGTCCAGG	NO	NO	Yes	Yes_rs77270326_0.18116224648986	UbD
6	170871037	Rev	TGTTGCTGTTGCTGCTG	NO	NO	NO	NO	UbD	17	21319436	For	CCTGGACCGCATCTTTC	NO	NO	Yes	Yes_rs77270326_0.18116224648986	UbD
6	170871037	For	CAGCAGCAACAGCAACA	NO	NO	NO	NO	UbD	17	21318603	For	GCAGGAGCCGCCCTGCC	NO	Yes_TP	Yes	NO	UbD
6	168476571	Rev	GCCGGAAAT	NO	NO	NO	NO	bD	17	21318588	For	GCTGTCGTCTCTGTTGC	NO	NO	Yes	NO	UbD
6	168377010	Rev	CTCCTCCCCACACACTG	NO	NO	NO	Yes_rs9364374_0.0455223880597015	UbD	17	21318586	For	GTCTCTGTT	NO	NO	Yes	NO	bD
6	168377010	For	CAGTGTGTGGGGAGGAG	NO	NO	NO	Yes_rs9364374_0.0455223880597015	UbD	17	21217660	Rev	GCCCCCTCGGTTCCTCC	NO	Yes_TP	Yes	NO	UbD
6	168376953	For	GGGTCATTACCCCTGCA	NO	NO	NO	Yes_rs2516607_0.245276292335116	UbD	17	21217596	Rev	CTGCCCCCGCAGATGGG	NO	Yes_TP	Yes	Yes_rs2363373_0.000199680511182109	UbD
6	168376951	For	GGGGGTCATTACCCCTG	NO	NO	NO	Yes_rs34562778_0.0277569392348087	UbD	17	21217400	For	GGGGAAGAGTGGCCACC	NO	NO	Yes	Yes_rs1622401_0.00259584664536741	UbD
6	168376908	For	GGGGGTCATTCCCCCTG	NO	NO	NO	Yes_rs2516605_0.00964868876793666	UbD	17	21217397	For	GTTGGGGAAGAGTGGCC	NO	NO	Yes	Yes_rs1657685_0.00259584664536741	UbD
6	168376892	For	GGGAGGAGAAGGCAGTG	NO	NO	NO	Yes_rs9355147_0.00667388167388167	UbD	17	21217290	For	GAGCCACAGATGCCATC	NO	NO	Yes	NO	UbD
6	167786961	Rev	AGCCGCGCCCACTGACT	NO	NO	Yes	NO	UbD	17	21216710	For	CCTGCTGCTAGGCCTGG	NO	NO	Yes	NO	UbD
6	167786750	Rev	CATCCAAAGGGCAAGGC	NO	NO	Yes	Yes_rs143094588_0.192375214459562	UbD	17	21216686	For	CTTCTTCCCTGGGTGCA	NO	NO	Yes	NO	UbD
6	167786750	For	GCCTTGCCCTTTGGATG	NO	NO	Yes	Yes_rs143094588_0.192375214459562	UbD	17	21216664	For	GAAGCCCTAGACAGTCC	NO	NO	Yes	NO	UbD
6	154567666	For	TCACCTGATGCCTTTAA	NO	NO	NO	Yes_rs9479798_0.283945686900958	UbD	17	21216661	For	GAAGAAGCCCTAGACAG	NO	NO	Yes	NO	UbD
6	152489037	Rev	GCTGGACAAAAAGGTTA	NO	NO	NO	Yes_rs997389_0.290716519062976	UbD	17	21215700	Rev	GCCAAGTCATCTGGGGG	NO	NO	Yes	NO	UbD
6	152489037	For	TAACCTTTTTGTCCAGC	NO	NO	NO	Yes_rs997389_0.290716519062976	UbD	17	21215682	Rev	TTATTTGCCAAAGGAAT	NO	NO	Yes	NO	UbD
6	151674116	Rev	TCTAAATCCTCAATTGC	NO	Yes_TP	Yes	NO	UbD	17	21208292	For	TTTGGGGATTGGGAGGC	NO	NO	Yes	NO	UbD
6	151674116	For	GCAATTGAGGATTTAGA	NO	Yes_TP	Yes	NO	UbD	17	21208203	For	GGGTTGGGCCTTCTAAG	NO	NO	Yes	NO	UbD
6	147635318	For	GATTATTTACACACACA	NO	NO	Yes	NO	UbD	17	21207992	Rev	CCCCACTACAGCAGGAA	NO	NO	NO	NO	UbD
6	136582194	For	AATCAGGTAAAAAAAAT	NO	NO	Yes	NO	UbD	17	21207989	Rev	CACTACAGCAGGAAACG	NO	NO	Yes	Yes_rs9893916_0.0181709265175719	UbD
6	136582113	For	AAGACTTAAAACAAAAT	NO	NO	Yes	NO	UbD	17	21207945	Rev	ACAGAGGGTATGGCCGG	NO	NO	Yes	NO	UbD
6	128030957	For	TATCATAGGTTTCTGTA	NO	Yes_TP	Yes	Yes_rs490008_0.159744408945687	UbD	17	21207649	For	GTCTTGGGCGAGGGAGG	NO	NO	Yes	NO	UbD
6	125139473	For	TGTGTGGGTGCTTGATC	NO	NO	Yes	NO	UbD	17	21207602	For	CCCTGGTGCAACCTGCC	NO	NO	Yes	NO	UbD
6	123539684	For	CTGAACTACTGTGGACA	Yes	Yes_TP	Yes	NO	UbD	17	21207582	For	CCCTGGTCTGACCCTTG	NO	NO	Yes	NO	UbD
6	123038887	For	AACATTTACTTCTGAAT	Yes	Yes_TP	Yes	NO	UbD	17	21206654	Rev	CGTCACAACCTATCACC	NO	NO	Yes	NO	UbD
6	121560155	For	AGTATGAACGAATTCAT	Yes	NO	Yes	Yes_rs12191616_0.260183706070288	UbD	17	21206642	Rev	TCACCTCTCGGAGCCAC	NO	NO	Yes	Yes_rs59259996_0.00419329073482428	UbD
6	119627976	For	AGGGCCATGTTACACAG	Yes	NO	Yes	Yes_rs6915559_0.292332268370607	UbD	17	21206635	Rev	TCGGAGCCACAGTTGCT	NO	NO	Yes	NO	UbD
6	114178831	For	GCTGCTGCT	NO	NO	Yes	NO	bD	17	21206616	Rev	TAGGGAGACGTCCACCC	NO	NO	Yes	Yes_rs57761454_0.0023961661341853	UbD
6	112508769	Rev	ACTGATGCACTGCGGTT	Yes	Yes_TP	Yes	NO	UbD	17	21206385	For	CTGTGCAGCGGGAGCGG	NO	NO	Yes	Yes_rs72838547_0.00858626198083067	UbD
6	112508769	For	AACCGCAGTGCATCAGT	Yes	Yes_TP	Yes	NO	UbD	17	21206372	For	GAGGGTGCGTAGCCTGT	NO	NO	Yes	NO	UbD
6	109850199	For	AAAAAAAACAAAACTAC	NO	NO	NO	Yes_rs577355457_0.231803826862344	UbD	17	21206360	For	AGGCTGGGATGAGAGGG	NO	NO	Yes	Yes_rs67280439_0.000199680511182109	UbD
6	108243123	Rev	TTTTTTTTGTTTTTCCC	NO	NO	NO	Yes_rs2064201_0.0894568690095847	UbD	17	21205655	Rev	CACCATCAATAGGCAGG	NO	NO	Yes	NO	UbD
6	108243122	Rev	TTTTTTTTGTTTTTCCC	NO	NO	NO	Yes_rs2064201_0.0894568690095847	UbD	17	21205654	Rev	ACCATCAATAGGCAGGC	NO	NO	Yes	NO	UbD
6	108197874	Rev	TTTTTTTTG	NO	Yes	NO	NO	Ub	17	21205642	Rev	CAGGCTCCATGCCTGCT	NO	NO	Yes	NO	UbD
6	107372242	For	TCTGTTTTCTTTTAATT	NO	NO	NO	NO	UbD	17	21205592	Rev	GACCCCCCGCCAGCCCA	NO	NO	Yes	Yes_NA_0.000938086303939962	UbD
6	105300262	Rev	GAACTCTTTGTTTTTTT	NO	Yes_TP	Yes	NO	UbD	17	21205396	For	AGGCAGTGCAGGTGGTG	NO	NO	Yes	NO	UbD
6	84290137	For	CAGTTAAAGAAAAAAAA	NO	NO	NO	Yes_NA_0.00298730395817774	UbD	17	21205361	For	GAGAGGGGGCTGGGGCT	NO	NO	Yes	NO	UbD
6	79595167	Rev	TTTGTTAAGTACATACC	Yes	Yes_TP	Yes	NO	UbD	17	21205360	For	CGGAGAGGGGGCTGGGG	NO	NO	Yes	NO	UbD
6	70778378	Rev	TGTTCCAGCTTTACTTA	NO	NO	NO	NO	UbD	17	21205352	For	CACACGTCGGAGAGGGG	NO	NO	Yes	NO	UbD
6	70778378	For	TAAGTAAAGCTGGAACA	NO	NO	NO	NO	UbD	17	21204441	Rev	AACAGCCAAGAGACGGG	NO	NO	Yes	NO	UbD
6	64708907	For	CATAAAAAATACTTATC	Yes	NO	Yes	Yes_rs66502009_0.269768370607029	UbD	17	21204437	Rev	GCCAAGAGACGGGGGAG	NO	NO	Yes	NO	UbD
6	57512826	Rev	GGGAAGGTTGAGGCTGC	NO	NO	Yes	NO	UbD	17	21204318	Rev	CTGCATCATGCAAGGAC	NO	NO	Yes	Yes_rs74869330_0.122158098691104	UbD
6	57512825	Rev	GGAAGGTTGAGGCTGCA	NO	NO	Yes	NO	UbD	17	21204318	For	GTCCTTGCATGATGCAG	NO	NO	Yes	Yes_rs74869330_0.122158098691104	UbD
6	57512792	Rev	TCACAATTACACAACAG	NO	NO	Yes	NO	UbD	17	21204315	Rev	CTGCATCATGCAAGGAC	NO	NO	Yes	NO	UbD
6	57512779	Rev	ACAGAGTGCAACACTTT	NO	NO	Yes	NO	UbD	17	21204315	For	GTCCTTGCATGATGCAG	NO	NO	Yes	NO	UbD
6	57512775	Rev	AGTGCAACACTTTTTCA	NO	NO	Yes	NO	UbD	17	21204074	Rev	CTAGCCTCG	NO	NO	Yes	NO	bD
6	57512352	For	AGAATGCAAACAGCTAG	NO	NO	Yes	NO	UbD	17	21204074	For	CGAGGCTAGGCTTTTTG	NO	NO	Yes	NO	UbD
6	57512340	For	TGATTAAGCGTTAGAAT	NO	NO	Yes	NO	UbD	17	21204067	Rev	CCTAGCCTCGGTACTGA	NO	NO	Yes	NO	UbD
6	57499188	Rev	GTCTGGTCGTTCAATAA	NO	NO	Yes	NO	UbD	17	21204067	For	TCAGTACCGAGGCTAGG	NO	NO	Yes	NO	UbD
6	57499171	Rev	ATGCTGTAAGTTACTTA	NO	NO	Yes	NO	UbD	17	21203711	For	CTGAGGGTCACACAGCA	NO	NO	Yes	NO	UbD
6	57498894	For	GGCCAACAGCTGTGAGC	NO	NO	Yes	NO	UbD	17	21202370	Rev	CGAGGCTCT	NO	NO	Yes	NO	bD
6	57472534	Rev	TGACAAGTAATACACAG	NO	NO	Yes	NO	UbD	17	21202318	Rev	CCTCCCCCCAAAGTGGC	NO	NO	Yes	Yes_rs376150010_0.0291533546325879	UbD
6	57472533	Rev	GACAAGTAATACACAGC	NO	NO	Yes	NO	UbD	17	21202316	Rev	TCCCCCCAAAGTGGCCC	NO	NO	Yes	NO	UbD
6	57472495	Rev	AAAACCGAGAGATCTGT	NO	NO	Yes	NO	UbD	17	21202311	Rev	CCAAAGTGGCCCTGCAG	NO	NO	Yes	NO	UbD
6	57472315	Rev	TTATTTACTCACAACTT	NO	NO	Yes	NO	UbD	17	21202310	Rev	CAAAGTGGCCCTGCAGG	NO	NO	Yes	NO	UbD
6	57472315	For	AAGTTGTGAGTAAATAA	NO	NO	Yes	NO	UbD	17	21202102	Rev	GGCCTGGCG	NO	NO	Yes	Yes_rs35701021_0.00619009584664537	Ub
6	57472299	Rev	TAGTACAATACCACACT	NO	NO	Yes	NO	UbD	17	21202102	For	CGTCCCCACGCCAGGCC	NO	NO	Yes	Yes_rs35701021_0.00619009584664537	UbD
6	57472299	For	AGTGTGGTATTGTACTA	NO	NO	Yes	NO	UbD	17	21202078	For	GCGCCTCCGGGGCAGGA	NO	NO	Yes	NO	UbD
6	57472255	For	TAATAAGCCGTTAGACA	NO	NO	Yes	NO	UbD	17	21202067	For	GGCCGGAGAAGGCGCCT	NO	NO	Yes	NO	UbD
6	57472212	For	AAGAGGAATGCAGTTGC	NO	NO	Yes	NO	UbD	17	21202063	For	AGGTGGCCGGAGAAGGC	NO	NO	Yes	NO	UbD
6	57472201	For	ATTTAGATAAAAAGAGG	NO	NO	Yes	NO	UbD	17	21202056	For	TGAGAGGAGGTGGCCGG	NO	NO	Yes	NO	UbD
6	57467323	Rev	CATAAGCTA	NO	NO	Yes	NO	bD	17	21202015	For	ATGAGGCCGTGCCCAGC	NO	NO	Yes	Yes_rs34956489_0.000599041533546326	UbD
6	57467317	Rev	GACATAAGCTACAAACT	NO	NO	Yes	NO	UbD	17	21201981	For	ACTTCTCAC	NO	NO	Yes	Yes_rs8072533_0.000199680511182109	bD
6	57467303	Rev	ACTAGGTCTTGACAGGA	NO	NO	Yes	NO	UbD	17	21201934	Rev	CAGCCGTCACCTGGCTG	NO	NO	Yes	Yes_rs4986015_0.0483226837060703	UbD
6	57467282	Rev	AAAAAGCCACAACATTG	NO	NO	Yes	NO	UbD	17	21201894	Rev	GACAGGACGAGGTAAAG	NO	NO	Yes	Yes_rs8071865_0.000199680511182109	UbD
6	57467279	Rev	AAGCCACAACATTGTTT	NO	NO	Yes	NO	UbD	17	21201890	Rev	GGACGAGGTAAAGCAGG	NO	NO	Yes	NO	UbD
6	57466987	For	GAAAATATTGCTTTGCA	NO	NO	Yes	NO	UbD	17	21201851	Rev	TCCTTCCATTGAGCCCT	NO	NO	Yes	Yes_rs8072386_0.000599041533546326	UbD
6	57466969	For	AATGCTGAATTTATTCT	NO	NO	Yes	NO	UbD	17	21201851	For	AGGGCTCAATGGAAGGA	NO	NO	Yes	Yes_rs8072386_0.000599041533546326	UbD
6	57466958	For	ATCTGCCCATGAATGCT	NO	NO	Yes	NO	UbD	17	20908081	For	GCCTTTCCTAGTGGAGC	NO	NO	NO	NO	UbD
6	57398081	For	TTACAGGGTACCTGATA	NO	NO	Yes	NO	UbD	17	20768850	Rev	ATTCCCTTCATTTTTTT	NO	NO	Yes	Yes_rs114103783_0.151664551400957	UbD
6	57398058	For	TGAATTAGCAACCATAA	NO	NO	Yes	NO	UbD	17	20768816	Rev	AGAGTATTGTCATCTGC	NO	NO	NO	Yes_rs73298040_0.0849113375591752	UbD
6	57398047	For	TCCAATTAAATTGAATT	NO	NO	Yes	NO	UbD	17	20768803	Rev	CTGCATGAGCAAAGGGT	NO	Yes_FP	NO	Yes_rs76135364_0.0476850724176051	UbD
6	57398009	For	AATAAAGCAGCACTTTC	NO	NO	Yes	NO	UbD	17	20768803	For	ACCCTTTGCTCATGCAG	NO	Yes_FP	NO	Yes_rs76135364_0.0476850724176051	UbD
6	57398008	For	TAATAAAGCAGCACTTT	NO	NO	Yes	NO	UbD	17	20768788	Rev	GTTCATCACCACATCTT	NO	Yes_FP	NO	Yes_rs78365129_0.026876285425454	UbD
6	57397982	For	CAGATAAGCGTCATCTG	NO	NO	Yes	NO	UbD	17	20768788	For	AAGATGTGGTGATGAAC	NO	Yes_FP	NO	Yes_rs78365129_0.026876285425454	UbD
6	57393278	Rev	GTGAAATAATATTCTAT	NO	NO	Yes	NO	UbD	17	20768744	Rev	TCATGTGTCTAATGTTG	NO	NO	NO	Yes_rs62066974_0.0111197912338148	UbD
6	57372480	Rev	TTGAATAATGGAATGAA	NO	NO	Yes	NO	UbD	17	20768744	For	CAACATTAGACACATGA	NO	NO	NO	Yes_rs62066974_0.0111197912338148	UbD
6	57372437	Rev	TAATTGAAGCGGTTTCA	NO	NO	Yes	NO	UbD	17	20768730	Rev	TTGTAAGGTAAACATTC	NO	NO	NO	Yes_rs4605228_0.054382927398646	UbD
6	57372436	Rev	AATTGAAGCGGTTTCAC	NO	NO	Yes	NO	UbD	17	20768730	For	GAATGTTTACCTTACAA	NO	NO	NO	Yes_rs4605228_0.054382927398646	UbD
6	57372199	For	CTCCCTATTTGCCCTTC	NO	NO	Yes	NO	UbD	17	18541296	Rev	CTTGGCCCCGCCCTACA	NO	NO	Yes	Yes_rs2386412_0.121642631212137	UbD
6	57372126	For	CAGGTTTTT	NO	NO	Yes	NO	bD	17	18395647	For	GAACCAAGGAAAGATAT	NO	NO	Yes	NO	UbD
6	57246773	For	GAAGACTTAGGATTTCT	NO	NO	Yes	NO	UbD	17	17770355	Rev	CCTAGCCCCCAGGTGGG	Yes	NO	Yes	Yes_rs11657074_0.281749201277955	UbD
6	57246708	For	ATCATTTTGATATTCTT	NO	NO	Yes	Yes_rs6913779_0.00319488817891374	UbD	17	15639142	Rev	GGCTGGACCGACAGGAG	NO	NO	NO	NO	UbD
6	57244893	Rev	ATGATGAGA	NO	NO	Yes	NO	bD	17	14095639	Rev	AGAATTAATGTGGGATG	NO	NO	Yes	NO	UbD
6	52847273	For	ACACACACA	NO	NO	Yes	NO	bD	17	14095639	For	CATCCCACATTAATTCT	NO	NO	Yes	NO	UbD
6	47649573	Rev	AACAATTTGCTTGGCCT	Yes	Yes_TP	Yes	NO	UbD	17	13977813	Rev	AGAGAAAAAGGCAGAAA	Yes	Yes_TP	Yes	NO	UbD
6	47649573	For	AGGCCAAGCAAATTGTT	Yes	Yes_TP	Yes	NO	UbD	17	10357281	Rev	GTCACTGAAATTTCCAC	NO	NO	NO	Yes_rs3826444_0.29452875399361	UbD
6	43014298	Rev	ACCCACCAACCCATCAA	Yes	Yes_TP	Yes	NO	UbD	17	10258155	For	TTTTTTTTT	NO	NO	Yes	Yes_rs5010940_0.00439297124600639	bD

6	43014298	For	TTGATGGGTTGGTGGGT	Yes	Yes_TP	Yes	NO	UbD
6	41774685	Rev	TACGGCTCGCCCAGGAC	NO	Yes_FP	NO	Yes_rs201585090_0.270616093152589	UbD
6	41754573	For	GAACACTCCTCGCTGCT	Yes	Yes_TP	Yes	NO	UbD
6	41196104	For	TCTGACTCCTCCTGAGA	Yes	Yes_TP	Yes	NO	UbD
6	39866805	Rev	ACCCATTGTTTTTTGCC	Yes	NO	Yes	NO	UbD
6	32916540	For	CAGAGACTTCTACCCTA	Yes	Yes_TP	Yes	NO	UbD
6	31922368	Rev	ATCGGGAGCGGGATCGA	NO	NO	NO	Yes_NA_0.000881560847073412	UbD
6	31922368	For	TCGATCCCGCTCCCGAT	NO	NO	NO	Yes_NA_0.000881560847073412	UbD
6	31922365	Rev	GGGAGCGGGATCGAGAC	NO	NO	NO	Yes_NA_0.00156739811912226	UbD
6	31922365	For	GTCTCGATCCCGCTCCC	NO	NO	NO	Yes_NA_0.00156739811912226	UbD
6	31922363	Rev	GAGCGGGATCGAGACCG	NO	NO	NO	Yes_NA_0.00209200949675009	UbD
6	31922363	For	CGGTCTCGATCCCGCTC	NO	NO	NO	Yes_NA_0.00209200949675009	UbD
6	31922360	Rev	CGGGATCGAGACCGAGA	NO	NO	NO	Yes_NA_0.00228345698044128	UbD
6	31922360	For	TCTCGGTCTCGATCCCG	NO	NO	NO	Yes_NA_0.00228345698044128	UbD
6	31922356	Rev	ATCGAGACCGAGACCGA	NO	Yes_FP	NO	Yes_NA_0.0036465416986971	UbD
6	31922356	For	TCGGTCTCGGTCTCGAT	NO	Yes_FP	NO	Yes_NA_0.0036465416986971	UbD
6	24547650	Rev	CCCATCACAGGACCTGC	NO	NO	NO	Yes_rs34829630_0.287739616613419	UbD
6	10887251	For	GGGCTCTTTCGGCAGCG	Yes	Yes_TP	Yes	NO	UbD
6	6589192	Rev	CTCTCCCCTCCCACCCC	NO	NO	Yes	NO	UbD
6	6318921	Rev	TTTTTTTTCTGAAGGAC	NO	Yes_FP	Yes	Yes_rs202215504_0.246373112939005	UbD
6	3264526	Rev	TGTGGGAAGAGTGGAGC	Yes	Yes_TP	Yes	NO	UbD
6	3264526	For	GCTCCACTCTTCCCACA	Yes	Yes_TP	Yes	NO	UbD
6	350940	Rev	TTAAACTCTATGTTGCT	NO	NO	Yes	NO	UbD
6	348051	For	GGGCGATGAACCCGGAG	NO	Yes_TP	Yes	NO	UbD
6	335268	Rev	CAGCAAGGGTCCCGTGA	NO	NO	Yes	NO	UbD
6	335251	Rev	GCTCTGACAACTTCACA	NO	NO	Yes	NO	UbD
7	157691523	Rev	GTCATGTAATTGAAATT	Yes	NO	Yes	Yes_rs56112490_0.237819488817891	UbD
7	157370645	For	TCCCATTCGGAGGAGGA	NO	NO	Yes	Yes_rs736940_0.289536741214057	UbD
7	157367205	Rev	CTCTCTCTCATTATTAC	NO	NO	Yes	NO	UbD
7	156759597	For	GTGATATATTATAGTAA	Yes	NO	Yes	NO	UbD
7	152513769	Rev	TTGTATTTCCATTACAG	NO	NO	NO	NO	UbD
7	151971052	Rev	ATTTGTTTTACTTGTGA	NO	NO	Yes	NO	UbD
7	151971043	Rev	CTTGTGATATACAGAGA	NO	NO	Yes	NO	UbD
7	151970672	For	GATTTTAAAGATTATGT	NO	NO	Yes	NO	UbD
7	151962068	For	GCACCTAGTAATAGGGT	NO	NO	Yes	NO	UbD
7	151552663	For	ACACACACA	NO	NO	NO	NO_151552663	bD
7	151552231	For	ACACACACA	NO	NO	NO	NO	bD
7	148904343	For	CACCCAGGAGGAGAGGG	Yes	NO	Yes	NO	UbD
7	143095256	Rev	CATTCTGCTCTCCAGCC	NO	NO	NO	Yes_rs35251323_0.167332268370607	UbD
7	143094635	For	GGTTGGGGAGGGGGCAG	NO	NO	NO	Yes_rs372030742_0.0153925265683922	UbD
7	143088526	For	CACCCTGATTGGGCATG	NO	NO	NO	Yes_rs1131885_0.279725739112725	UbD
7	142498708	For	GCCCCATTACCTCTTCC	NO	Yes_TP	Yes	NO	UbD
7	142498707	For	AGCCCCATTACCTCTTC	NO	NO	NO	Yes_rs371096403_2.90587859239241e-05	UbD
7	142498706	For	TAGCCCCATTACCTCTT	NO	NO	NO	NO	UbD
7	142480126	Rev	GGGCTGAAGCCAAGCTC	NO	NO	Yes	NO	UbD
7	142479841	For	GGAAGCAAACGCAGGCT	NO	NO	Yes	NO	UbD
7	142479819	For	TGTTAAGGA	NO	NO	Yes	NO	Ub
7	142479787	For	GAGCTCCCTCCCTTGCC	NO	NO	NO	NO_142479787	UbD
7	142460503	Rev	TCAGCCCCACCACCTTT	NO	NO	Yes	NO	UbD
7	142460495	Rev	ACCACCTTTTGAGTTCA	NO	NO	NO	NO_142460495	UbD
7	142460494	Rev	CCACCTTTTGAGTTCAA	NO	NO	Yes	Yes_rs564920652_0.000199680511182109	UbD
7	142460482	Rev	TTCAAATCCTTTTCCCT	NO	NO	Yes	NO	UbD
7	142460461	Rev	GGGCCTAGGTATCAGTG	NO	NO	Yes	NO	UbD
7	142460461	For	CACTGATACCTAGGCCC	NO	NO	Yes	NO	UbD
7	142460238	Rev	ATGTGGGTCAGGCCAGA	NO	NO	Yes	Yes_NA_0.0178694522509147	UbD
7	142460238	For	TCTGGCCTGACCCACAT	NO	NO	Yes	Yes_NA_0.0178694522509147	UbD
7	142460226	For	ATTGTCTCCTTCTCTGG	NO	NO	Yes	NO	UbD
7	142460216	For	ATCCAAGATTATTGTCT	NO	NO	Yes	Yes_rs549425727_0.000798722044728434	UbD
7	142460212	For	TTCCATCCAAGATTATT	NO	NO	Yes	NO	UbD
7	142460203	For	TTTCCATCC	NO	NO	Yes	NO	bD
7	142460201	For	GTTTTCCAT	NO	NO	Yes	NO	bD
7	142460200	For	TTTCCATCC	NO	NO	Yes	NO	bD
7	142458338	For	GAAAGCAATCACAGGCT	NO	NO	Yes	NO	UbD
7	142458337	For	AGAAAGCAATCACAGGC	NO	NO	Yes	NO	UbD
7	142458331	For	ATTAGCAGAAAGCAATC	NO	NO	Yes	NO	UbD
7	142458316	For	TGTTAAGGATTTCTAAT	NO	NO	Yes	NO	UbD
7	142458304	For	CCTCACTGTGCTTGTTA	NO	NO	Yes	NO	UbD
7	142458284	For	GAGCTCCCTCCCTTGCC	NO	NO	NO	NO	UbD
7	142458277	For	GGTGGCAGAGCTCCCTC	NO	NO	NO	NO	UbD
7	142458270	For	CACCAGGGGTGGCAGAG	NO	NO	NO	NO	UbD
7	142247571	Rev	GCCCTGACTCTGTCATG	NO	NO	Yes	Yes_rs376050671_0.106876383612561	UbD
7	142247571	For	CATGACAGAGTCAGGGC	NO	NO	Yes	Yes_rs376050671_0.106876383612561	UbD
7	142247567	Rev	TGACTCTGTCATGGGCA	NO	NO	Yes	Yes_rs372792593_0.0816339455351488	UbD
7	142247567	For	TGCCCATGACAGAGTCA	NO	NO	Yes	Yes_rs372792593_0.0816339455351488	UbD
7	142247554	Rev	GGCACCAGGCTCCTCTG	NO	NO	Yes	Yes_rs10265119_0.0958900638535086	UbD
7	142247554	For	CAGAGGAGCCTGGTGCC	NO	NO	Yes	Yes_rs10265119_0.0958900638535086	UbD
7	142247541	Rev	TCTGCTGGGCAGCCCTG	NO	NO	Yes	NO	UbD
7	142247541	For	CAGGGCTGCCCAGCAGA	NO	NO	Yes	NO	UbD
7	142247540	Rev	CTGCTGGGCAGCCCTGT	NO	NO	Yes	Yes_NA_0.0139873248237079	UbD
7	142247540	For	ACAGGGCTGCCCAGCAG	NO	NO	Yes	Yes_NA_0.0139873248237079	UbD
7	142247539	Rev	TGCTGGGCAGCCCTGTG	NO	NO	NO	Yes_NA_0.0139827025481577	UbD
7	142247539	For	CACAGGGCTGCCCAGCA	NO	NO	NO	Yes_NA_0.0139827025481577	UbD
7	142247530	Rev	GCCCTGTGCCTCCTGGG	NO	NO	Yes	Yes_NA_0.0091016484991677	UbD
7	142247530	For	CCCAGGAGGCACAGGGC	NO	NO	Yes	Yes_NA_0.0091016484991677	UbD
7	142247347	Rev	AGGGAAAATATGTAGAG	NO	Yes_FP	Yes	NO	UbD
7	142247347	For	CTCTACATATTTTCCCT	NO	Yes_FP	Yes	NO	UbD
7	142168662	Rev	CAGTGGACGCTGGAGTC	NO	NO	Yes	NO	UbD
7	142168662	For	GACTCCAGCGTCCACTG	NO	NO	Yes	NO	UbD
7	142139271	For	CTATGCCATGCTGTGGC	NO	NO	Yes	NO	UbD
7	142131957	Rev	AGGAAAATCAAGGCCCA	NO	NO	Yes	NO	UbD
7	142131747	Rev	AGGACCTCCAGGCTGTC	NO	NO	Yes	Yes_rs368257836_0.214513788098694	UbD
7	142131747	For	GACAGCCTGGAGGTCCT	NO	NO	Yes	Yes_rs368257836_0.214513788098694	UbD
7	142131746	Rev	GGACCTCCAGGCTGTCC	NO	NO	Yes	Yes_NA_0.20873820754717	UbD
7	142131746	For	GGACAGCCTGGAGGTCC	NO	NO	Yes	Yes_NA_0.20873820754717	UbD
7	142099571	Rev	GACAAATCGGGGCTGCC	NO	NO	NO	Yes_NA_0.118193939393939	UbD
7	142099571	For	GGCAGCCCCGATTTGTC	NO	NO	NO	Yes_NA_0.118193939393939	UbD
17	10258155	Rev	AAAAAAAAA	NO	NO	Yes	Yes_rs5010940_0.00439297124600639	Ub
17	10258155	Rev	AACAACCCA	NO	NO	Yes	Yes_rs5010940_0.00439297124600639	bD
17	10258155	For	TGGGTTGTT	NO	NO	Yes	Yes_rs5010940_0.00439297124600639	Ub
17	10258153	For	TTTTTTTTT	NO	NO	Yes	Yes_rs398100417_0.00439297124600639	bD
17	10258153	Rev	AAAAAAAAA	NO	NO	Yes	Yes_rs398100417_0.00439297124600639	Ub
17	10258153	Rev	ACAACCCAG	NO	NO	Yes	Yes_rs398100417_0.00439297124600639	bD
17	10258153	For	CTGGGTTGT	NO	NO	Yes	Yes_rs398100417_0.00439297124600639	Ub
17	8387438	For	ACAAATGGATAAGTAAC	Yes	Yes_TP	Yes	NO	UbD
17	7124751	For	TCAGGAACT	NO	NO	NO	NO	bD
17	7124750	For	TCAGGAACT	Yes	NO	Yes	NO	bD
17	5424129	For	CCCTGAGACCTGCCCCA	NO	NO	NO	Yes_rs12946467_0.20926517571885	UbD
17	5421005	For	TGCCCAGCTCCTTCATT	NO	NO	NO	NO	UbD
17	5036383	Rev	TGCCACCCGCCCTCTGA	NO	NO	Yes	NO	UbD
17	5036210	Rev	TCCCGCCGAATTTTCTG	NO	NO	Yes	NO	UbD
17	5036210	For	CAGAAAATTCGGCGGGA	NO	NO	Yes	NO	UbD
17	5036152	For	TCCATGGGCGGCTCTGG	NO	NO	Yes	NO	UbD
17	5036148	For	AGCATCCATGGGCGGCT	NO	NO	Yes	NO	UbD
17	5036138	For	CCCAAGACTCAGCATCC	NO	NO	NO	NO	UbD
17	5036122	For	CACCAGGGCTCCAGAGC	NO	NO	NO	NO	UbD
17	5036110	For	CCAGTTGGGTCCCACCA	NO	NO	NO	NO	UbD
17	5036106	For	CCTTCCAGTTGGGTCCC	NO	NO	NO	NO	UbD
17	4686340	Rev	CCGCAAGAAATTTCACC	NO	NO	Yes	NO	UbD
17	4686340	For	GGTGAAATTTCTTGCGG	NO	NO	Yes	NO	UbD
17	4458314	Rev	GGATGCCAGGGTGCGCG	Yes	NO	Yes	NO	UbD
17	4350373	Rev	TGACCACCACCCCCCGT	Yes	NO	Yes	Yes_rs373747529_0.0419329073482428	UbD
17	2597723	Rev	TCCCCGCTGCGTCCCTG	NO	NO	Yes	NO	UbD
17	2597723	For	CAGGGACGCAGCGGGGA	NO	NO	Yes	NO	UbD
18	77096644	Rev	GTCCTTCGCGAGCCATG	NO	Yes_FP	NO	Yes_rs201591038_0.0408331955695156	UbD
18	77096644	For	CATGGCTCG	NO	Yes_FP	NO	Yes_rs201591038_0.0408331955695156	Ub
18	61231143	For	AAAAAAAAAAAATTATC	Yes	NO	Yes	NO	UbD
18	61226740	For	CAAAACACGCATCTTTT	Yes	NO	Yes	NO	UbD
18	54354076	For	AAATCTTTCTTTTCAGG	Yes	Yes_TP	Yes	NO	UbD
18	33848663	Rev	GAAACTTTATGCTAAAA	Yes	Yes_TP	Yes	NO	UbD
18	32886528	For	AACATTTTAGAGTTCTG	Yes	NO	Yes	NO	UbD
18	32650341	Rev	AAATCACATTCCACATG	Yes	NO	Yes	Yes_rs3744923_0.297523961661342	UbD
18	28666526	Rev	ACATAAACAATAAAACC	Yes	Yes_TP	Yes	NO	UbD
18	28666526	For	GGTTTTATTGTTTATGT	Yes	Yes_TP	Yes	NO	UbD
18	9859134	For	ACAGCCCAAAGCAGGAG	Yes	NO	Yes	NO	UbD
18	3277466	Rev	TATTTCACAGAAGTACT	NO	NO	Yes	NO	UbD
19	58600105	Rev	GCTCGCGAGCCCCAGCC	NO	NO	NO	Yes_NA_0.130726763348715	UbD
19	55903158	Rev	TGTGATCATGAGTCCAT	Yes	Yes_TP	Yes	NO	UbD
19	54724794	Rev	GGACAGAGA	NO	NO	Yes	NO	bD
19	54257190	For	GGGCCAATACATGCATC	Yes	NO	Yes	NO	UbD
19	53651845	For	TTCAAACTTGTTTTCCC	Yes	NO	Yes	NO	UbD
19	52920006	Rev	TTGCCATACAGTTTGTA	Yes	Yes_TP	Yes	NO	UbD
19	50764079	Rev	CACACACACTTGCACAC	NO	NO	Yes	NO	UbD
19	50764017	Rev	TCATGCACGCACACACA	NO	NO	Yes	Yes_rs551937796_0.000287092328892972	UbD
19	50496281	Rev	GGGCTCCCTCCTGGTCT	Yes	NO	Yes	NO	UbD
19	49894027	For	ACACCATTTCCTGCTTC	Yes	NO	Yes	NO	UbD
19	49207255	Rev	TCAAAGAATGGGCCAGC	Yes	Yes_TP	Yes	NO	UbD
19	46327209	Rev	GCCCCGCAAGCCACCCT	Yes	NO	Yes	NO	UbD
19	45157113	For	CCCGCGTACCCCAGGCC	Yes	NO	Yes	NO	UbD
19	45024879	Rev	CACACACACGCACACAC	NO	NO	Yes	NO	UbD
19	41387656	Rev	TCCATCAGTTCAGTGAG	NO	NO	Yes	Yes_rs10425176_0.299210232637995	UbD
19	41387656	For	CTCACTGAA	NO	NO	Yes	Yes_rs10425176_0.299210232637995	Ub
19	41382653	Rev	GGGCACACTAGTTCCCC	NO	NO	Yes	NO	UbD
19	41382612	Rev	AATCCCCCGACCCTCCT	NO	NO	Yes	Yes_rs2316208_3.48705431087089e-05	UbD
19	41382607	Rev	CCCGACCCTCCTCATCA	NO	NO	Yes	Yes_rs139072168_0.000854383358098068	UbD
19	41381868	Rev	CCTCCTCCCTGGGAGAG	NO	NO	Yes	NO	UbD
19	40901647	Rev	GGGGCAGGT	NO	NO	NO	Yes_NA_0.000133500208594076	Ub
19	40901647	For	CGGCTGGGA	NO	NO	NO	Yes_NA_0.000133500208594076	Ub
19	40368498	Rev	GACCCCTTTCATCTGGC	NO	NO	Yes	NO	UbD
19	38135444	For	AAAAAAACAAAGACAGC	Yes	NO	NO	NO	UbD
19	35506729	Rev	GCCAAGTCC	NO	NO	NO	Yes_rs2290647_0.289850249584027	Ub
19	35506729	For	TCTCCAGCGGACTTGGC	NO	NO	NO	Yes_rs2290647_0.289850249584027	UbD
19	34986776	Rev	TGGCAAAACTAAAATTT	Yes	NO	Yes	Yes_rs35570240_0.262779552715655	UbD
19	34986767	Rev	TAAAATTTTGATAAGGT	Yes	NO	Yes	Yes_rs35126035_0.263977635782748	UbD
19	34922865	Rev	CCCTTATACAACTTTGC	Yes	Yes_TP	Yes	NO	UbD
19	33700445	Rev	GGTCGGGGGTACAGACA	Yes	NO	Yes	Yes_rs2303093_0.284944089456869	UbD
19	33695830	Rev	TGCCGCCCCTGTGGGCC	Yes	NO	Yes	Yes_rs11666900_0.218051118210863	UbD
19	33490442	For	GAGCAGGACCAGCTGGG	NO	NO	Yes	Yes_NA_0.175475280873477	UbD
19	33490429	For	TGTGGCTGGCTGTGAGC	NO	NO	Yes	NO	UbD
19	33444456	For	AGCAATGCACTAATTTA	NO	NO	Yes	Yes_rs2944021_0.183905750798722	UbD
19	33444453	For	CTTAGCAATGCACTAAT	NO	NO	Yes	NO	UbD
19	21299774	Rev	TTAGTTCATAACCTTTT	NO	Yes_TP	Yes	NO	UbD
19	20844340	Rev	CTGGTTCTTCTCCTAAA	Yes	NO	Yes	NO	UbD
19	19750822	For	CTGGGAAACGAGGGGAT	NO	NO	NO	Yes_rs45458192_0.239416932907348	UbD
19	19035354	For	CTGTCCCGGCCTCCACC	NO	NO	NO	Yes_rs2301659_0.297523961661342	UbD
19	18897526	Rev	CAGGCTGGCTCATCCTA	Yes	NO	Yes	NO	UbD
19	17887563	Rev	ACACCAGGCGGAGACAT	Yes	NO	Yes	Yes_rs13345379_0.290535143769968	UbD
19	17516400	Rev	GCCAGACTCTAAAGGGG	Yes	Yes_TP	Yes	NO	UbD
19	17366110	For	AAAAAAACA	NO	NO	Yes	NO	Ub
19	16593189	For	CCTGTCTCAACAAAACA	Yes	NO	Yes	NO	UbD
19	16339886	Rev	CTCCGTCCA	NO	NO	Yes	NO	bD
19	16186815	For	ACGTGACCC	Yes	Yes_TP	Yes	NO	Ub
19	12577723	Rev	AACTGAGTTTATATCCC	NO	NO	Yes	NO	UbD
19	11230650	For	AAACAAACA	NO	NO	Yes	NO	bD
19	10091280	For	TCTCTCTCACACACACA	NO	NO	NO	Yes_rs58569966_0.00303297672879674	UbD
19	10091267	For	ACTCAGAATCTCTCTCT	NO	NO	Yes	NO	UbD
19	9006418	Rev	CTCCCTCCCCAACAGCT	NO	NO	NO	Yes_9006418_rs201532862_0.1522801963	UbD
19	9006418	For	AGCTGTTGGGGAGGGAG	NO	NO	NO	Yes_9006418_rs201532862_0.1522801963	UbD
19	9006416	Rev	CCCTCCCCAACAGCTCA	NO	NO	NO	Yes_rs77248019_0.224375620157494	UbD
19	9006416	For	TGAGCTGTTGGGGAGGG	NO	NO	NO	Yes_rs77248019_0.224375620157494	UbD
19	9002636	Rev	TGGATGCTGTCTGCACC	NO	NO	NO	NO	UbD
19	9002636	For	GGTGCAGACAGCATCCA	NO	NO	NO	NO	UbD
19	9002612	Rev	CTGACCCCAAAAGCCCT	NO	NO	Yes	NO	UbD

7	142099385	For	TTTAGGGTCTCTAGGAG	NO	NO	NO	NO	UbD
7	142099379	For	CCTCTGTTTAGGGTCTC	NO	NO	NO	NO	UbD
7	142099369	For	CAAAGAGAGGCCTCTGT	NO	NO	NO	NO	UbD
7	142099358	For	AAAGTGAGGAGCAAAGA	NO	NO	NO	NO	UbD
7	142099354	For	TGAGGAGCA	NO	NO	NO	NO	bD
7	142099351	For	AAGTGAGGA	NO	NO	NO	NO	bD
7	142099350	For	AAAGTGAGG	NO	NO	NO	NO	bD
7	142012957	For	CCAGGGCCA	Yes	NO	Yes	Yes_rs2003790_0.206669329073482	bD
7	141952110	Rev	TTGAGAGCCGATGTTGG	NO	NO	NO	Yes_rs1052406_0.286703170970906	UbD
7	141760293	Rev	GACACATCACCTAAGTG	NO	NO	Yes	NO	UbD
7	141740077	Rev	ACTAAAAACTACTCCTC	Yes	NO	Yes	Yes_rs2960753_0.279952076677316	UbD
7	141635538	For	CACTCCCCCAAGTTTCT	NO	NO	Yes	NO	UbD
7	141635534	For	CACTCCCCCAAGTTTCT	NO	NO	NO	NO	UbD
7	138356732	For	CTCAGTTGCTGTGTTCA	NO	NO	NO	NO	UbD
7	138356664	For	AGTGGTGAG	Yes	NO	Yes	NO	bD
7	134221826	Rev	TTCTCCTGTGTGAGGTA	NO	NO	Yes	Yes_rs28545160_0.105862908837857	UbD
7	134221826	For	TACCTCACACAGGAGAA	NO	NO	Yes	Yes_rs28545160_0.105862908837857	UbD
7	128119648	Rev	TTACTTTTATTGTCAGC	NO	NO	Yes	NO	UbD
7	127670327	Rev	ACGGCCTGGCCAGCCTC	NO	NO	NO	Yes_NA_0.0407994553588796	UbD
7	127670327	For	GAGGCTGGCCAGGCCGT	NO	NO	NO	Yes_NA_0.0407994553588796	UbD
7	124499002	For	TTCCTACTATACATCAC	NO	Yes_TP	Yes	NO	UbD
7	120450678	Rev	GTGTGTGTGAATCTGTG	NO	NO	Yes	NO	UbD
7	111423074	Rev	TTTTTTTTC	NO	NO	NO	NO	Ub
7	107577456	For	TTTCCCCAA	NO	NO	Yes	Yes_rs56285692_0.265974440894569	bD
7	103838120	For	CTTTTACAACACAATCC	Yes	NO	Yes	Yes_rs13238183_0.278154952076677	UbD
7	102412901	Rev	CTTAGCACATACCTACT	NO	NO	NO	Yes_rs2539288_0.285879725370171	UbD
7	102412901	For	AGTAGGTATGTGCTAAG	NO	NO	NO	Yes_rs2539288_0.285879725370171	UbD
7	100613112	Rev	TGAGGTCACAGCACGGG	NO	NO	Yes	NO	UbD
7	100613107	Rev	TCACAGCACGGGTCTCC	NO	NO	Yes	NO	UbD
7	100612869	For	CTGACGAGGGAAGACGG	NO	NO	Yes	Yes_rs74554402_0.296468979236603	UbD
7	100612842	For	TGGGAAGCTTTCATTTC	NO	NO	Yes	NO	UbD
7	100612839	For	TCCTGGGAAGCTTTCAT	NO	NO	Yes	NO	UbD
7	100416332	Rev	AAAAAAAAATTTTTTTT	Yes	NO	Yes	Yes_rs314364_0.234424920127796	UbD
7	100374265	Rev	AGGGGAGGGAAGGGAAG	NO	NO	Yes	NO	UbD
7	99507346	Rev	AACAAAAAGAAAAAAGA	NO	NO	NO	NO	UbD
7	82581488	Rev	ATAGAAGATGAAAAACC	Yes	Yes_TP	Yes	NO	UbD
7	82581488	For	GGTTTTTCATCTTCTAT	Yes	Yes_TP	Yes	NO	UbD
7	75614863	For	GGCCTCGGTGTGGCGGT	Yes	Yes_TP	Yes	NO	UbD
7	74524986	Rev	AGCCTGGGCGACAGAGT	NO	NO	Yes	NO	UbD
7	73771749	Rev	TGTGGCTTG	NO	Yes_FP	NO	Yes_rs75980392_0.147382654046745	Ub
7	72738762	Rev	AGCCTGCTGGAGCTGGA	NO	NO	Yes	NO	UbD
7	65552230	For	GTGCGTTCGTGTGTGTG	NO	NO	Yes	NO	UbD
7	45697295	Rev	GGGGCTAGG	Yes	Yes_TP	Yes	NO	Ub
7	45697295	For	AGGTGGGCCCTAGCCCC	Yes	Yes_TP	Yes	NO	UbD
7	45123937	For	GGGGCTGATGAGAGATC	NO	NO	NO	Yes_rs112035890_0.00255399883245768	UbD
7	36447349	For	AAGCTACTTCTCCAGTG	NO	Yes_TP	Yes	NO	UbD
7	30811183	Rev	GGAGGGGGCGAGCCGTT	NO	NO	NO	Yes_rs371170994_0.111033052321691	UbD
7	21941861	For	TAAAAAAATCTTTCTTA	NO	NO	Yes	NO	UbD
7	18067479	Rev	CCATTAGAGTCTGTGCT	NO	Yes_TP	Yes	NO	UbD
7	14517650	For	TAATATTTACACACACA	NO	NO	Yes	Yes_rs397806961_0.298921725239617	UbD
7	7491873	For	AGTCTGTAAGGCAATTG	NO	NO	Yes	NO	UbD
7	4172109	Rev	TCTCCCAAGGTGCTGGG	Yes	NO	Yes	Yes_rs13224986_0.244808306709265	UbD
7	2296722	Rev	CGCTGGTGTTCGGGCCT	Yes	NO	Yes	Yes_rs3815313_0.218051118210863	UbD
7	1528947	For	GCTCCCCATCCCGGGCC	NO	Yes	Yes	NO_1528947	UbD
7	1528946	For	TGCTCCCCATCCCGGGC	NO	NO	NO	NO	UbD
7	1528945	For	CTGCTCCCCATCCCGGG	NO	NO	NO	NO	UbD
7	1132016	Rev	CGTGACCTCGAGCCACT	NO	Yes_FP	NO	Yes_rs199736734_0.00322018487649644	UbD
7	1132016	For	AGTGGCTCG	NO	Yes_FP	NO	Yes_rs199736734_0.00322018487649644	Ub
7	940258	Rev	ACCTTTGCCCCTAGGAG	NO	NO	Yes	Yes_NA_0.000278736679602907	UbD
8	143922788	Rev	ACCAAAGCGTGCAGAAA	NO	NO	NO	Yes_rs3802226_0.28594249201278	UbD
8	133041177	For	GAAGGAAGG	NO	NO	NO	NO	bD
8	129021000	For	ACAATGCTAAGCATGTG	Yes	NO	NO	NO	UbD
8	124664873	Rev	GAGGCACATATTGCCAC	Yes	Yes_TP	Yes	NO	UbD
8	124664873	For	GTGGCAATATGTGCCTC	Yes	Yes_TP	Yes	NO	UbD
8	117778911	For	GGAGTCCGCGAGCCGTA	NO	NO	NO	Yes_117778911_NA_0.00736870310825295	UbD
8	117778909	For	GGAGTCCGCGAGCCGTA	NO	NO	NO	NO	UbD
8	101721899	Rev	CCCCAGAAGAAGCCACT	NO	NO	NO	Yes_rs142985461_0.245508452944295	UbD
8	101721899	For	AGTGGCTTCTTCTGGGG	NO	NO	NO	Yes_rs142985461_0.245508452944295	UbD
8	101721870	Rev	GAAATGAACGGTAGAAT	NO	NO	NO	Yes_rs146609644_0.217824438072334	UbD
8	101721870	For	ATTCTACCGTTCATTTC	NO	NO	NO	Yes_rs146609644_0.217824438072334	UbD
8	101721727	Rev	CTACCAGCCAGCACCTC	NO	Yes_FP	NO	Yes_rs139094790_0.105632898622499	UbD
8	101721727	For	GAGGTGCTGGCTGGTAG	NO	Yes_FP	NO	Yes_rs139094790_0.105632898622499	UbD
8	101721451	Rev	CCCTGCAGACTCAGAAC	NO	NO	NO	Yes_NA_0.000122838050314465	UbD
8	101721451	For	GTTCTGAGTCTGCAGGG	NO	NO	NO	Yes_NA_0.000122838050314465	UbD
8	101721442	Rev	CTCAGAACCGTGCTGCA	NO	NO	NO	Yes_NA_0.000344409092400039	UbD
8	101721442	For	TGCAGCACGGTTCTGAG	NO	NO	NO	Yes_NA_0.000344409092400039	UbD
8	101718932	For	ATTGCGAACTCCTGCAG	NO	NO	NO	Yes_rs62513920_0.255470109757397	UbD
8	97257982	For	TAACTTATTTTAAAGTC	NO	NO	Yes	Yes_rs871798_0.236421725239617	UbD
8	95718131	For	TTTTTTTTT	NO	NO	NO	Yes_NA_0.0255563707507948	bD
8	92082753	Rev	GCCACGCGC	Yes	NO	Yes	NO	bD
8	86126827	Rev	AAACCAGTTGATGAAGC	Yes	Yes_TP	Yes	NO	UbD
8	86126827	For	GCTTCATCAACTGGTTT	Yes	Yes_TP	Yes	NO	UbD
8	86114525	Rev	GAAGAAAAAAGGTTTCA	Yes	Yes_TP	Yes	NO	UbD
8	52733076	Rev	GACAGTGAAGAGGATGA	NO	NO	NO	Yes_rs77409139_0.28971028971029	UbD
8	52733076	For	TCATCCTCTTCACTGTC	NO	NO	NO	Yes_rs77409139_0.28971028971029	UbD
8	28186607	For	AGCAGCCCTCCGAGAAT	NO	Yes_TP	Yes	Yes_rs2645721_0.223841853035144	UbD
8	27101204	Rev	ATGTGTTAAGGGGCGTA	Yes	Yes_TP	Yes	NO	UbD
8	27101204	For	TACGCCCCTTAACACAT	Yes	Yes_TP	Yes	NO	UbD
8	24339896	Rev	GTTTTTTTTCTGTTAAT	NO	NO	Yes	NO	UbD
8	22974450	Rev	GGCTGGTGATCATTGTC	Yes	Yes_TP	NO	Yes_rs9644063_0.058804096465147	UbD
8	22974450	For	GACAATGATCACCAGCC	Yes	Yes_TP	NO	Yes_rs9644063_0.058804096465147	UbD
8	21931182	For	AAAAAAGAGAAAAAAGA	NO	NO	NO	NO	UbD
8	20022694	Rev	TAATTGTACTTTCTCAG	Yes	NO	Yes	Yes_rs3779670_0.203274760383387	UbD
8	18729817	Rev	CAACAAAACGCTCCCTG	Yes	Yes_TP	Yes	NO	UbD
8	18729817	For	CAGGGAGCGTTTTGTTG	Yes	Yes_TP	Yes	NO	UbD
8	17796382	Rev	TTGGGGGGTTTGTAGAT	Yes	Yes_TP	Yes	NO	UbD
19	9002612	For	AGGGCTTTTGGGGTCAG	NO	NO	Yes	NO	UbD
19	9002454	Rev	GCTGGACAG	NO	NO	Yes	Yes_rs372480224_0.0341042528207485	Ub
19	9002454	For	AATAGAGGCTGTCCAGC	NO	NO	Yes	Yes_rs372480224_0.0341042528207485	UbD
19	9001833	Rev	TGGACCCATGAGTGAGT	NO	Yes_TP	Yes	NO	UbD
19	9001833	For	ACTCACTCATGGGTCCA	NO	Yes_TP	Yes	NO	UbD
19	8999560	Rev	CCAGGCCCAAGAAGGAT	NO	NO	NO	Yes_rs77501519_8.01853171774768e-05	UbD
19	8999560	For	ATCCTTCTTGGGCCTGG	NO	NO	NO	Yes_rs77501519_8.01853171774768e-05	UbD
19	8999538	Rev	AGCCACCAAAGTGGATG	NO	NO	Yes	Yes_rs78334969_0.000166054885509526	UbD
19	8999538	For	CATCCACTTTGGTGGCT	NO	NO	Yes	Yes_rs78334969_0.000166054885509526	UbD
19	8999474	Rev	GAGCAGCTATACTGGGA	NO	NO	NO	Yes_rs199946350_0.0366018739821062	UbD
19	8999474	For	TCCCAGTATAGCTGCTC	NO	NO	NO	Yes_rs199946350_0.0366018739821062	UbD
19	8999386	Rev	ATGGTGAGTAGTTGTGA	NO	NO	Yes	Yes_rs77049866_0.0632257578042528	UbD
19	8999386	For	TCACAACTACTCACCAT	NO	NO	Yes	Yes_rs77049866_0.0632257578042528	UbD
19	8999383	Rev	GTGAGTAGTTGTGATGT	NO	NO	Yes	Yes_rs79909361_0.060441952865618	UbD
19	8999383	For	ACATCACAACTACTCAC	NO	NO	Yes	Yes_rs79909361_0.060441952865618	UbD
19	8999311	For	ATAGCCAGGCAGGGAGC	NO	NO	Yes	Yes_rs11673197_0.240814696485623	UbD
19	8999305	For	CATTCAATAGCCAGGCA	NO	NO	NO	NO	UbD
19	8321328	For	CCAAGCAGG	NO	NO	Yes	NO	bD
19	8154990	Rev	CCCCAGGAG	NO	NO	NO	Yes_rs35002391_0.295864132923399	Ub
19	8154990	For	CCGCTGCCCTCCTGGGG	NO	NO	NO	Yes_rs35002391_0.295864132923399	UbD
19	7619647	Rev	GGACGGGACCCGAGGTG	Yes	Yes_TP	Yes	Yes_rs599328_0.020751378558054	UbD
19	6836844	For	AGAGATGGACACACACA	NO	NO	Yes	NO	UbD
19	6471567	Rev	GCAGGCTTCGAAGTGCT	Yes	NO	Yes	Yes_rs8105247_0.220447284345048	UbD
19	6471170	For	GCCCTAACCGGTCTCAA	Yes	NO	Yes	Yes_rs11882213_0.254992012779553	UbD
19	5787215	Rev	CCTCCCACC	NO	NO	Yes	NO	Ub
19	5787215	For	TGGGAGATGGTGGGAGG	NO	NO	Yes	NO	UbD
19	5787214	Rev	CTCCCACCA	NO	NO	NO	Yes_rs28420751_0.040650406504065	Ub
19	5787214	For	GTGGGAGATGGTGGGAG	NO	NO	NO	Yes_rs28420751_0.040650406504065	UbD
19	4960583	For	GTGAGACTGTCCCCACA	NO	NO	NO	NO	UbD
19	4513143	Rev	GGGTGACTGGTGCCATG	Yes	NO	Yes	NO	UbD
19	4513143	For	CATGGCACCAGTCACCC	Yes	NO	Yes	NO	UbD
19	4511581	Rev	ACCAAGGATGCTGTGTG	NO	NO	Yes	Yes_4511581_rs75029810_0.21791410766	UbD
19	4511581	For	CACACAGCATCCTTGGT	NO	NO	Yes	Yes_4511581_rs75029810_0.21791410766	UbD
19	4511578	Rev	AAGGATGCTGTGTGCAG	NO	NO	Yes	Yes_rs62115186_0.225828262339419	UbD
19	4511578	For	CTGCACACAGCATCCTT	NO	NO	Yes	Yes_rs62115186_0.225828262339419	UbD
19	4511575	Rev	GATGCTGTGTGCAGTGG	NO	NO	Yes	Yes_rs62115185_0.25306404506102	UbD
19	4511575	For	CCACTGCACACAGCATC	NO	NO	Yes	Yes_rs62115185_0.25306404506102	UbD
19	4511491	Rev	CTGACCGGTACCAAGGA	NO	NO	NO	Yes_rs10423324_0.00416571559004793	UbD
19	4511491	For	TCCTTGGTACCGGTCAG	NO	NO	NO	Yes_rs10423324_0.00416571559004793	UbD
19	4453324	For	ACAACCTGTGTCCACCT	Yes	NO	Yes	NO	UbD
19	2341059	Rev	CTGGTGCACCCGAGGAC	Yes	Yes_TP	Yes	NO	UbD
19	968345	For	CAACAGAGCAAGACTCT	Yes	NO	Yes	NO	UbD
20	48562083	Rev	GGCATGCAACCTCTTTG	NO	NO	Yes	NO	UbD
20	45808688	Rev	AAAAAAAAA	Yes	Yes	Yes	Yes_rs6066225_0.273961661341853	Ub
20	43743534	For	ACACACGCGCACACACA	NO	NO	Yes	Yes_rs11700135_0.257787539936102	UbD
20	43743522	For	ACACACGCGCACACACA	NO	NO	NO	NO	UbD
20	33637606	For	TTATTCAGTTTTGAGAA	Yes	Yes_TP	Yes	NO	UbD
20	29589675	Rev	AAAAAAAAA	NO	NO	Yes	Yes_rs74625909_0.134384984025559	Ub
20	29589675	For	TTTTTTTTT	NO	NO	Yes	Yes_rs74625909_0.134384984025559	bD
20	26094689	Rev	TTTAGATGTTTTCCTAG	NO	NO	Yes	NO	UbD
20	26062036	Rev	GGAGGATGTAGCGGATA	NO	NO	Yes	Yes_rs77424811_0.24294471431876	UbD
20	26062021	Rev	TACCTTCCGAAGTTTGT	NO	NO	Yes	Yes_rs79451834_0.261458568525627	UbD
20	26061960	Rev	GCCGACTGAGGCGCAGA	NO	NO	Yes	Yes_rs144572655_0.299793127226756	UbD
20	26061922	Rev	AGGCAGTGACTTTGCTG	NO	NO	Yes	Yes_rs75066699_0.136006016429481	UbD
20	26061922	For	CAGCAAAGT	NO	NO	Yes	Yes_rs75066699_0.136006016429481	Ub
20	26061909	Rev	GCTGGGCACCCGCAGTG	NO	NO	NO	Yes_rs116264914_0.110130425005622	UbD
20	26061909	For	CACTGCGGGTGCCCAGC	NO	NO	NO	Yes_rs116264914_0.110130425005622	UbD
20	26061884	Rev	CCCAAGTGGAGATCCCC	NO	NO	NO	Yes_rs78196071_0.0967535923363491	UbD
20	26061884	For	GGGGATCTCCACTTGGG	NO	NO	NO	Yes_rs78196071_0.0967535923363491	UbD
20	26061880	Rev	AGTGGAGATCCCCACTT	NO	NO	Yes	Yes_rs80023315_0.103899721448468	UbD
20	26061880	For	AAGTGGGGATCTCCACT	NO	NO	Yes	Yes_rs80023315_0.103899721448468	UbD
20	26061877	Rev	GGAGATCCCCACTTCGT	NO	NO	Yes	Yes_rs74473084_0.102631006740596	UbD
20	26061877	For	ACGAAGTGGGGATCTCC	NO	NO	Yes	Yes_rs74473084_0.102631006740596	UbD
20	26061859	For	ACGTGGAGGAGGAGGAA	NO	NO	Yes	Yes_rs74752031_0.164775826333588	UbD
20	18429539	For	TGGGCAAAA	NO	NO	NO	NO	bD
20	16729654	Rev	AAATTATCTTACACTGC	Yes	Yes_TP	Yes	NO	UbD
20	4880132	For	GCCTTCTGCGCTCAATG	Yes	Yes_TP	Yes	NO	UbD
20	4768475	Rev	AAGGTGCCTTTGGACCC	Yes	Yes_TP	Yes	NO	UbD
20	3657803	Rev	TCCACACATTTTCTTGC	Yes	Yes_TP	Yes	NO	UbD
21	47419780	Rev	CTCACGGGT	NO	NO	Yes	NO	bD
21	46930992	For	CCCCACACACCACACAC	NO	Yes_TP	Yes	NO	UbD
21	46086757	Rev	CTGCTGCGCGCCCAGCC	NO	NO	Yes	NO	UbD
21	46086718	Rev	GCCCAGCTCCTGCCAGG	NO	Yes_TP	Yes	NO	UbD
21	46086718	For	CCTGGCAGGAGCTGGGC	NO	Yes_TP	Yes	NO	UbD
21	46020473	For	TCACTCACTCACTCACT	NO	NO	Yes	NO	UbD
21	43897241	For	CACACCCTC	NO	NO	NO	NO	bD
21	42852320	For	TGCCGGTGCTTTCACAG	NO	NO	Yes	Yes_rs734056_0.284544728434505	UbD
21	42843962	Rev	GTTCATCCACTGAGAGC	NO	NO	Yes	Yes_rs458213_0.225239616613419	UbD
21	42842713	Rev	GAGACATAGGTGATACC	NO	Yes_TP	Yes	NO	UbD
21	42824874	Rev	GAGCAAGAC	NO	NO	Yes	Yes_rs467798_0.275359424920128	bD
21	42809206	Rev	TGGCCTGAT	NO	NO	Yes	Yes_rs468646_0.232228434504792	bD
21	41506058	Rev	TGTCCAGAT	Yes	NO	Yes	Yes_rs403892_0.291333865814697	bD
21	40646230	For	CAGAGAGCGAGAGAGAG	NO	NO	Yes	NO	UbD
21	38469017	Rev	TATACCTGAAAAACAGA	Yes	NO	Yes	NO	UbD
21	37728892	For	TTGATTTTA	Yes	Yes_TP	Yes	NO	bD
21	34915324	For	AATTGGCCCGTGCGCCT	Yes	Yes_TP	Yes	NO	UbD
21	30414724	For	CCACAGGGCAGGGAGGA	NO	NO	NO	Yes_rs2254872_0.29452875399361	UbD
21	26973663	For	AAAAGACATATATGTAT	Yes	Yes_TP	Yes	NO	UbD
21	11058404	Rev	TGTACAGAGTTTTTTAT	NO	NO	Yes	NO	UbD
21	11058227	Rev	GAGCCGACACCTTTCAG	NO	NO	Yes	NO	UbD
21	11058227	For	CTGAAAGGTGTCGGCTC	NO	NO	Yes	NO	UbD
21	11058043	For	GATTTTAAAGATTATGT	NO	NO	Yes	NO	UbD
21	11058025	For	TCTGGCAATGATCCCTC	NO	NO	Yes	NO	UbD
21	11058008	For	TTCCCTCCGCAACAATC	NO	NO	Yes	NO	UbD
21	11049763	Rev	ACCTCTTTCTGATTAGA	NO	NO	Yes	NO	UbD
21	11049503	Rev	GTTACTCCATTAAAACG	NO	NO	Yes	NO	UbD

8	17796382	For	ATCTACAAACCCCCCAA	Yes	Yes_TP	Yes	NO	UbD
8	17732084	Rev	CCCTAAGGCGGGGGGAG	Yes	NO	Yes	Yes_rs2517294_0.000399361022364217	UbD
8	13356818	Rev	TAGTACAAAATGAGTTC	Yes	Yes_TP	Yes	NO	UbD
8	13356818	For	GAACTCATTTTGTACTA	Yes	Yes_TP	Yes	NO	UbD
8	11666218	For	TGCCCCAGTCCCACTCC	NO	NO	Yes	NO	UbD
8	11400944	Rev	GGAAGGCACTTTTGCTA	NO	NO	Yes	Yes_rs2245232_0.297124600638978	UbD
8	11400680	For	TGTGCAGAGGATCTGAG	NO	NO	Yes	Yes_rs2245250_0.299321086261981	UbD
8	10467637	Rev	GTTAGAGGAAACTAAAA	NO	Yes	Yes	NO	UbD
8	10467637	For	TTTTAGTTTCCTCTAAC	NO	Yes	Yes	NO	UbD
8	10467636	Rev	TTAGAGGAAACTAAAAC	NO	Yes_FP	NO	Yes_rs4840500_0.164703401743042	UbD
8	10467636	For	GTTTTAGTTTCCTCTAA	NO	Yes_FP	NO	Yes_rs4840500_0.164703401743042	UbD
8	7752127	For	GCACCCAATACCAGTTC	NO	NO	NO	NO	UbD
8	3855369	For	GTTGGGGGAAAAAACGG	Yes	NO	Yes	NO	UbD
8	2965122	For	TTGTTTCTC	Yes	NO	Yes	NO	bD
8	2071261	Rev	TTTGCTTTCTTGTGTTA	Yes	Yes_TP	Yes	NO	UbD
9	140777432	Rev	CCCAGTGAGGGGACTGC	NO	NO	Yes	NO	UbD
9	139944689	For	CCCGCCTGTCCCTACCC	NO	NO	NO	Yes_rs7869777_0.00811280086217219	UbD
9	139916471	Rev	TCTGAGCCATTCCCCCA	Yes	Yes_TP	Yes	NO	UbD
9	139639568	For	TGGGCCCCGAGTGGCCC	NO	NO	NO	NO	UbD
9	139639563	For	CAACGTGGGCCCCGAGT	NO	NO	NO	NO	UbD
9	139639560	For	TCCCAACGTGGGCCCCG	NO	NO	NO	NO	UbD
9	139639556	For	CAGCTCCCAACGTGGGC	NO	NO	NO	NO	UbD
9	138518015	Rev	ACTGGCTCGCTCCTGTC	NO	NO	Yes	Yes_rs202181415_0.286497569516437	UbD
9	138518015	For	GACAGGAGCGAGCCAGT	NO	NO	Yes	Yes_rs202181415_0.286497569516437	UbD
9	137982220	For	TGTGTGTGT	NO	NO	Yes	Yes_NA_3.57513138607844e-05	bD
9	137982220	Rev	ACACACACATACAAACT	NO	NO	Yes	Yes_NA_3.57513138607844e-05	UbD
9	137715150	For	TGTGGCTGATAGGGCCA	Yes	NO	Yes	NO	UbD
9	136135237	Rev	CTGCAGCATGTCTCGTT	Yes	Yes_TP	Yes	NO	UbD
9	136135237	For	AACGAGACATGCTGCAG	Yes	Yes_TP	Yes	NO	UbD
9	136135137	For	CCCACATGCTTCTGTCC	Yes	NO	Yes	NO	UbD
9	136133380	For	GAGAGAACGGGGAAGCA	Yes	NO	Yes	NO	UbD
9	134497450	Rev	GGAAGAATGGGGGGAGG	Yes	NO	Yes	Yes_rs7023385_0.275159744408946	UbD
9	132891113	Rev	CAGCCGACCCGATTCAC	Yes	NO	Yes	NO	UbD
9	132862830	For	ATTTCCCAAATGGGTCC	Yes	NO	Yes	NO	UbD
9	132636031	Rev	GAGGCCACTGGCTCACC	Yes	Yes_TP	Yes	NO	UbD
9	132636031	For	GGTGAGCCAGTGGCCTC	Yes	Yes_TP	Yes	NO	UbD
9	131904617	For	TGTGCGTTTGTGTGTGT	NO	NO	Yes	NO	UbD
9	131761659	Rev	GGAAGCCCTGCCTGGCT	NO	NO	Yes	Yes_rs2302813_0.261781150159744	UbD
9	131353567	For	CATCTCCTG	NO	NO	Yes	NO	bD
9	130932396	Rev	CTTTGACTGTTGTCATT	Yes	Yes_TP	Yes	NO	UbD
9	130932396	For	AATGACAACAGTCAAAG	Yes	Yes_TP	Yes	NO	UbD
9	125772742	For	TTCAGTTCGCCTGCCAC	NO	NO	NO	Yes_NA_0.0376834944504117	UbD
9	125772740	For	TTCAGTTCGCCTGCCAC	NO	NO	NO	Yes_NA_0.0376834944504117	UbD
9	115969433	For	GAAAAAGTAGCTTAACT	Yes	NO	Yes	NO	UbD
9	115934022	Rev	TCACCTGGTTCCTGAGG	Yes	Yes_TP	Yes	NO	UbD
9	115567015	For	TCCAGTATGTTTTCATT	Yes	Yes_TP	Yes	NO	UbD
9	111637331	Rev	TTTTCTGGGGGTAGGGA	NO	NO	Yes	NO	UbD
9	97080944	For	GGCAGGAGAAGGTGATG	NO	Yes	Yes	NO	UbD
9	115969433	For	GAAAAAGTAGCTTAACT	Yes	NO	Yes	NO	UbD
9	115934022	Rev	TCACCTGGTTCCTGAGG	Yes	Yes_TP	Yes	NO	UbD
9	115567015	For	TCCAGTATGTTTTCATT	Yes	Yes_TP	Yes	NO	UbD
9	111637331	Rev	TTTTCTGGGGGTAGGGA	NO	NO	Yes	NO	UbD
9	97080944	For	GGCAGGAGAAGGTGATG	NO	Yes	Yes	NO	UbD
9	97080944	Rev	CATCACCTTCTCCTGCC	NO	Yes_TP	Yes	NO	UbD
9	96425363	Rev	CACCTCTGCTCACACTC	Yes	NO	Yes	Yes_rs3750359_0.198282747603834	UbD
9	96425363	For	GAGTGTGAGCAGAGGTG	Yes	NO	Yes	Yes_rs3750359_0.198282747603834	UbD
9	96392182	Rev	CTGCGAAGCTCTAGGGA	Yes	NO	Yes	Yes_rs10992812_0.25	UbD
9	96392182	For	TCCCTAGAGCTTCGCAG	Yes	NO	Yes	Yes_rs10992812_0.25	UbD
9	78790153	Rev	ATTCCATTTCATTCCAT	NO	NO	NO	Yes_rs4281168_0.208964084298011	UbD
9	35101383	Rev	CCTCAGGGTGCTCCAGT	Yes	Yes_TP	Yes	NO	UbD
9	35101383	For	ACTGGAGCACCCTGAGG	Yes	Yes_TP	Yes	NO	UbD
9	33798073	Rev	CCACCAAAGCTCAGAGT	NO	Yes_FP	Yes	Yes_NA_0.296949062610541	UbD
9	33797783	For	GGGTGCCCAGGCTGTGG	NO	NO	Yes	Yes_rs371972822_0.118172335716285	UbD
9	33797764	For	ACCCTGGGGAAGGTGGG	NO	NO	Yes	NO	UbD
9	33797760	For	GAGGACCCTGGGGAAGG	NO	NO	Yes	NO	UbD
9	33797752	For	GTGAGCTTGAGGACCCT	NO	NO	Yes	NO	UbD
9	33797745	For	ATGAGCAGTGAGCTTGA	NO	NO	Yes	NO	UbD
9	33797733	For	ATCCATGAG	NO	NO	Yes	NO	bD
9	33386510	Rev	TGACCTTTGCTAACTGT	NO	Yes_FP	Yes	NO	UbD
9	33386510	For	ACAGTTAGCAAAGGTCA	NO	Yes_FP	Yes	NO	UbD
9	33386465	Rev	TTCCGGTCTATGTGCTG	NO	Yes_FP	Yes	Yes_rs74668961_0.0650218452839887	UbD
9	33386465	For	CAGCACATAGACCGGAA	NO	Yes_FP	Yes	Yes_rs74668961_0.0650218452839887	UbD
9	33386430	For	GTGGCAGCCGCCAGGAA	NO	UNKNO	NO	Yes_rs74589499_0.0305036984853822	UbD
9	33386400	For	GGATACTCACTGTAGAA	NO	NO	Yes	Yes_NA_0.0652044847350033	UbD
9	33386399	For	AGGATACTCACTGTAGA	NO	NO	Yes	Yes_NA_0.0645921094351588	UbD
9	33386387	For	GGACACCCGGGCAGGAT	NO	NO	Yes	Yes_rs77429065_0.0887950713715999	UbD
9	33386378	For	GCCAGAGGCGGACACCC	NO	NO	Yes	Yes_rs74462413_0.0942603367344071	UbD
9	33385852	Rev	GGCTGACCGGGATGCTC	NO	NO	Yes	Yes_rs202183465_0.266219388826325	UbD
9	33385828	Rev	GTCTCTTCGCCATCACG	NO	NO	Yes	Yes_rs73478963_0.292133772962789	UbD
9	33385784	Rev	GGAACAGAGGCGCTGGT	NO	NO	NO	NO	UbD
9	33385771	Rev	TGGTGATAGGCATCCTC	NO	Yes_FP	NO	Yes_rs117663392_0.0450752985978882	UbD
9	33385750	Rev	TCATCATCGGGGTGTCC	NO	NO	NO	Yes_rs114484742_0.0594989561586639	UbD
9	33385750	For	GGACACCCC	NO	NO	NO	Yes_rs114484742_0.0594989561586639	Ub
9	33385740	Rev	GGTGTCCCTTGGCATGA	NO	NO	NO	Yes_rs201773300_0.0515655315447916	UbD
9	33385740	For	TCATGCCAAGGGACACC	NO	NO	NO	Yes_rs201773300_0.0515655315447916	UbD
9	33385712	Rev	GCCATCAACCCGTCCCG	NO	NO	NO	Yes_rs115575789_0.0425457744725353	UbD
9	33385712	For	CGGGACGGGTTGATGGC	NO	NO	NO	Yes_rs115575789_0.0425457744725353	UbD
9	33385709	Rev	ATCAACCCGTCCCGGGA	NO	NO	NO	Yes_rs116294914_0.0382788233144528	UbD
9	33385709	For	TCCCGGGACGGGTTGAT	NO	NO	NO	Yes_rs116294914_0.0382788233144528	UbD
9	33385667	For	TTGCCCCAACCAGCAAT	NO	NO	Yes	Yes_rs79779983_0.243176233105143	UbD
9	12708910	For	AATTTCATCTGTCCACT	Yes	NO	Yes	Yes_rs2733834_0.265575079872204	UbD
9	6424124	Rev	TCTCCTGCC	Yes	NO	Yes	NO	bD
9	6424124	For	GGCAGGAGA	Yes	NO	Yes	NO	Ub
21	11049503	For	CGTTTTAATGGAGTAAC	NO	NO	Yes	NO	UbD
21	11049404	Rev	TTAAGAGGGACCCTATT	NO	NO	Yes	NO	UbD
21	11049404	For	AATAGGGTCCCTCTTAA	NO	NO	Yes	NO	UbD
21	11049395	Rev	ACCCTATTA	NO	NO	Yes	NO	Ub
21	11049395	For	GCGCCTAGTAATAGGGT	NO	NO	Yes	NO	UbD
21	10973798	Rev	GTCTGACTATATTCTTT	NO	NO	Yes	NO	UbD
21	10973798	For	AAAGAATAT	NO	NO	Yes	NO	Ub
21	10973787	Rev	TTCTTTTGACATCCTCT	NO	NO	Yes	NO	UbD
21	10973787	For	AGAGGATGTCAAAAGAA	NO	NO	Yes	NO	UbD
21	10935116	Rev	GAATCTCCACTAGAAGA	NO	NO	Yes	NO	UbD
22	51015210	For	CCTCTCCCGTCTAGCTC	Yes	NO	Yes	Yes_rs140515_0.00399361022364217	UbD
22	46765181	Rev	GAGAGAAGACATTGATC	Yes	Yes_TP	Yes	Yes_NA_0.000277882210634716	UbD
22	45724029	Rev	CCAGCCCATACCTTGGT	NO	NO	NO	NO	UbD
22	45724029	For	ACCAAGGTATGGGCTGG	NO	NO	NO	NO	UbD
22	45723980	Rev	TGTTCTGCCGCCGGCCA	NO	NO	NO	Yes_rs13053749_0.0166484716157205	UbD
22	45723980	For	TGGCCGGCGGCAGAACA	NO	NO	NO	Yes_rs13053749_0.0166484716157205	UbD
22	44177972	For	AATGTCATATGAGTCCT	Yes	NO	Yes	NO	UbD
22	43213631	For	TGAATAACAAAAAAAAA	Yes	NO	Yes	Yes_rs28362469_0.290934504792332	UbD
22	42575528	For	AAAGAAAAGAAAAAAAA	NO	NO	NO	NO	UbD
22	42575523	For	AAAGAAAAGAAAAGAAA	NO	NO	NO	NO	UbD
22	42526571	For	TGGTGGGGCATCCTCAG	NO	NO	Yes	Yes_NA_1.52452968259292e-05	UbD
22	42526561	For	TGTTTGCTGGTGGTGGG	NO	NO	Yes	NO	UbD
22	42159072	For	GTGGGCATATTGGGGCA	Yes	NO	Yes	NO	UbD
22	39636928	Rev	TGTGTGTGTGCGCATCT	NO	NO	Yes	NO	UbD
22	31018817	For	ACCTGTGGCCAGGTGGC	NO	NO	Yes	Yes_rs2301957_0.288138977635783	UbD
22	31013549	Rev	CTTCTGTGA	Yes	NO	Yes	Yes_rs740235_0.287939297124601	bD
22	31011906	Rev	TTTTGGCTGCTGTGTCC	NO	NO	Yes	Yes_rs5749135_0.288538338658147	UbD
22	31010556	Rev	GGTATGGAGGTCATCAG	Yes	NO	Yes	Yes_rs5997703_0.288538338658147	UbD
22	31010242	For	AAAAGAGCGTGTTGGAA	Yes	NO	Yes	Yes_rs5749134_0.288538338658147	UbD
22	30822920	Rev	GGAGTGTCCACAGGGAC	Yes	NO	Yes	NO	UbD
22	25750814	Rev	CATGGCCCTGGCCCATC	NO	NO	Yes	NO	UbD
22	25747933	Rev	TGTGGCTTGGCTGCACC	NO	NO	NO	NO	UbD
22	24579157	Rev	TCATGGATCCTGAGTAG	NO	NO	NO	NO	UbD
22	24579157	For	CTACTCAGGATCCATGA	NO	NO	NO	NO	UbD
22	24300683	Rev	GCAGTAGGCCGGGGCCA	NO	NO	Yes	Yes_rs140193_0.149444373483203	UbD
22	24300683	For	TGGCCCCGGCCTACTGC	NO	NO	Yes	Yes_rs140193_0.149444373483203	UbD
22	24300660	For	GGGGCAGAGAGTGGGTC	NO	NO	Yes	Yes_rs74718778_0.138717195257878	UbD
22	24300660	Rev	GACCCACTCTCTGCCCC	NO	NO	Yes	Yes_rs74718778_0.138717195257878	UbD
22	24300634	Rev	TTGGGGCCCCTCATTGG	NO	Yes_TP	Yes	Yes_rs56104230_0.207301925295005	UbD
22	24300634	For	CCAATGAGGGGCCCCAA	NO	Yes_TP	Yes	Yes_rs56104230_0.207301925295005	UbD
22	24176959	For	CTCAGGGCAAGAGGCTC	NO	Yes_TP	Yes	NO	UbD
22	23962744	Rev	TGAGAAACAAACTTGCT	NO	NO	Yes	NO	UbD
22	23962744	For	AGCAAGTTTGTTTCTCA	NO	NO	Yes	NO	UbD
22	22899363	Rev	CATGTATTCTTCTTTTT	NO	NO	NO	NO	UbD
22	22730534	For	GCACCCATTAATGTCTG	NO	NO	NO	Yes_NA_0.277231638418079	UbD
22	21344032	Rev	CCACCTGCGCCCTGGCT	NO	NO	NO	NO	UbD
22	21344019	Rev	GGCTCCCCGTGCCCCTC	NO	NO	NO	Yes_rs7410545_0.00136217287212086	UbD
22	21344018	Rev	GCTCCCCGTGCCCCTCA	NO	NO	NO	Yes_rs12162762_0.000463177396943029	UbD
22	22899363	Rev	CATGTATTCTTCTTTTT	NO	NO	NO	NO	UbD
22	22730534	For	GCACCCATTAATGTCTG	NO	NO	NO	Yes_NA_0.277231638418079	UbD
22	21344032	Rev	CCACCTGCGCCCTGGCT	NO	NO	NO	NO	UbD
22	21344019	Rev	GGCTCCCCGTGCCCCTC	NO	NO	NO	Yes_rs7410545_0.00136217287212086	UbD
22	21344018	Rev	GCTCCCCGTGCCCCTCA	NO	NO	NO	Yes_rs12162762_0.000463177396943029	UbD
22	21344002	Rev	ACCTCCTCTACCTGCGC	NO	NO	NO	Yes_rs7410444_0.150414937759336	UbD
22	21332394	Rev	ATGCCTAACAGTGTCCA	NO	NO	Yes	NO	UbD
22	21064356	Rev	TCAGCCAGTCCACAGGA	NO	NO	Yes	NO	UbD
22	19163577	For	CTGGAATCGTGAGACAA	NO	NO	Yes	NO	UbD
22	19109753	Rev	TCCGGCCTTCCAGCCGC	NO	NO	Yes	Yes_rs11550628_0.280899506505159	UbD
22	18898874	For	GGGGCAGAG	NO	NO	NO	NO	Ub
22	18084001	Rev	AGTGGGTTATGAGCCGA	NO	NO	Yes	NO	UbD
X	147919063	For	ATTAGGTCAGTTAAGTT	NO	NO	Yes	NO	UbD
X	118603773	Rev	GCGAAGTTAAGAGCCTG	NO	NO	Yes	Yes_rs148920941_0.191180343906385	UbD
X	118603773	For	CAGGCTCTTAACTTCGC	NO	NO	Yes	Yes_rs148920941_0.191180343906385	UbD
X	118603747	Rev	GTATCTGATGACATTGG	NO	Yes_FP	Yes	Yes_rs141428607_0.16845486170909	UbD
X	118603747	For	CCAATGTCATCAGATAC	NO	Yes_FP	Yes	Yes_rs141428607_0.16845486170909	UbD
X	118603742	Rev	TGATGACATTGGCCAGG	NO	Yes_FP	Yes	Yes_rs148294496_0.171104150352388	UbD
X	118603742	For	CCTGGCCAATGTCATCA	NO	Yes_FP	Yes	Yes_rs148294496_0.171104150352388	UbD
X	107400223	For	GTGGCAGGT	NO	NO	Yes	Yes_rs199981380_0.21503608812304	Ub
X	82764040	For	AGGTTCGAAGGCTTGCA	Yes	Yes_TP	Yes	NO	UbD
X	71495388	Rev	ATGGGTAAGAAGTTTTA	Yes	Yes_TP	Yes	NO	UbD
X	71495388	For	TAAAACTTCTTACCCAT	Yes	Yes_TP	Yes	NO	UbD
X	69749852	Rev	TTGGATTATAGAAAGAC	Yes	Yes_TP	Yes	NO	UbD
X	69478801	For	ACAGGCGACGAGCCATG	NO	Yes_FP	NO	Yes_rs202027224_0.0137229086489771	UbD
X	52891818	Rev	ACAGAGGTCTAACTGAA	NO	NO	Yes	NO	UbD
X	52891739	Rev	GTGTCCTTTGCCTACCT	NO	NO	Yes	NO	UbD
X	52891738	Rev	TGTCCTTTGCCTACCTT	NO	NO	Yes	NO	UbD
X	52891736	Rev	TCCTTTGCCTACCTTTA	NO	NO	Yes	NO	UbD
X	52891732	Rev	TTGCCTACCTTTAATGT	NO	NO	Yes	NO	UbD
X	52891584	For	GTTTATTTCGATATTTT	NO	Yes_TP	Yes	NO	UbD
X	52891565	For	TTTGGAGAAAGCTGCAA	NO	NO	Yes	NO	UbD
X	52891542	For	GAACAAAAAATGTGCAA	NO	NO	Yes	NO	UbD
X	52891523	For	AAATATTTGGGAATAAA	NO	Yes_TP	Yes	NO	UbD
X	52891513	For	CAATAGTACCAAATATT	NO	Yes	Yes	NO	UbD
X	52891512	For	TCAATAGTACCAAATAT	NO	Yes_TP	Yes	NO	UbD
X	35821055	Rev	GATGAGCATTCTGGGCT	Yes	Yes_TP	Yes	NO	UbD
X	23689636	Rev	TAAGAAAATAAAATACA	NO	Yes_TP	Yes	NO	UbD
X	23689636	For	TGTATTTTATTTTCTTA	NO	Yes_TP	Yes	NO	UbD
X	15262814	Rev	TTTTTTTTAAAAAAAAG	NO	NO	Yes	Yes_NA_0.0712582781456954	UbD
X	15262813	Rev	TTTTTTTTAAAAAAAAG	NO	NO	Yes	Yes_NA_0.0712582781456954	UbD
X	1317599	Rev	GTGTGTGTATGTGTATG	NO	NO	NO	NO	UbD

***UbD=Upstream motif + Error base + Downstream**

The remaining columns contain some additional annotation for the reported RSE: present in the NIST list ("InNIST"), in the Broad list ("InBroad"), and in the DBall ("InDBall(rsid_MAF)") along with rsid and MAF. Moreover, column "InHC" shows that this RSE is called by another pipeline where HaplotypeCaller (HC) is used for variant calling (cf. Chapter 4). The last column ("MotifLoc") shows three different positional forms of 17mer (or 9mer): UbD=Upstream motif + Error base + Downstream motif, Ub= Upstream motif + Error base, and bD= Error base + Downstream motif. Furthermore, this table has two different parts, the left part with above-mentioned columns contain information about a RSE that belong to one chromosomal location, whereas, right part contains the information about another RSE on different chromosomal location.

Supplementary material

Supporting information files (S1 to S6)

These files (mentioned in Chapter 3) are available along with online version of the published paper (Kawalia et al., 2015).

URL: http://journals.plos.org/plosone/article?id=10.1371/journal.pone.0126321#sec036

(Accessed on: 21 October 2015)

Erklärung

Ich versichere, dass ich die von mir vorgelegte Dissertation selbständig angefertigt, die benutzten Quellen und Hilfsmittel vollständig angegeben und die Stellen der Arbeit-einschließlich Tabellen und Abbildungen-, die anderen Werken im Wortlaut oder dem Sinn nach entnommen sind, in jedem Einzelfall als Entlehnung kenntlich gemacht habe; dass diese Dissertation noch keiner anderen Fakultät oder Universität zur Prüfung vorgelegen hat; dass sie - abgesehen von unten angegebenen Teilpublikationen - noch nicht veröffentlicht worden ist, sowie, dass ich eine solche Veröffentlichung vor Abschluss des Promotionsverfahrens nicht vornehmen werde.

Die Bestimmungen dieser Promotionsordnung sind mir bekannt. Die von mir vorgelegte Dissertation ist von Herrn Prof. Dr. Peter Nürnberg betreut worden.

Köln, den 17.10.2016

____

Amit Kawalia