KU LEUVEN

ARENBERG DOCTORAL SCHOOL
FACULTY OF ENGINEERING SCIENCE

A development framework for data analytics in genomics



Amin Ardeshirdavani

Supervisor:
Prof. Yves Moreau

Co-Supervisor:
Prof. Joris Robert Vermeesch
Pro. Jan Aerts

Dissertation presented in partial fulfilment of the requirements for the degree of Doctor of Engineering Science (PhD): Electrical Engineering

February 2017

A development framework for data analytics in genomics

Amin Ardeshirdavani

Supervisor:
Prof. Yves Moreau

Co-supervisors:
Prof. Joris Robert Vermeesch
Prof. Jan Aerts

Examination committee:
Prof. Yves Willems, chair
Prof. Bart Van Den Bosch
Prof. Bettina Berendt
Prof. Bart De Moor
Prof. Sonia Van Dooren
(Vrije Universiteit Brussel, Brussels)

Dissertation presented in partial fulfillment of the requirements for the degree of Doctor of Engineering Science (PhD): Electrical Engineering

February 2017

© 2017 KU Leuven – Faculty of Engineering Science
Uitgegeven in eigen beheer, Amin Ardeshirdavani, Kasteelpark Arenberg 10, bus 2446, B-3001 (Belgium)

Alle rechten voorbehouden. Niets uit deze uitgave mag worden vermenigvuldigd en/of openbaar gemaakt worden door middel van druk, fotokopie, microfilm, elektronisch of op welke andere wijze ook zonder voorafgaandelijke schriftelijke toestemming van de uitgever.

All rights reserved. No part of the publication may be reproduced in any form by print, photoprint, microfilm, electronic or any other means without written permission from the publisher.

Preface

Although only one name appears on the title page as the author, undoubtedly, this work could have never been accomplished without the help and support of several individuals and groups to whom I owe my sincere gratitude. Their help was not limited to practical and scientific contributions, but beyond that, it extended to the spiritual support, making it much easier for me to pursue my goals passionately. It was my great honor to work with you.

First and most, I'm grateful to my supervisor, Prof. Yves Moreau, for the sustained encouragement, guidance, advice, and support he generously provided during this research work. Thanks to your foresight, expertise, generosity, and patience, I become passionate about the fascinating world of bioinformatics. The joy and enthusiasm you have in research was motivational for me and made my Ph.D. experience productive and stimulating. Every single meeting or even casual discussion that we had together during the past few years (even on our road trip from Strasbourg to Leuven you tried to solve an equation on a piece of paper) was informative and energizing. This was exactly what I needed to face the numerous challenges that arose in the course of my work.

I also would like to express my sincere gratitude to my co-supervisors, Prof. Joris Robert Vermeesch and Prof. Jan Aerts for your absolute support during my study. Prof. Vermeesch without your kindness support in motivating members of BeMGI to participate in NGS-Logistics, we were not able to achieve our goals in genomics data sharing. Prof. Aerts, you are always full of ideas, not only in the field that you are well known for – data visualization – but also for nice ideas on how we should and must have a better and more attractive workplace.

For this dissertation, I would like to express my great appreciation to the advisory committee and all the members of my Ph.D. examination board: Prof. Yves Willems, Prof. Bart Van Den Bosch, Prof. Bettina Berendt, Prof. Bart De Moor, Prof. Sonia Van Dooren. I am appreciating your comments and questions during my preliminary defense to challenge and help me to improve the quality of my thesis. The most important thing I learned from you all was never is late for learning.

The members of the STADIUS group at the department ESAT of KU Leuven, including the professors, senior and junior, former and current, post-docs, Ph.D. colleagues, and technical staff, also contributed immensely to my personal and professional experience at STADIUS. To me, this group is not only one of the leading research centers but also at the same time the best source of friendship and collaboration where I made some of the best friends of my life. I would like to thank Aldona, Elsy, Ida, Liesbeth, John, Maarten, Mimi, and Wim for being the kindest, warmest and the most helpful staffs of STADIUS. Adam, Alejandro, Arnaud, Bobis, Daniel, Daniele, Dusan, Gorana, Houda, Inge, Jaak, Jansi, Leon, Maira, Marc, Marin, Mauricio, Oliver, Rio, Sarah, Supinya, Thomas, Xian, Yousef; Thanks for being an amazing colleague and friend, thanks for all the moments that we spent together and for all the fun we have had in the last four years. I also would like to thank the technical staff of the ESAT, IT group, especially Rik for their supports.

Griet, Raf, Veronique, Jeroen, Matthias and Anneleen. You all always have been there for me from the first date that I came to Belgium and helped me to integrate to Belgian society. You are the most amazing classmates I ever had. I was so lucky to have you all. Raf, thank you very much for all the translations I needed for official letters and for teaching me most of the Belgian rules and regulations. Griet, working with you was a nice experience for me, thanks for all the motivations you gave me and for your time and efforts you spent for the project we did together and I am happy that the Dutch translation of the abstract, your final contribution to my Ph.D., will remain with me forever.

Special thanks to all the members of BeMGI, and the NGS-Logistics center admins: Luc, Jeroen, Leonor, Raphael, Didier, and Geert. I would like to also thank the UZ Leuven and VSC system admins for their great and kindness supports: Herman, Mario, Martijn and Jan.

My sincere thanks also goes to Prof. Marc De Maeyer, Prof. Arnout Voet, and Dr. Joren De Raeymaecker, who provided me an opportunity to join their team as a master student.

I would like to express my deepest sense of gratitude to Erika for our useful scientific discussions and for patiently helping me during past four years. Apart from being my Ph.D. comrade, you kindly dedicated part of your time to improve the quality of my thesis.

My time in Leuven was made enjoyable in large part due to the good friends that become part of my life. They never let my wife and I feel alone. I'm incredibly thankful to all of you: Farhad, Elham & Sina, Armaghan & Siavash, Fatemeh and Masoud, Anali & Mohsen, Soosan, Paquita, Maria, Behta, Hassan, Farshid, Shahram, Rohangiz & Ahmad, Mana & Javad, Maryam & Kambiz, Forouz & Nemat, Parinoosh & Darush, Armita, Mohammadreza, Sima & Hamed, Aram & Ali, Nasima & Majid, Rouzbeh, Salim, Mohsen, Mahsa & Hossein, Baharak & Babak, Hamed, Shirin. Members of ESAT.IR: Amin, Iman, Neda & Amirhossein, Mojtaba, Zahra & Reza, Sina, Siamak and Amir. That's a great pity that I am not able to acknowledge all my good friends in this limited space the way they truly deserve but you should know that you are always in my mind and heart.

Pooya, from the first day of my Ph.D. we always had our own type of communication. You are one of the people I can truly trust and count on no matter what. Thank you very much for all the good times we shared together.

I must express my very profound gratitude to my in-laws Baba Vahab, Maman Badri, Peyman, Mahsa, Iman, Shahram, Soudabeh, Saina, Bahar and Kobra for providing me with unfailing support and continuous encouragement.

Most importantly, I would like to thank my family to whom I owe a great deal. To my late father, Borzou thank you for showing me that the key to life is enjoyment. I wish you were here today. To my mother Akhtar, thank you for your continuous support and encouragement, without you, I would never have been able to achieve my goals. You are a great inspiration to me. Also thanks to my sister Nooshin, my brother-in-law Babak and my nephew Alireza. Nooshin, my great appreciation and enormous thanks to you, you were the one suggested me the world of Bio-Informatics, thanks for pushing me in this direction. To my brother Afshin, you are the kindest person that I know, thank you very much for your invaluable support.

And finally, who has made all this possible, my love and my life Bita. You have been a constant source of support and encouragement. You made an untold number of sacrifices for me. Words cannot describe how lucky I am to have you in my life. You have selflessly given more to me than I ever could have asked for. I love you and look forward to our lifelong journey with our beloved one Hana.

Amin Ardeshirdavani

February 2017
Leuven, Belgium

دل گرچه در این بادیه بسیار شتافت یک موی ندانست ولی موی شکافت
اندر دل من هزار خورشید بتافت واخر به کمال ذره‌ای راه نیافت

(ابوسعید ابوالخیر - ابن سینا)

Though my heart had in this wilderness its full share,
It did not know a thing; yet it could split many a hair.
In my heart, myriad suns did shine bright,
Yet, it failed to fathom a single particle alright.

(Abusa'id Abolkhayr – Avicenna)

Abstract

Next-generation sequencing (NGS) is a disruptive technology. The enormous throughput of NGS instruments has mandated the development of a new generation of algorithms and data formats capable of storing, processing, and analyzing massive amounts of sequence data. NGS has led to an increase by several orders of magnitude in biological data available for genomics and transcriptomics. Therefore, extensive bioinformatics frameworks have to be developed to ensure a correct biological interpretation of this data. The primary focus of this PhD work is the development of sophisticated bioinformatics frameworks to store, process and integrate biological data. These data may have different sources, including curated public data repositories or in-house generated data. When the data sources are available, some steps need to be taken. Firstly, they need to be stored in a structured way. Secondly, depending on the question under investigation, they need to be analyzed in an integrated way or separately. Finally, the result must be returned in a way that users can gain as much as information they need in a simple and informative format. In practice, each one of these steps has its own challenges. In this thesis, we propose a novel comprehensive approach to tackle all these challenges in a user-friendly fashion.

A key achievement in NGS bioinformatics was the specification of the Sequence Alignment/Map (SAM) format and its binary equivalent (BAM). These formats have been primarily introduced to the scientific community in order to have a standard global format to store and process aligned sequencing reads. To study inherited and acquired human genetic disorders, researchers need to find the relation between single nucleotide mutations and disorders, specifically the relationship to phenotypes of patients. Therefore, variant call format (VCF) files were developed to encode the genetic variations in a text file format. This format allows scientists to finalize their sequencing analysis - no matter which sequencing technologies they used - into a single standard format. Since VCFs take much less storage space than BAM files, we have the opportunity to collect and store many of these files. By having access to enough clinical/biological samples, researchers may gather enough evidence to confirm their hypotheses.

The goal of this thesis has been to develop a bioinformatics workflow (pipeline) for multi-step processing of human genome sequencing data, from the data generated by the sequencing instrument to making the data ready for interpretation by a genomics specialist. Our suggested pipeline consists of several steps. Each step has two sets of parameters: input and output. The input parameters come from the previous step and need to be analyzed in the current step, whereas the output parameters are generated by the current step and passed on to the next step. Each one of these steps includes a number of tools that perform the main processing in this step. The key steps that we will explain in more detail are preprocessing and storage, quality control, alignment and mapping followed by compression, sorting and indexing, variant calling and genotyping. After these steps the results are annotated and finally, by filtering the list of variations, we make a list of candidate variations ready for interpretation.

One of the most important challenges that we address in this work is to secure the data. Since we are talking about human genome information, in all circumstances data must be

kept safe from unwanted and unauthorized access. Therefore, to secure our pipeline and the data generated in every step, we propose a data model that allows us to run an access control list application on top of it to monitor and manage user access to the data.

As already mentioned, by having access to more data and samples, researchers can generate more accurate results. Finding and allocating samples and data sets that match relevant to a given study is a complex and time-consuming procedure. Even if you find a data set with the kind of attributes you are looking for, there is no guarantee that this data set has the exact information you need. Finally, getting access to data that you assume to be useful for your research is difficult as well. Most of the data controllers are not in favor of shipping their data outside of their network, as they know that after transferring the data to another location, they will not have control on the data anymore. To address these issues, we designed an infrastructure for data sharing, in the sense that access to the data is provided so that it can be analyzed but that data is not sent outside the center that produced it. The question here is: why sharing? First of all, by sharing data, the data will stay at the center that produced it. In this way, data controllers retain control over who has access to the data, how, and most important to which data. Secondly, to start an inquiry you often do not need to have access to the entire data. By simply answering some elementary questions you can already gain enough information to only spend time on the subjects you need to focus on. To improve the quality and performance of such analyses, we have started to implement a reliable infrastructure for large-scale processing of sequencing variants. Our design aims on the one hand at storing information in a compact and efficient way by considering the structure and requirements of the task in the best possible way and on the other hand at offering a wide-range of dynamic reports and outputs based on different kind of filters via a web application. We named our data sharing platform *NGS-Logistics*.

Finally, a large part of this PhD project has been in connection with designing and developing the data structures and software packages. Each one of these applications has their difficulties and their own complexity, from the data level and infrastructure complexity to results visualization difficulties. However, a critical aspect of any project is data. When generating data, one should always think about how to store the data, which technologies should be used, and how generate queries to answer all relevant questions. As a consequence, when designing, developing and maintaining tools, you always face different difficulties here referred to as *complexity*. In this thesis, we show how to tackle such complexity. In this work, we also demonstrate how we use new techniques and combine them to analyze massive amounts of data generated by NGS instruments. Huge amounts of complex data are known as Big Data. We discuss how we used Apache Hadoop to store and query data that we were not able to analyze in a short time by ordinary techniques, such as text processing or relational databases.

Beknopte samenvatting

Next-generation sequencing (NGS) is een disruptieve technologie. De enorm hoge doorvoer van NGS-toestellen vereiste de ontwikkeling van een nieuwe generatie algoritmen en gegevensformaten geschikt voor het opslaan, verwerken en analyseren van de grote hoeveelheid aan sequentiedata. NGS bracht binnen genomics en transcriptomics een toename in biologische data van meerdere grootteordes teweeg. Om een correcte biologische interpretatie van deze data te kunnen garanderen, is een gefundeerde bio-informatica-gebaseerde omkadering noodzakelijk. De voornaamste focus van dit proefschrift betreft de ontwikkeling van een gesofisticeerd bio-informatica raamwerk voor het opslaan, verwerken en integreren van biologische data. Deze gegevens kunnen afkomstig zijn van verschillende bronnen, waaronder gecureerde publieke databanken of in-house gegenereerde data. Wanneer deze databronnen beschikbaar zijn, dienen bepaalde stappen ondernomen te worden. In de eerste plaats dienen de data op gestructureerde wijze te worden opgeslagen. Ten tweede, afhankelijk van de eigenlijke onderzoeksvraag, dienen de gegevens geïntegreerd of afzonderlijk te worden geanalyseerd. Tot slot dienen de resultaten zo gerapporteerd te worden dat de gebruikers alle nodige informatie verkrijgen op eenvoudige doch informatieve wijze. In de praktijk heeft elk van deze stappen zijn eigen uitdagingen. In dit proefschrift stellen we een nieuwe alomvattende benadering voor om al deze uitdagingen op een gebruiksvriendelijke manier aan te pakken.

Eén van de belangrijkste verwezenlijkingen binnen de NGS-bio-informatica is de specificatie van het sequentie-alignering/map formaat (SAM) en diens binaire equivalent (BAM). Beide formaten werden voornamelijk geïntroduceerd om over een standaard gegevensformaat te beschikken voor het opslaan en verwerken van gealigneerde sequencing reads. Bij het bestuderen van humane genetische aandoeningen gaan onderzoekers op zoek naar de relatie tussen single nucleotide mutaties en deze aandoeningen, en meer specifiek naar het verband met de fenotypes van patiënten. Om zulke genetische variaties te coderen in tekstformaat werd het variant call formaat (VCF) bestand in het leven geroepen. Dit bestand laat wetenschappers toe de resultaten van hun sequentie-analyse – ongeacht de gebruikte sequencing-technologie – in een standaardformaat te beschrijven. Omdat VCF's heel wat minder opslagruimte innemen in vergelijking met BAM-bestanden, wordt het mogelijk een groter aantal bestanden te verzamelen en op te slaan. De toegang tot voldoende klinische/biologische stalen stellen onderzoekers in staat tot het statistisch staven van hun onderzoekshypothesen.

Het doel van dit proefschrift was de ontwikkeling van een bio-informatica-workflow (pipeline) voor de meerstappenverwerking van humane genoom sequencing-data vanaf het punt waarop de data gegenereerd worden door de sequencing-machine tot het voorbereiden van de data voor interpretatie door een genomics-specialist. De door ons voorgestelde pipeline bestaat uit meerdere stappen. Elke stap wordt gedefinieerd door twee attributen: input en output. De inputparameters zijn afkomstig van de vorige stap en worden geanalyseerd in de huidige stap, terwijl de outputparameters gegenereerd worden in de huidige stap en doorgegeven worden aan de volgende stap. Elk van deze stappen omvat een aantal tools voor het uitvoeren van de dataverwerking in de betreffende stap. De voornaamste stappen die hier in detail aan bod komen zijn voorbewerking, opslag, kwaliteitscontrole, alignering en mapping, gevolgd door compressie, sorteren en

indexeren, variant calling en genotypering. Na het uitvoeren van deze stappen worden de resultaten geannoteerd en tenslotte wordt, door het filteren van de lijst met variaties, een lijst met kandidaat variaties voor verdere interpretatie opgesteld.

Eén van de belangrijkste uitdagingen die we aanpakken in dit proefschrift is het beveiligen van de data. Daar het hier humane genoomdata betreft, dienen deze data in alle omstandigheden beschermd te worden van ongewenste en onbevoegde toegang. Om de data gegenereerd in de verschillende stappen van onze pipeline veilig te stellen, introduceren we een datamodel dat ons toelaat de data te onderwerpen aan een access control list applicatie voor het controleren en beheren van de gebruikerstoegang tot de data.

Zoals reeds vermeld kunnen onderzoekers nauwkeurigere resultaten genereren wanneer ze toegang hebben tot meer data en stalen. Het vinden en toewijzen van stalen en datasets die overeenstemmen met een bepaalde studie is een ingewikkelde en tijdrovende procedure. Zelfs wanneer een dataset met de gewenste karakteristieken gevonden wordt, is er geen garantie dat deze dataset exact de nodige informatie bevat. Daarenboven blijkt ook het verwerven van toegang tot data die bruikbaar geacht worden vaak moeizaam te verlopen. De meeste onderzoekers die over data beschikken zien deze niet graag verspreid buiten hun netwerk, daar ze nog slechts over weinig controle beschikken wanneer de gegevens zich op een andere locatie bevinden. Om dit probleem aan te pakken hebben we een infrastructuur ontwikkeld voor het delen van data, op zo'n manier dat toegang tot de data verschaft wordt om analyse van de data mogelijk te maken, maar zonder het verzenden van de data buiten het centrum dat deze produceerde. De vraag hier is: waarom data delen? In de eerste plaats kunnen de data hierdoor in het centrum dat ze geproduceerd heeft blijven. Zo controleert de eigenaar van de data steeds wie toegang heeft tot de data, op welke manier en vooral tot welke data. Ten tweede, om een onderzoek te starten is vaak geen toegang tot de volledige dataset vereist. Door het simpelweg beantwoorden van enkele elementaire vragen kan vaak reed genoeg informatie verzameld worden om enkel verder te gaan met de relevante data. Voor het verbeteren van de kwaliteit en performantie van zulke analyses zijn we gestart met de implementatie van een betrouwbare infrastructuur voor het verwerken van sequencing-varianten op grote schaal. Ons ontwerp beoogt aan de ene kant het opslaan van informatie op compacte en efficiënte wijze door het optimaal in rekening brengen van de structuur en de vereisten van de taak, en aan de andere kant het aanbieden van een breed scala aan dynamische rapporten en outputs gebaseerd op verschillende soorten filters via een webapplicatie. Ons platform voor het delen van data kreeg de naam *NGS-Logistics*.

Tot slot spitste een groot deel van dit PhD-project zich toe op het ontwerpen en ontwikkelen van gegevensstructuren en softwarepakken. Elke applicatie kent bepaalde moeilijkheden en beschikt over een bepaalde complexiteit, van de infrastructuur tot de visualisatie van resultaten. Het voornaamste aspect van elke project blijft echter de data zelf. Reeds bij het genereren van data dient nagedacht te worden over hoe deze data op te slaan, welke technologieën gebruikt dienen te worden en hoe de data te bevragen om alle onderzoeksvragen te beantwoorden. Bij het ontwerpen, ontwikkelen en onderhouden van applicaties dient dus steeds rekening te worden gehouden met deze zogenaamde *complexiteit*. In dit proefschrift tonen we aan hoe dit kan. We demonstreren tevens hoe we nieuwe technieken voor het analyseren van grote hoeveelheden NGS-data gebruiken en combineren. Grote hoeveelheden complexe data zijn gekend onder de naam big data. We beschrijven hier hoe we de Apache Hadoop gebruikt hebben voor het opslaan en bevragen

van data die niet binnen een aanvaardbare tijdspanne geanalyseerd kunnen worden door middel van standaardtechnieken als tekstverwerking en relationele databanken.

چکیده

نسل جدید تعیین توالی های ژنتیکی (NGS) یک تکنولوژی مبتکرانه است. حجم عظیم خروجی ابزارهای NGS باعث توسعه و ایجاد نسل جدیدی از الگوریتم ها و فرمت داده ها شده است -که قادر به ذخیره سازی، پردازش، تجزیه و تحلیل حجم انبوهی از داده ها، می باشند. NGS منجر به افزایش تصاعدی داده های بیولوژیکی در زمینه های ژنومیکس و ترنسکریپتومیکس شده است؛ بنابراین برای اطمینان از تفسیر صحیح بیولوژیکی این داده ها باید ساختارهای گسترده ای از علم بیوانفورماتیک ایجاد شوند. تمرکز اصلی این رساله دکتری ایجاد ساختارهای پیچیده بیوانفورماتیک برای ذخیره سازی، پردازش و ادغام داده های بیولوژیکی می باشد. این داده ها ممکن است از منابع مختلفی از جمله مکان های عمومی ذخیره سازی عمومی داده ها و یا داده های تولید شده در-محل بدست آمده باشند. هنگامی که منابع داده در دسترس باشند، باید مراحل گوناگونی طی شود. نخست، داده ها باید به صورت سازمان یافته ذخیره شوند. در مرحله بعد با توجه به سوال درخواست شده، داده ها به طور یکجا یا جداگانه مورد تجزیه و تحلیل قرار گیرند. نهایتا، نتیجه باید به صورتی باشد که کاربران بتوانند حداکثر اطلاعاتی را که نیاز دارند، در قالبی ساده و آموزنده بدست آورند. در عمل هر یک از این مراحل دارای چالشهای خاص خود می باشند. در این پایان نامه ما یک رویکرد جامع و بدیع برای رویارویی با تمام این چالشها را در قالبی کاربر پسند ارئه کرده ایم.

تعیین فرمت توالی SAM و معادل دودویی آن BAM دستاوردی کلیدی در بیوانفورماتیک NGS بود. این فرمتها در درجه اول به منظور تعیین یک فرمت استاندارد جهانی برای ذخیره سازی و پردازش توالی های ژنتیکی تراز شده به جامعه علمی معرفی شده اند. به منظور مطالعه اختلالات ارثی و اکتسابی ژنتیک انسانی، محققان نیاز به پیدا کردن رابطه بین جهش تک نوکلئوتیدی و نا هنجاری ها به ویژه در مشخصات بالینی بیماران دارند. بنابراین فایلهای VCF برای رمزگذاری این تغییرات ژنتیکی در قالب یک فایل متنی ایجاد شده اند. این قالب به دانشمندان اجازه می دهد تا بدون توجه به تکنولوژی تعیین توالی استفاده شده، تجزیه و تحلیل خود را در یک قالب واحد استاندارد نهایی کنند. از آنجایی که این فایل ها فضای کمتری نسبت به فایل های BAM اشغال می کنند، امکان جمع آوری و نگهداری تعداد زیادی از این فایل ها را خواهیم داشت؛ در نتیجه با در اختیار داشتن تعداد کافی نمونه های بالینی و بیولوژیکی، محققان قادر به اثبات آماری فرضیه های خود خواهند بود.

هدف از این پایان نامه، پیاده سازی یک فرایند بیوانفورماتیکی برای پردازش چند مرحله ای داده های توالی ژنوم انسانی تولید شده توسط دستگاهای توالی یابی می باشد؛ به نحوی که توسط متخصص ژنتیک قابل تفسیر باشند. فرایند پیشنهادی ما شامل چندین مرحله می باشد. هر مرحله دارای دو پارامتر ورودی و خروجی می باشد. پارامترهای ورودی از مرحله قبل آمده و در این مرحله نیاز به تجزیه و تحلیل دارند، در حالیکه پارامترهای خروجی در این مرحله ایجاد شده و به مرحله بعد منتقل می شوند. هر کدام از این مراحل شامل تعدادی ابزار است که انجام پردازش اصلی در این مرحله را برعهده دارند. مراحل کلیدی کار که به صورت کامل توضیح داده خواهند شد شامل موراد ذیل می باشند: آماده سازی، ذخیره سازی، کنترل کیفیت، هم تراز سازی به همراه فشرده سازی، مرتب سازی و نشانه گذاری، مشخص کردن جهش ها و گونه ژنتیکی آنها . پس از این مراحل، نتایج دسته بندی گردیده و در نهایت با فیلترکردن لیست جهش ها، لیستی نهایی از جهش های آماده برای تفسیر تهیه می گردد.

امنیت داده ها یکی از مهمترین چالش هایی که در این کار به آن پرداخته شده است، از آنجایی که ما در مورد اطلاعات ژنوم انسانی بحث می کنیم، این داده ها باید در برابر دسترسی های ناخواسته و غیر مجاز در تمامی شرایط محافظت گردند. بنابراین برای تامین امنیت روال برنامه و داده های تولید شده در هر مرحله، یک مدل داده که به ما اجازه می دهد تا عملکرد کاربران و چگونگی دسترسی آنها را بررسی کنیم، پیشنهاد شده است.

همانطور که پیشتر ذکر شد، با داشتن دسترسی به داده ها و نمونه های بیشتر، محققان می توانند نتایج دقیق تری را تولید کنند. پیدا کردن نمونه ها و مجموعه اطلاعات مطابق و مربوط به یک مطالعه، فرایندی پیچیده و سنگین است. حتی اگر شما مجموعه اطلاعاتی مطابق با ویژگی های مورد نظر را پیدا کنید، هیچ تضمینی وجود ندارد که این مجموعه دارای اطلاعات دقیق مورد نیاز شما باشد. در نهایت دسترسی به اطلاعاتی که به نظر شما برای انجام تحقیقاتتان می تواند مفید باشد، دشوار خواهد بود. بسیاری از کنترل کنندگان داده ها تمایلی به انتقال داده های خود به خارج از شبکه تحت کنترل خود ندارند؛ چراکه پس از انتقال دادها به مکانی دیگر، کنترل داده ها در اختیار آنها نخواهد بود. برای حل این مشکل، ما زیرساختی را برای به اشتراک گذاری داده ها طراحی کرده ایم؛ به این صورت که دسترسی به داده ها طوری تامین شده است که بدون انتقال به خارج از مرکز تولید کننده قابل تجزیه و تحلیل باشند. در اینجا سؤال این است: چرا به اشتراک گذاری؟ اول اینکه، با به اشتراک گذاشتن داده ها، آنها در مرکز تولید کننده باقی خواهند ماند؛ در نتیجه، کنترل کنندگان حق چگونگی دسترسی به داده را برای خود محفوظ می دارند. دوم اینکه، برای شروع، غالبا نیازی به دسترسی به کل داده نیست. در واقع با پاسخ دادن به برخی سوالات اولیه می توانید اطلاعات کافی برای تمرکز بر روی موضوعات خاص تر بدست

آورید. به منظور بهبود عملکرد وکیفیت این فرایند، ما شروع به پیاده سازی یک زیرساخت قابل اعتماد برای پردازش جهش های توالی در مقیاسی بزرگ کرده ایم. هدف از این طراحی، از یک سو ذخیره سازی اطلاعات به روشی جامع و کارآمد -با در نظر گرفتن ساختار و الزامات کار در بهترین حالت ممکن- و از سویی دیگر با ارایه طیف گسترده ای از گزارش های پویا از طریق یک محیط مبتنی بر وب می باشد. ساختار به اشتراک گذاری داده ها *NGS-Logistics* نامگذاری شده است.

در نهایت، بخش بزرگی از این پروژه دکترا در ارتباط با طراحی و توسعه ساختارهای داده و بسته های نرم افزاری می باشد. هر یک از این برنامه ها دارای مشکلات و پیچیدگی های خاص خود هستند. با این حال یکی از جنبه های اساسی هر پروژه داده های آن می باشد. هنگام تولید داده، باید همیشه به چگونگی ذخیره سازی داده، فن آوری مورد استفاده و چگونگی ایجاد پرسش ها برای پاسخ به سوالات مربوطه توجه شود. در نتیجه هنگام طراحی، توسعه و نگهداری برنامه ها با مشکلات گوناگونی مواجه خواهید شد -که در اینجا از آنها به پیچیدگی تعبیر خواهد شد. در این پایان نامه، ما چگونگی مقابله با این پیچیدگی ها را نشان خواهیم داد. همچنین نحوه استفاده از روش های جدید و ترکیب آنها برای تجزیه و تحلیل هجم وسیعی از داده های تولید شده توسط ابزارهای توالی یابی، نشان داده خواهد شد. به مقدار قابل توجهی از داده های پیچیده، "کلان داده" گفته می شود. همچنین در مورد چگونگی استفاده از تکنولوژی Hadoop برای ذخیره سازی و بررسی داده هایی که در یک زمان کوتاه و با استفاده از روش های رایج مانند پردازش متن و پایگاه های داده قابل تجزیه و تحلیل نمی باشند، توضیحاتی ارائه خواهد شد.

Abbreviations

ACL	Access Control List
API	Application Program Interfaces
AUC	Area Under the ROC curve
BWA	Burrows-Wheeler Aligner
CAF	Complexity Adjustment Factor
CDB	Centralized DataBase
CLI	Command Line Interface
DAC	Data Access Control
DAM	Data Access Management
DHT	Distributed Hash Table
EGA	European Genome-phenome Archive
FC	Functional Complexity
FDBS	Federated DataBase System
FDR	False discovery rate
HDFS	Hadoop Distributed File System
HGMD	Human Gene Mutation Database
HPO	Human Phenotype Ontology
IDE	Integrated Development Environment
InDel	Insertion and Deletion
GATK	Genome Analysis Toolkit
MAF	Minor Allele Frequency
MSST	Mass Storage Systems and Technologies
NGS	Next Generation Sequencing
NIPT	Non-invasive Prenatal Testing
OMIM	Online Mendelian Inheritance in Man
PI	Principal Investigator
PL	Phred-scaled genotype likelihood
RDBMS	Relational database management system
RDF	Resource Description Framework
SA	Suffix array
SAM	Sequence alignment/map
SNP	Single Nucleotide Polymorphism
SNV	Single nucleotide variation
VCF	Variant call format

Contents

Preface I
Abstract V
Beknopte samenvatting VII
چکیده XI
Abbreviations XV
Contents XVII
List of figures XXI
List of tables XXIII
Chapter 1 Introduction 1
1.1. DNA sequencing 2
1.2. Clinical genome interpretation 4
1.3. Genomic data processing 5
1.4. Thesis overview 8
Chapter 2 Material and methods 9
2.1. Variant calling pipeline 9
2.1.1. Choosing the right platform and strategy 9
2.1.2. NGS workflow 11
2.1.3. Sequencing reads 11
2.1.4. Analysis pipeline 12
2.1.4.1. Storage 13
2.1.4.2. Quality control 15
2.1.4.3. Mapping 16
2.1.4.4. Compression, sorting and indexing 18
2.1.4.5. Variant calling and genotyping 19
2.1.4.5.1. SAMtools' mpileup 20
2.1.4.5.2. FreeBayes 21
2.1.4.5.3. GATK 22
2.1.4.5.4. Variant caller comparison 23
2.1.4.6. SNV annotation 24
2.1.4.6.1. ANNOVAR 24
2.1.4.6.2. eXtasy 25
2.1.4.7. SNV filtering 26
2.1.4.8. Interpretation 26
2.2. Access Control List (ACL) 28
2.2.1. Data access committee 28
2.2.2. Data access management 28
2.2.2.1 Data layer 30
2.2.2.2 Business layer 31
2.2.2.3 Presentation layer 32
2.3. Collaboration 33
2.3.1. Locating and getting access to the data 33

2.3.2. Data access solution ... 34
2.3.2.1 Centralized databases ... 35
2.3.2.2 Federated databases ... 35
2.3.2.3. Peer to Peer (P2P) ... 38
2.4. Complexity ... 39
2.4.1. Functional complexity ... 39
2.4.1.1. Problem complexity ... 40
2.4.1.2. System design complexity ... 41
2.4.1.3. Procedural complexity ... 46
2.5. Big Data ... 48
2.5.1. Apache Hadoop ... 49
2.5.1.1. MapReduce ... 50
2.5.1.2. HDFS ... 50
2.5.1.3. Apache HBase ... 51
2.5.1.4. Apache Hive ... 52
2.6. Project descriptions ... 53
Chapter 3 NGS-Logistics ... 55
3.1. Abstract ... 56
3.2. Background ... 56
3.2.1. Data sharing ... 57
3.2.2. Privacy ... 57
3.2.3. Data management ... 58
3.2.4. Data storage and processing ... 58
3.3. Aims ... 59
3.4. Implementation ... 59
3.4.1. Administration ... 60
3.4.1.1. Sample list ... 61
3.4.2. Query manager ... 61
3.4.3. User interface ... 61
3.4.3.1. User settings ... 61
3.4.3.2. Query builder ... 62
3.5. Results and discussion ... 62
3.5.1. Use Case 1 ... 62
3.5.2. Use Case 2 ... 65
3.6. Conclusions ... 65
Chapter 4 GALAHAD ... 69
4.1. Abstract ... 70
4.2. Introduction ... 70
4.3. Input ... 71
4.4. Methods ... 71
4.4.1. Quality control ... 72
4.4.2. Data exploration ... 72
4.4.3. Differential expression ... 72
4.4.4. Drug target prioritization ... 72
4.4.5. Enrichment ... 73
4.5. Output ... 73
4.5.1. Quality control and data exploration ... 73
4.5.2. Differential expression ... 75
4.5.3. Drug target prioritization ... 76
4.5.4. Enrichment ... 76

4.6. Implementation ... **78**
4.7. Example ... **78**
4.8. Conclusion and outlook ... **78**
Chapter 5 SNIPT ... **79**
5.1. Abstract ... **80**
5.2. Introduction ... **80**
5.2.1. Variant discovery ... 80
5.2.2. Sequencing coverage ... 80
5.2.3. Integration of Apache Hive and HBase ... 81
5.3. Data use case ... **82**
5.4. Aims ... **82**
5.5. Methods ... **82**
5.6. Results and discussion ... **84**
5.6.1. Performance and comparison ... 84
5.6.2. System power and results ... 86
5.7. Future work ... **88**
Chapter 6 Endeavour ... **89**
6.1. Abstract ... 90
6.2. Introduction ... 90
6.3. Endeavour methodology ... 91
6.4. Evaluation results ... 94
6.5. Conclusion ... 95
Chapter 7 Conclusions ... **97**
Bibliography ... **101**
Appendix A: Glossary ... **111**
Curriculum vitae ... **119**
List of publications ... **121**

List of figures

Figure 1: Data, the fundamental object behind all projects. 1
Figure 2: Sequencing cost. 3
Figure 3: World map of high-throughput sequencers. 3
Figure 4: Biological data generation pathway at the clinical level. 4
Figure 5: Software development components. 5
Figure 6: Data consistency lifecycle objects. 6
Figure 7: Thesis outline. 8
Figure 8: Illumina NGS sequencing workflow. 10
Figure 9: FASTQ file sample - one read. 11
Figure 10: GATK best practice variant detection workflow. 12
Figure 11: Variant calling pipeline. 13
Figure 12: Data storage mechanism. 14
Figure 13: Data storage structure. 14
Figure 14: FastQC results - graphs of comparisons between Illumina data of good and poor quality. 16
Figure 15: Assembly vs. mapping. 17
Figure 16: Example of the SAM file format. 17
Figure 17: Example of Pileup file. 20
Figure 18: Comparison between variant calling tools and SNPs reported in dbSNP. 23
Figure 19: Example of ANNOVAR exonic variant annotation. 25
Figure 20: Example of ANNOVAR variant annotation. 25
Figure 21: Main modules of data access management systems. 29
Figure 22: Organization of samples from PIs into two groups: Diagnostics and Research. 29
Figure 23: Relation between DAM entities. 29
Figure 24: Multi-tier application architecture. 30
Figure 25: Database entity-relationship model for data access management. 31
Figure 26: Screenshot of the NGS-Logistics ACL management tool. 32
Figure 27: Applying for data through dbGaP/EGA process. 34
Figure 28: Data sharing mechanisms. 35
Figure 29: Federated system and its nodes. 36
Figure 30: Five-level schema architecture of an FDBS. 37
Figure 31: Functional complexity factors. 39
Figure 32: Relation between problems, algorithms, and programs. 40
Figure 33: NGS-Logistics: variants calling workflow. 41
Figure 34: eXtasy web application structure. 43
Figure 35: *GALAHAD* web application structure. 45
Figure 36: *GALAHAD* data structure ER diagram. 45
Figure 37: Structure and components of the NGS-Logistics platform. 47
Figure 38: A simplified Hadoop cluster. 49
Figure 39: A MapReduce process. 50
Figure 40: HDFS data nodes. 50

Figure 41: A schema of a data cell in HBase. 51
Figure 42: The architecture of the Apache integration. 53
Figure 43: NGS-Logistics components. 60
Figure 44: NGS-Logistics user types and their access levels. 60
Figure 45: NGS-Logistics area query results page (Position to Sample section) for SMARCA2 63
Figure 46: Single point query result page (Statistics section) for chr9:2115841. 63
Figure 47: Single point query results page (Sample to SNV section) for chr9:2115841. 64
Figure 48: Single point query results page (Sample to SNV section) for chr9:108363420 and chr9:108397495 64
Figure 49: Overview of the different *GALAHAD* analysis steps. 71
Figure 50: *GALAHAD* diffusion-based drug target prioritization performance. 73
Figure 51: Quality control. 74
Figure 52: Data exploration. 74
Figure 53: Differential expression. 75
Figure 54: Drug target prioritization. 75
Figure 55: Drug-gene network. 76
Figure 56: Enrichment tab: Pathway enrichment and disease enrichment. 77
Figure 57: Integration Hive – HBase. 81
Figure 58: NIPT variant calling pipeline. 83
Figure 59: Hadoop cluster performance. 86
Figure 60: Query pane: Interactive interface on top of the integration model. 87
Figure 61: Position vs. coverage relation. 87
Figure 62: The Endeavour algorithm. 92
Figure 63: Endeavour result page. 92

List of tables

Table 1: NGS sequencing platforms. 9
Table 2: Coverage recommendation by application. 10
Table 3: SAM file - mandatory fields. 18
Table 4: VCF data section - mandatory fields and VCF example. 21
Table 5: Variant caller statistics. 23
Table 6: List of public databases. 27
Table 7: Project descriptions 54
Table 8: Performance comparison for three solutions - 1 sample. 85
Table 9: Performance comparison for three solutions - 200 samples 85
Table 10: Results of the leave-one-out cross-validation on “gold standard” gene sets. 95
Table 11: System use report. 98

Chapter 1

Introduction

Thousands of years ago when our ancestors tried to record the events of their life by painting on cave walls, the first piece of a thing that nowadays we call *data* was produced. Data is inseparable from our life. It could be generated and stored in different ways. The way in which we handle data is critical. If we take a look at the history of computer science, we easily notice that how much technological advances affect the way that we produce and store data [1]. Since the invention of the first computer, three key questions keep recurring: "How should we store this large amount of data?" and "How should we analyze it?" and "How should we represent the results?" (Figure 1). Consequently, today's buzz word of "Big Data" was always there, no matter how much disk space we have and how much computing power we have, we are always facing data that is bigger and more complex than we expected [2]. The true reason is that the more data we have access to, the more knowledge we can gain, which is the key to tackling our problems [3].

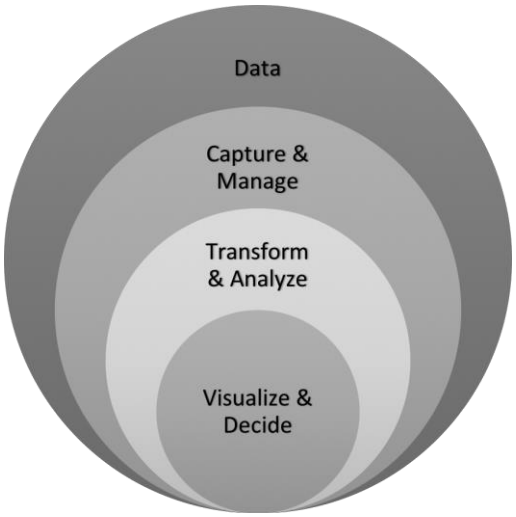


Figure 1: Data, the fundamental object behind all projects. A key element for a successful project is how data is to be captured, managed, and used. Raw data is of limited value until we analyze it with a well-defined algorithm and transform it into a model that we can use to take decisions**.**

Materials and methods are varied in different projects, but all projects have things in common: there is always a goal, and there are many limitations. To get close to the goal, we should have a clear plan to overcome the limitations [4]. Therefore, choosing the best design path and method is essential for a successful project. There are different types of application design methods and each one has its advantages and disadvantages [5]. Since computers have become a crucial part of every project, there is a big interest in using more powerful software to get more accurate results and analyze data faster. We do not only need better software, but we also need better hardware [6]. So at every step of the design,

we need to take into account all these aspects: users, instruments, network, storage devices, and so on.

Software designers and developers used to store the data that was both directly related to a project and really needed in the processing phase, and they did their best to answer the questions one at a time. During the past ten years, the situation has entirely changed. As more powerful processors and storage devices entered the market, people started storing as much data as possible [7]. The philosophy has moved from "store what you need to analyze" to "store first, analyze later". In this way, we are now facing not only huge amounts of data but also the possibility to tackle a myriad of complex problems together, which is the exact definition of Big Data [8].

Today life sciences are on the edge. Advanced technologies help scientists to generate more and more data by analyzing their biological samples, which means we produce more and more data every day. One of these technologies is Next-Generation Sequencing (NGS). Recently NGS has become an essential tool in the research and diagnosis of human Mendelian, oligogenic, and complex disorders [9]. It is not only used in human research projects, but also in research to elucidate the complexity of all life forms [10]. Data produced by NGS can be categorized as *huge* and *complex* [11]. Several steps must be completed to analyze this data [12]. As a result, in any research institute, there is a pressing need and interest to have procedures to store, organize and prepare the data [12]. As a result, bioinformatics has become essential to analyze data in an effective way. Scientists with knowledge and skills in both biology and Information Technology (IT) can play an important role in this area.

1.1. DNA sequencing

To understand how cells function, we need to understand the genetic information "stored" in the "format" of the DNA sequence in the genome. To identify this sequence, we must somehow read this information from the cell. DNA sequencing is the process of determining the precise the order of the four nucleotides (Guanine, Adenine, Thymine, and Cytosine; G, A, T, and C) within DNA molecules [13]. By sequencing DNA, researchers gain essential knowledge to understand the nature of all living species. Since 1970, when DNA sequencing was carried out for the first time, numerous applied fields have started to use this information [14]. Technological advances in different areas of biology, chemistry, and physics have resulted in faster and more accurate methods. While sequencing was long based on Sanger sequencing technology, which was slow and expensive at the scale of the human genomes, new technologies eventually emerged to tackle sequencing at this scale. These techniques are often called *Next-Generation Sequencing* (NGS) [15]. NGS has revolutionized genomic research in nearly every field of biology [15]. Different NGS platforms have been developed commercially. For example, Roche (454), Illumina/Solexa (HiSeq and MiSeq), Life Technologies/ABI/Ion (SOLiD, Ion Torrent and Ion Proton), Helicos Biosciences, and Pacific Biosciences are offering or have offered such instruments. These instruments use different technology and have each their specific characteristics, differing in their structure, configuration, reagents, and read length [15, 16]. In the Materials and Methods chapter, NGS technology will be discussed in more details.

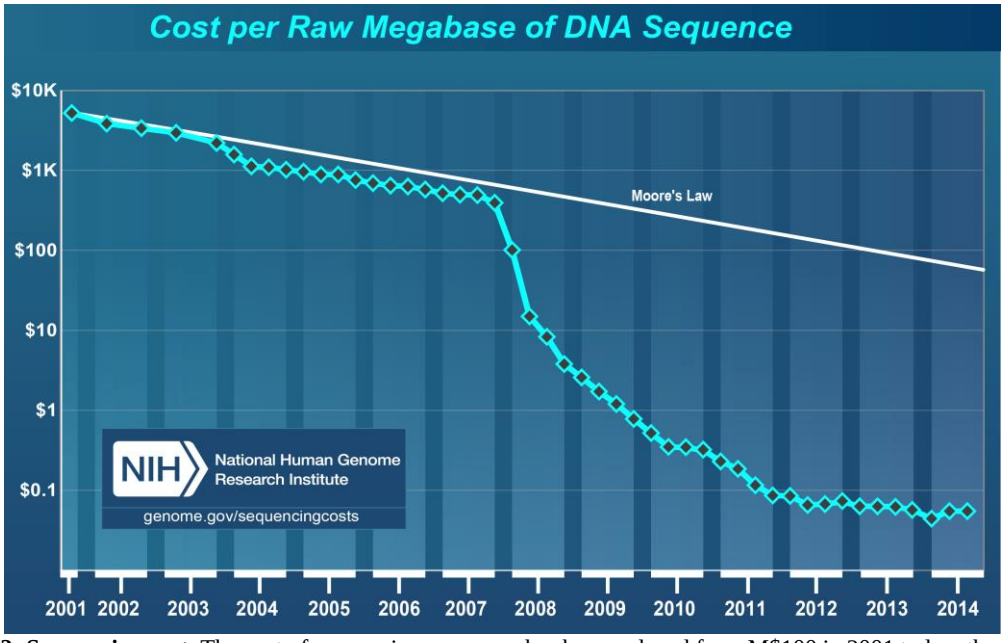


Figure 2: Sequencing cost. The cost of sequencing a genome has been reduced from M$100 in 2001 to less than $1,000 today. This cost includes administration, consumables, write-off of the sequencer, and informatics costs. (Reprinted from the NIH website: The cost of sequencing a human genome [17].)

In 2003, by using traditional Sanger sequencing, researchers completed the sequencing of the first human genome-sequencing after 13 years of effort and at a $2.7 billion expense [9]. Now, NGS makes it possible to sequence an entire human genome in days [16]. Thanks to the new NGS technologies, the costs of sequencing have also decreased sharply (Figure 2). The cost of sequencing a genome has been reduced from M$100 in 2001 to about $1,000 this year [17]. By mid of 2015, there were over one thousand centers worldwide with NGS facilities and over 2,700 machines are installed and operated at these centers (Figure 3) [18].



Figure 3: World map of high-throughput sequencers. As of March 2015, there were 2,758 NGS instruments at 1,027 centers. (Reprinted from http://omicsmaps.com/.)

1.2. Clinical genome interpretation

Geneticists and physicians will increasingly refer patients suffering from inherited and acquired human genetic disorders for genome sequencing if their condition warrants it. In multiple situations, genome sequence information can significantly improve the diagnostic process and help guide therapeutic decisions. In clinical use of NGS, lab technicians will then prepare the samples and run the test. As soon as the raw data becomes available, bioinformaticians can start to analyze the data, create an appropriate report, and send the results back to the clinic for diagnosis and follow-up (Figure 4). During the past ten years, multiple projects have contributed to the mapping of human genomic variation on a large scale using NGS, such as the Thousand Genomes Project [19], the UK 100k Genome Project [20], or The Genome of the Netherlands [21]. As a result, about 100,000 human exomes have been sequenced so far and over 25 billion bases of human genomic DNA sequence generated [22]. However, the flood of sequencing data requires massive computational power and storage, as well as optimized programming structures to handle this data efficiently. Because DNA consists of the 4 nucleic acids (A, G, C, and T), its naïve storage requires at least 2 bits per base (ASCII storage would even be 1 byte of storage per base) [23]. Considering that the number of reads sequenced in each run of an NGS instrument can reach several billions, the raw data of a run will often require hundreds of gigabytes or several terabytes of storage. The processing of the raw data into primary data as an assembled genome requires at least a terabyte per genome.

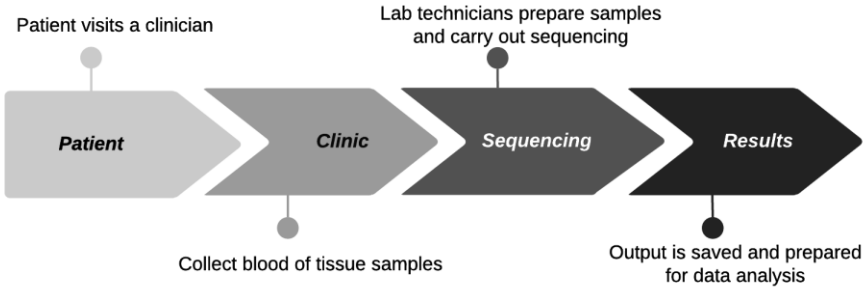


Figure 4: Biological data generation pathway at the clinical level. When a patient visits the clinic, the physician collects the sample appropriate for sequencing based on the pathology of the patient. Samples will be prepared by lab technicians, sequencing is then carried out, and finally the result of sequencing is stored for data analysis.

The change of a single nucleotide in the DNA sequence of an individual with respect to the reference genome is called a Single Nucleotide Variation (SNV). Such changes can sometimes have serious pathological consequences, while many often then only have mild or negligible effects. SNVs that occur frequently in a population are called Single Nucleotide Polymorphisms (SNP). While most of the genome sequence is identical between most people (typically, two people will have about 99.9% of their genome sequence identical), there are typically significant differences between the genome sequence of individuals [24]. The 0.1% variation between the genome of an individual and a human reference genome still translate into about ten million variants. The genome sequence of every individual is the genetic fingerprint of this person. Most of this variation consists of common SNPs with no or limited pathological consequences (*e.g.*, risk factors). Even between healthy individuals from the same population, there is a large difference of 20,000 SNVs between any two of them [25]. A single nucleotide change can happen at any location in the genome. SNVs that are located in the coding region of a gene can often be

responsible for unwanted events [26]. If an SNV does not affect the sequence of the resulting protein, this mutation is categorized as a *synonymous* mutation, while *non-synonymous* mutations change the amino acid sequence of a protein [27, 28]. Non-synonymous mutations fall into two groups: missense and nonsense mutations. Missense mutations result in different amino acids, while nonsense mutations cause some code to change into a stop codon and results in the truncation of the protein. Damage to the sequence of a protein can significantly affect its ability to carry out its functions in a living organism and potentially result in severe pathological consequences. Therefore, it is evident that it is both essential and challenging to interpret the genomic information correctly for clinical or research use.

1.3. Genomic data processing

Outstanding data processing is needed to extract reliable information from sequencing data [12]. Because human genome data is a sensitive type of information and in all cases must be well protected, hospitals and research institutes often prefer to analyze data themselves [29]. To do so, they prepare their own hardware and network infrastructure. First, they need to hire professionals to maintain the network and servers. Second, they need software developers to design applications for data processing. To automate the data processing task, software developers design a chain of processes called pipeline [30]. In a pipeline, different tools and applications may be combined. There will be a parent application designed to control the entire process. The output of every step will be used as an input for the next phase till the final results are ready. The software development lifecycle (SDLC) [31] is a process consisting of several steps to ensure quality and correctness of that software. Depending on the type of SDLC, the lifecycle can have different steps. All the steps of the pipeline lifecycle are necessary, important and one wrong step can cause a serious mistake in the development of the software. To choose the right SDLC, some decisions must be taken. Designers and developers must decide which type of data they need to handle, how they are going to collect and store this data, what the problems and questions are that they must tackle, and, importantly, which application and development language they are going to use for development (Figure 5).

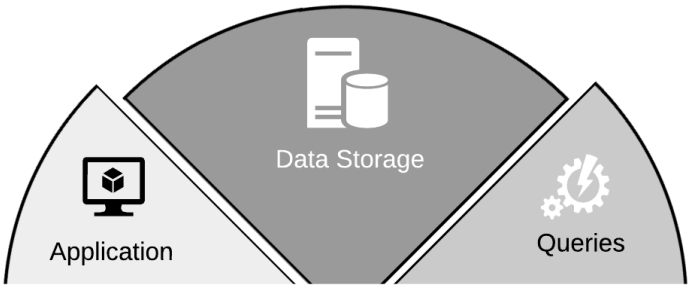


Figure 5: Software development components. Software designers and developers work closely together to clearly understand the problem and decide what the best options are to store the data collected and what the best tools and platforms are to analyze this data.

As said before, data is the fundamental object in every such project (Figure 1). When we consider data-centric applications, we need a consistent data lifecycle. Data consistency fits within these three parts [32]: The first part is how we capture and manage the different types of relevant data. Once we have collected data, the second part of the lifecycle is how we transform and analyze the data. This is where a lot of productive analytics come in and

are executed. Finally, the third part in a consistent data lifecycle focuses on interaction with the users and visualization. Users get the possibility to see information through a grid or an excellent visualization, and they can decide about the things they want to focus on. When we look at these three dimensions as a whole and talk about applications and organizations that are embracing data, each of these dimensions can be divided into different methods and technologies. Organizations, depending on their situation and goal, may use various sources for every aspect of their projects. They try to connect and combine different tools and methods to achieve their goals. An example of such a combining platform is presented in Figure 6. In the data capture layer, different tools may be used to store data, such as relational and non-relational databases, NoSQL techniques – or data could directly be streaming from another source inside or outside of the organization.

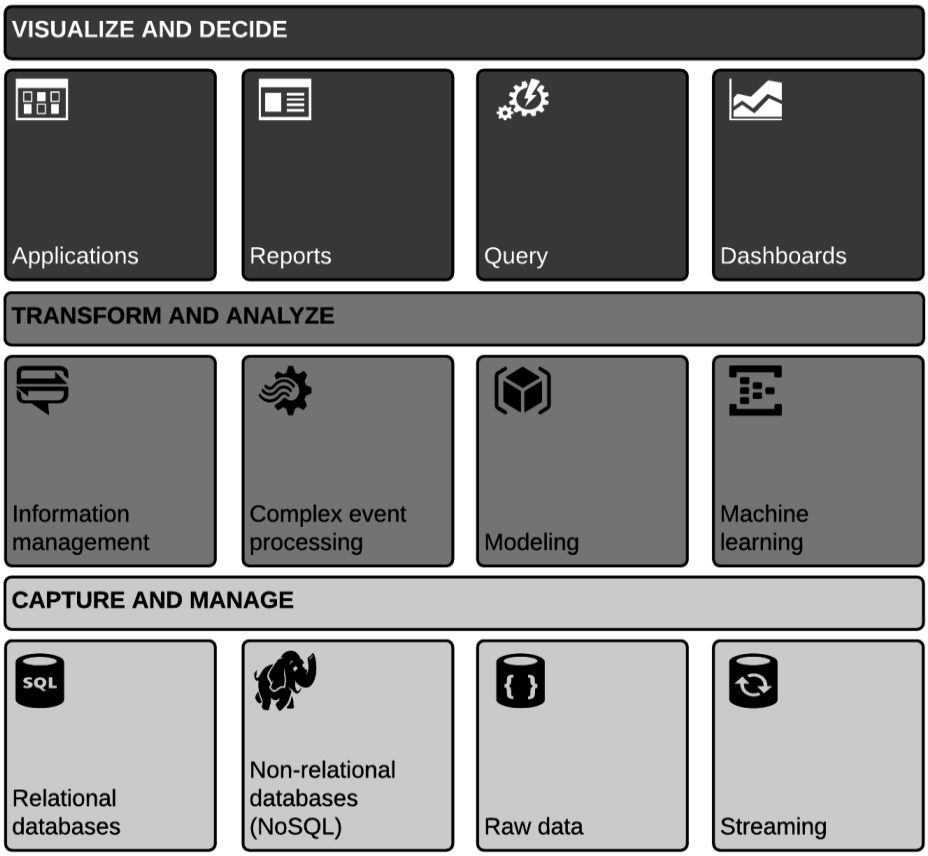


Figure 6: Data consistency lifecycle objects. In data-centric applications, at the base layer data can be stored and managed by different methods such as relational databases, non-relational databases, NoSQL, Raw data or directly streamed from the source. In the second layer, raw data will be analyzed and transformed to another format by applying different solutions. Finally, results are presented to the final user in different formats.

One level up, data must be managed. Data access rules and roles need to be defined by the administrators of the system. Complex processing procedures need to be designed and developed. To get the most out of the data, sometimes data analysts create complex data structures and run machine-learning algorithms on the data to predict relevant events and get the best possible overview of the relevant information present in the data. To organize data, software developers build applications, generate reports, and simplify results to provide relevant information through interactive visual components, as well as by making

Application Program Interfaces (API) that allow users to run structured queries on the data to which they have access rights.

From the engineering point of view, the whole system must work fast; components must be updatable and adaptable to new conditions and new versions. This means that the system must be sufficiently agile and – last but not least – the correct functionality of every single process in the system must be validated. In this thesis, tackle all these challenges on the systems that we developed, which allows us to capture data produced by NGS technology, analyze this data, and make the information and result available for the users.

As mentioned before, to process genomic information generated by NGS instruments, several steps must be carried out, which we will discuss in more details in the Materials and Methods section. Meanwhile, we should keep in mind that the data generated by sequencing machines consists of large and complex files. When we look at the single nucleotide variation in a human genome, the collection of files resulting from the completed analysis of a single whole genome study can require up to 50GB of disk space [33]. Therefore, a well-designed infrastructure with enough computational power is needed. One of the main values of genome data is that it can be reused in multiple clinical contexts. By choosing the right development method, developers design reusable functions that allow them to make the data ready to answer new questions by making limited changes to the code. Therefore, different queries may return significantly different results from each other. Of course, software at a certain point will need to be upgraded to a completely new version or even fully redesigned. By developing modular applications, developers can minimize the time and cost spent on the development of new software [34].

1.4. Thesis overview

In this thesis, we aim to formalize the way in which we handle data generated by NGS and other genomic technologies. We will go through all the aspects of a concrete solution to implement a successful method. In the following text, we discuss in more details aspects of data generation, data management, data security, genomics data sharing limitation and benefits, speed up of the data analysis, and last but not least data visualization.

The Introduction describes the research background and explains the motivation for pursuing this work and provides essential background information for the work presented in this thesis. Materials and Methods details of the challenges we had to face to analyze NGS and other genomics data, as well as the solutions. From chapter three to six, we demonstrate the validity of our methods by demonstrating four successful applications. Chapter seven concludes the work and discusses how genomics data analysis can help our community, which actions must be taken, and how we could handle data in more secure and robust ways.

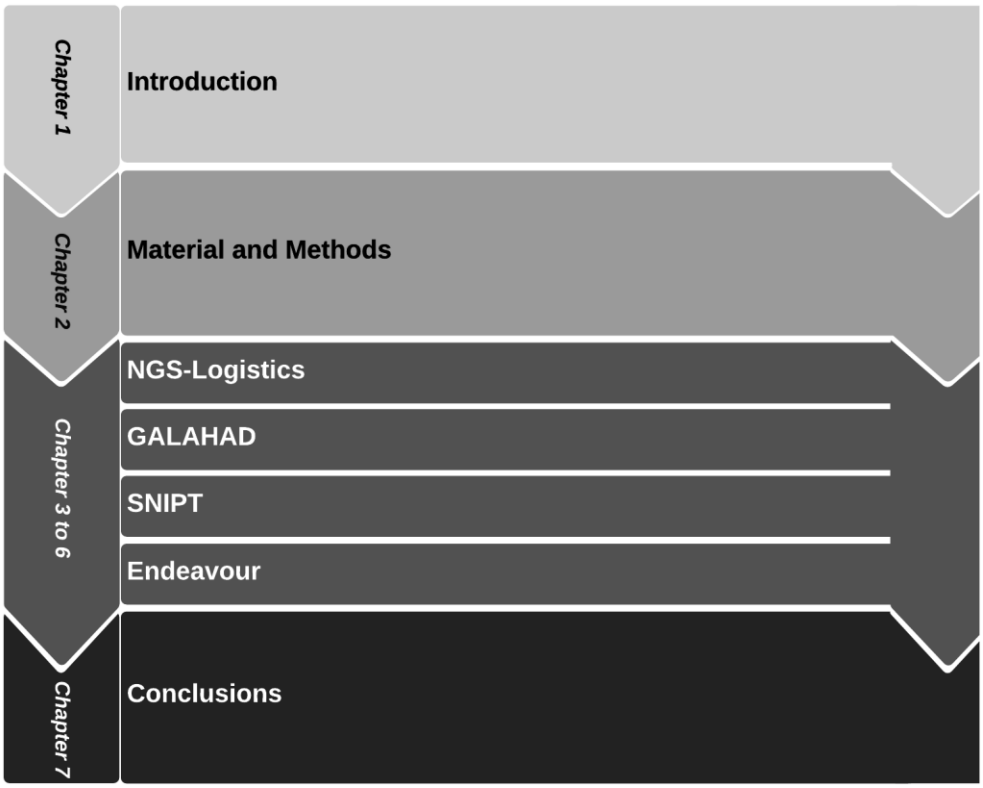


Figure 7: Thesis outline.

For the convenience of readers less familiar with bioinformatics, we supply a glossary of bioinformatics terms in Appendix A.

Chapter 2

Material and methods

In this chapter, we introduce the NGS data analysis pipeline for variant calling and the computational infrastructure that we leverage for our analyses. We will then introduce the tools and methods we used for design and development. Additionally, we will list the challenges that we tackled to share genomic data. Finally, we will conclude by explaining the methods that we used to analyze massive and complex data and visualize the results. Since the focus of this project is human genomic data, all data and the information are human related, even if the word "human" is not mentioned.

2.1. Variant calling pipeline

In Chapter 1, we introduced NGS and its benefits. Our focus in this thesis is on Single Nucleotide Variants (SNVs). Most of the human genome is identical between individuals. Indeed, almost 99.9% of our genome is identical to the genome of other individuals in and outside of our population [24]. Variants such as SNVs contribute to genetic differences within and among populations. Some changes in even one single nucleotide can cause differences between identical twins [35]. Therefore, the study of human genetic variation is critical for medical applications, as well as large-scale population genomics studies, such as creating population-based variation databases.

2.1.1. Choosing the right platform and strategy

The choice of the right NGS sequencing platform is based on the aims of the study. Platforms differ not only regarding their turn-around time and price but also, and more importantly, in terms of accuracy and read length. Different companies, such as Illumina, Pacific Biosciences (PacBio), SOLiD, and Ion Torrent have developed their NGS instruments for commercial use. They mostly differ in their structure, configuration, agents and read length, and thus in their applications (Table 1) [36].

Table 1: NGS sequencing platforms. The aims of the study and the read length that we need are the most important factors to take into account when deciding which platform to use.

Platform	Read Length	Ideal for
Illumina	250 – 600	Sequencing of new genomes, resequencing, transcriptome analysis, SNP detection and metagenomic studies.
Ion	200 – 400	Sequencing of small genomes, targeting of small regions within a genome.
PacBio	4,600 – 14,000	Sequencing of small (new) genomes, resequencing of large regions (for haplotyping for example) and G/C rich regions, methylation analysis.
SOLiD	100	Sequencing of new genomes, resequencing, transcriptome analysis, SNP detection and metagenomic studies.

From now on, in this thesis, our focus will be only on Illumina because it is currently the dominant platform for large-scale sequencing efforts. Illumina HiSeq instrument generate

billions of reads at low cost, but these reads are short. The main drawback of this method is that many overlapping reads are needed to reconstruct accurately the genome/region of interest. Moreover, the fragmentation is random and the coverage uneven. Therefore, to cover all bases of interest, many reads have to be generated. Thus, as a consequence, the size of the resulting FASTQ and BAM files is linearly related to the number of reads and the size of these files can grow enormously (much larger than the size of human exomes, which we will discuss it in details, in the Compression, Sorting, and Indexing section).

Table 2: Coverage recommendation by application. Since the read coverage has a significant impact on both quality and cost, choosing an appropriate level of coverage is essential **[37]**.

Application	Coverage	Application	Coverage
Whole genome – Homozygous SNVs	15x	Whole exome – Homozygous SNVs	100x
Whole genome – Heterozygous SNVs	33x	Whole exome – Heterozygous SNVs	100x
Whole genome – INDELs	60x	Whole exome – INDELs	-

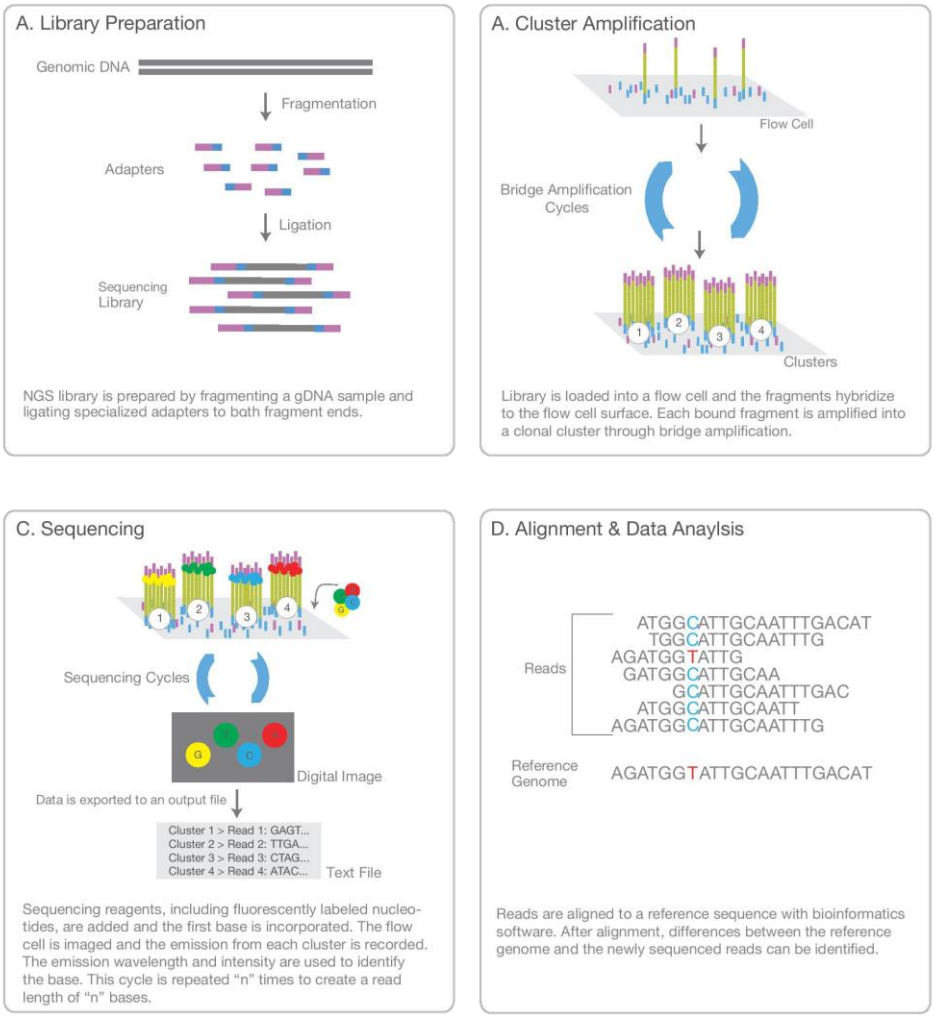


Figure 8: Illumina NGS sequencing workflow. First, a sequencing library is prepared by fragmenting the large piece of DNAs into smaller fragments and adding adapters to these short fragments. Second, the library is amplified into clusters. Third, sequencing cycle through fluorescent reaction steps that capture each base of a cluster in digital images and processes these images. Fourth, mapping, alignment, and variant calling are used to analyze the sequencing data. (Reprinted from the Illumina website: An introduction to Next-generation Sequencing Technology [38].)

After read length, the most important factors that can affect the study are the sequencing coverage and read depth. The depth of coverage is the average number of reads aligned to known reference bases, assuming that reads are randomly distributed across the genome. In general, more coverage means that each base is covered by a larger number of sequencing reads. Based on type of analysis required, different levels of coverage are recommended to get reliable results (Table 2) [39].

2.1.2. NGS workflow

Different steps take place before the data is generated by the instrument (Figure 8) [38]. In this thesis, the details of the NGS chemistry will not be described. Instead, we focus on the last step: the data analysis.

As mentioned in the previous section, the data analysis methods are highly dependent on the type of study. Some steps, such as image analysis and read alignment, may be common to some or all studies. Different tools and steps will take place till the final results are obtained. In the following section, we explain and cover all the steps necessary for the variant discovery pipeline.

2.1.3. Sequencing reads

The first output generated by Illumina instruments is a high-quality image taken by the device, in which each base is color-coded [15]. Each color represents a base and during the image analysis step, or base calling, colors are converted to the associated base. The sequence of bases from a single short DNA fragment is called a read. Sequencing reads and their quality scores are stored in a FASTQ file [40]. Each read is described in four lines (Figure 9). The first line starts with an "@" and is the read identifier. The second line is the DNA sequence. The third line starts with "+" sign and eventually contains the read identifier. The last line contains the Phred quality scores [41] of each base coded in ASCII. Phred quality score is the way to demonstrate the quality of each bases of DNA sequence which is basically a log value of the error probabilities:

$$Phred\ score = -10\ \log_{10}(error\ porbability)$$

Depending on the sequencing platform, different ways are being used for assigning phred score and they are encoded based on ASCII codes.

Depending on the number of reads and the read length, a single FASTQ file can take up to 200 Gigabytes of disk storage. For example, a whole genome sequence with an average coverage of 38x and a read length of 36 can take 193 GB while an exome sequence with a 40x coverage and a read length of 75 will only take 8 GB [42]. Sequencing can be done in paired-end which means that both ends of each fragment are sequenced, and two FASTQ files are then generated. Therefore, it is clear that we need large storage resources and computational power to analyze this data.

```
1. @SRR001666.1 071112_SLXA-EAS1_s_7:5:1:817:345
2. GGGTGATGGCCGCTGCCGATGGCGTCAAATCCCACC
3. +SRR001666.1 071112_SLXA-EAS1_s_7:5:1:817:345
4. IIIIIIIIIIIIIIIIIIIIIIIIIIIII9IG9IC
```

Figure 9: FASTQ file sample - one read. Each sequencing read is represented by four lines: the first one starts with "@" and contains the read identifier; the second one contains the sequence of the read; the third starts with "+" and repeats the read identifier; and the forth contains the quality of the sequencing at each base as estimated by the Phred quality score.

2.1.4. Analysis pipeline

Our variant calling pipeline is based on the GATK best practices for SNV calling (Figure 10) [43]. It contains a series of sequential steps. Different software packages can be used for every step. Research groups must choose the best software solution with respect to their goal and their computing resources. Choosing and using the right software is absolutely critical as it may increase the quality of the output, reduce the computing time and at the same time minimize costs. Regardless of the type of software needed, we should consider the following points with respects to our long-term goals:

- Know your needs
- Skills of the development team
- Output supported by other tools
- Security
- Ease of use
- Determine your objectives
- Support and upgrade
- Cost

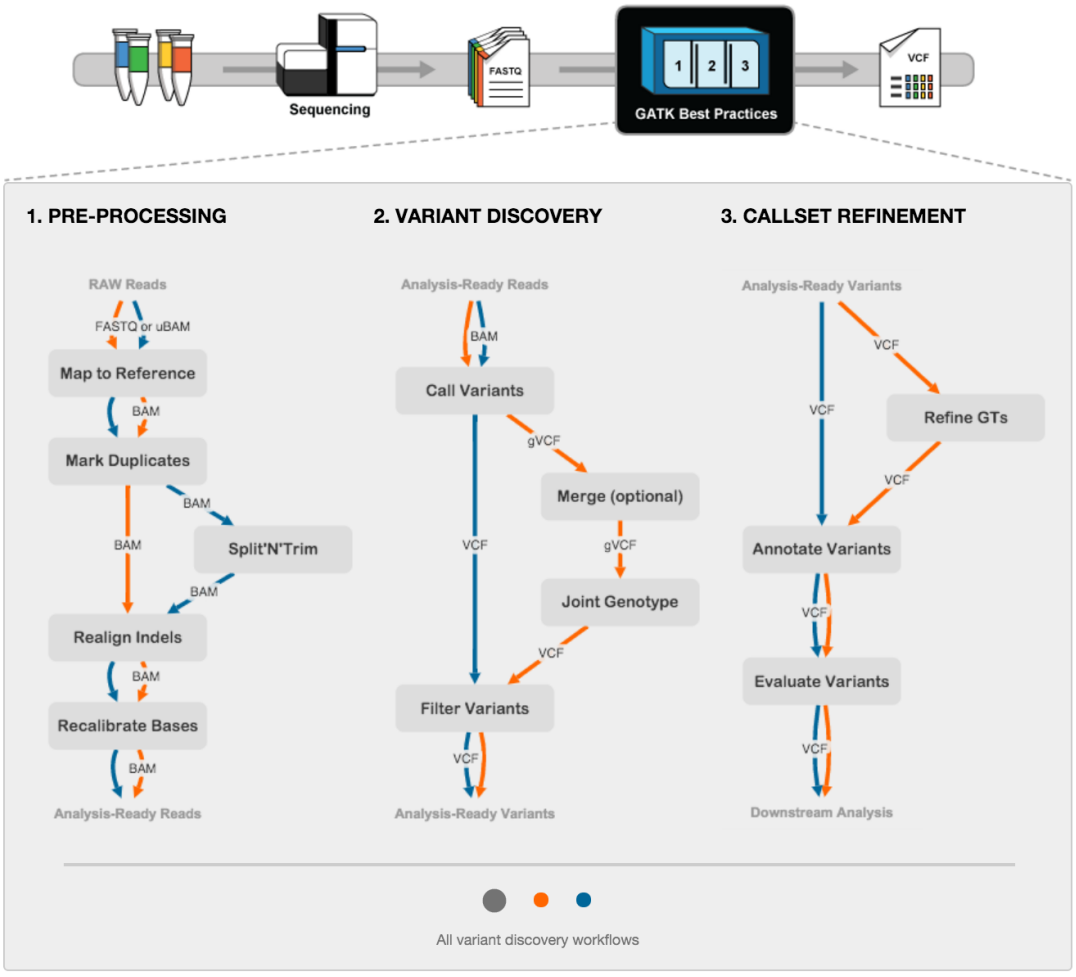


Figure 10: GATK best practice variant detection workflow. The GATK best practice is a recommended workflow for variant discovery analysis. Variant calling consists of three phases: preprocessing, variant discovery, and callset refinement. (Reprinted from GATK, Broad Institute documentation [43].)

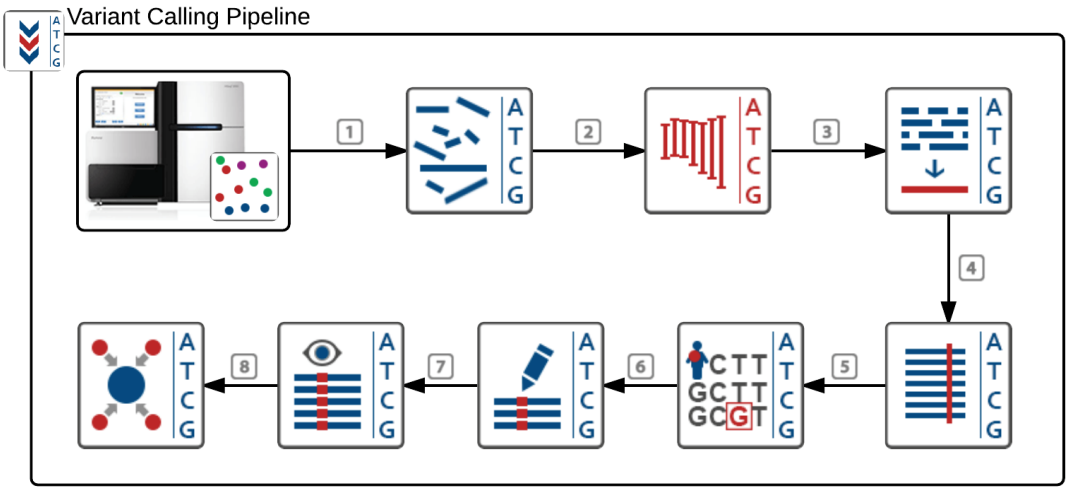


Figure 11: Variant calling pipeline. In the first steps, the raw data is transfered to the storage. In next steps, the quality of data is controled. Once the quality requirements are satisfied, short reads are aligned and mapped to the reference genome to save storage. In the following steps, the files are compressed and prepared for variant calling. By default, every variant callers generate a huge list of variants. To get a more interpretable view of the sequencing data, we need to annotate the variants and filter them to a significantly shorter list .

Our variant calling pipeline consists of eight steps (Figure 11). The different tools used at each of these steps will be discussed more specifically when explaining the steps:

1. Transfer raw data to storage
2. Quality control
3. Mapping
4. Compression, Sorting and Indexing
5. Variant calling and genotyping
6. Annotation
7. Filtering SNP candidates
8. Interpretation

2.1.4.1. Storage

Advances in sequencing technology have resulted in extraordinary growth in research data. Therefore, scale-out network storage solutions [44] increase the throughput of life science and bioinformatics workflows, while reducing complexity and costs. Depending on the size of the research center and the model organism under study, different storage solutions may be applicable. In-house storage clusters are used by bigger organizations and by those working with human data. Cloud-based storage can also be used by research centers or clinics. However, shipping out the data is a time-consuming task. Moreover, clinical centers might prefer to store confidential information in-house to protect the data as much as possible.

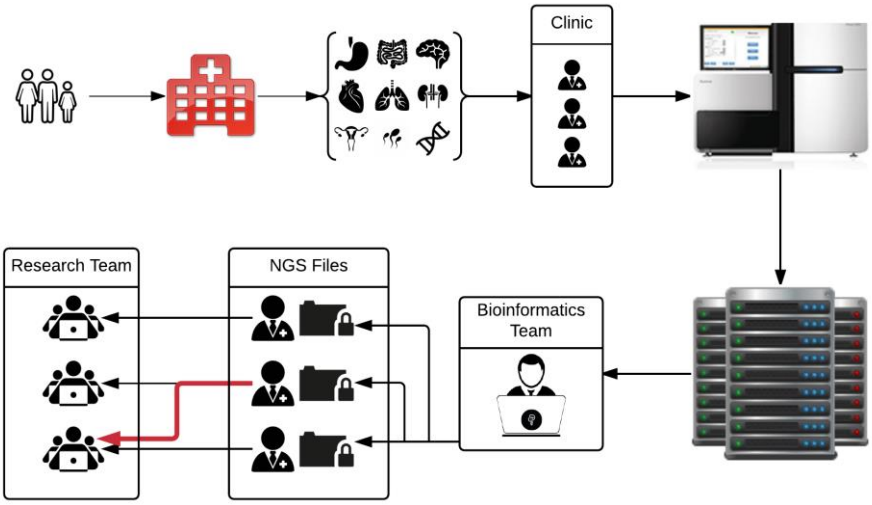


Figure 12: Data storage mechanism. The patient data is stored separately on the data storage devices. Only the data controler or a group of people allowed to work on the same project will have access directly to the corresponding data.

Patients with different diseases and phenotypes visit a doctor at the clinic for proper diagnosis and to evaluate therapeutic options. Unfortunately, understanding the causes of the patient's disease may not be easy, especially in the case of heterogeneous diseases with complex phenotypes. The clinician may therefore want to request the sequencing of the genome to diagnose the patient's phenotype better or to guide treatment. A biological sample is taken from the patient and sent to a sequencing center. The time and complexity of DNA extraction could vary depending on the type of the analysis (whole genome sequencing, exome sequencing, target sequencing, *etc.*). As explained before, the output from the instrument will be a raw file. Primary raw files will be stored in the data storage infrastructure. An IT or bioinformatics expert will automatically or manually start the data analysis (Figure 12). In a well-organized center, data could be stored on the cluster in a structured way with one directory per research group or Principal Investigator (PI). This directory will only be accessible by the users from and under that supervisor. Each sample will be assigned a unique name or ID and referred to as "Sample ID". Each sample folder may contain several subfolders created to categorize the data (Figure 13).

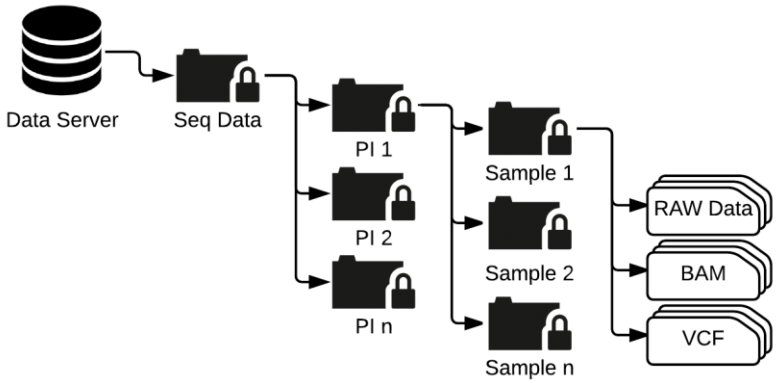


Figure 13: Data storage structure. Each research group has one directory that is accessible by the users from that group. A unique sample identifier is assigned to each sample and is used as the name of the directory containing all the sequencing files (FASTQ, BAM, VCF) for this sample.

The illustration of our data storage (Figure 13) shows that this simple structure requires serious effort and actions by the system administrator to control the users access level. In Section 2.2, the methods used to monitor and control data access by the users will be discussed.

2.1.4.2. Quality control

The NGS instrument generates the DNA reads and data is stored on hard drives. Before starting the data analysis, it is important to control the quality of the data [45]. The accuracy of the results and our final interpretation is directly related to the quality of the raw data. The quality of the reads can be assessed by taking a look at their Phred quality distribution, nucleotide frequencies, and complexity. Quality factors and parameters can vary from one technology to the other and final application of the sequencing [46]. Therefore, to generate high-quality sequencing data, different methods and tools may be used. The most famous tool for FASTQ quality control is FastQC. [47]. The most useful metrics are (Figure 14)

- Per Base Sequence Quality: This shows the quality score of reads as a box-plot, and higher values meaning better quality.
- Per Base Sequence Content: This shows the proportion of each base at each cycle. Normally, this should reflect the GC content of the sequenced species or regions. Deviation from standard base content can indicate issues with library quality.
- Duplicate Sequences: This shows the number of times the same read is seen in a 200,000 read subset of the data.

The raw sequences may contain problematic regions, which can be removed to avoid problems with downstream analyses. Such areas include vectors, adapters, low-quality regions, low complexity regions, contaminants, and duplicates. These parts can be kept if the downstream analysis software can handle this noisy information or would not be negatively affected by it.

The process of noise elimination is known as cleaning. Cleaning programs can be classified as filters and trimmers. Filtering software removes the reads that do not meet the criteria, while trimming software only clip certain regions. Many software tools can process the reads, the main ones being PrinSeq [48], FastX-toolkit [49], *etc*.



Figure 14: FastQC results - graphs of comparisons between Illumina data of good and poor quality. The accuracy of any analysis depends on the quality of the primary data. Several factors must be checked and based on the type of analysis different thresholds must be selected. (Reprinted from FastQC manual [50].)

2.1.4.3. Mapping

Two different kinds of analyzes can be done from high-quality reads. If no previous genomic information is available, the reads can be assembled into a genome (*de novo* assembly) [51]. Otherwise, the reads can be mapped to a reference genome [52] (Figure 15). Assembly is computationally more costly than mapping in terms of time, CPU usage and memory [53]. Here, we focus on the human genome, and since the reference is available, we only need to map the reads against the reference.

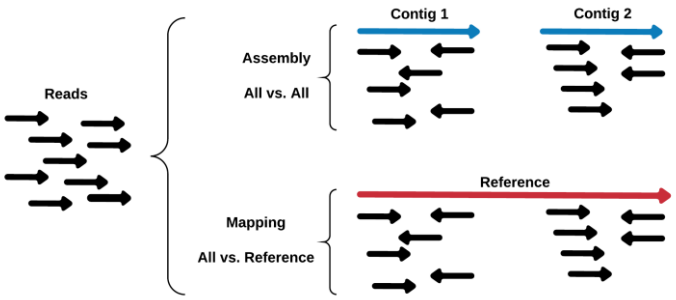


Figure 15: Assembly vs. mapping. In assembly, short reads are merged with each other to create the full-length sequence, whereas in mapping the reads are mapped to an existing reference sequence.

During mapping, the reads are compared to the reference genome. It is the task of the mapping tool to find the position of the read in the reference genome [53] since the FASTQ file does not contain this. Like for any other tasks, multiple mapping software tools are available. The most commonly used programs are Bowtie2 [54] and Burrows-Wheeler Aligner (BWA) [55]. These tools differ on the algorithm used, the sensitivity, the memory requirements, the speed, and the sequence length requirements [56]. In general, all mappers render the result in a common file format called SAM format [57]. SAM stands for Sequence Alignment/Map format. It is a TAB-delimited text format consisting of an optional header section and mandatory fields (Figure 16, Table 3) [57].

```
1.@HD VN:1.5 SO:coordinate
2.@SQ SN:ref LN:45
3.r001    99      ref 7   30  8M2I4M1D3M  =   37  39  TTAGATAAAGGATACTG   *
4.r002    0       ref 9   30  3S6M1P1I4M  *   0   0   AAAAGATAAGGATA      *
5.r003    0       ref 9   30  5S6M        *   0   0   GCCTAAGCTAA         *   "SA:Z:ref,29,17,0;"
6.r004    0       ref 16  30  6M14N5M     *   0   0   ATAGCTTCAGC         *
7.r003    2064    ref 29  17  6H5M        *   0   0   TAGGC               *
       "SA:Z:ref,9,+,5S6M,30,1;
8.r001    147     ref 37  30  9M          =   7   -39 CAGCGGCAT           *   NM:i:1
```

Figure 16: Example of the SAM file format. SAM is a text-based file format for storing sequence fragments aligned to a reference sequence.

The reference genome on which the mapping relies is known as the "genome sequence build" by NCBI or UCSC. NCBI reference names start by "GRC" (Genome Reference Consortium) [58] and UCSC only uses "hg" for their human genome references. On a regular basis UCSC and NCBI update the reference genome and publish a new version of the reference build. As mapping is a time-consuming task, it would be challenging to re-map all sequenced samples every time a new build becomes available. Since each build differs from the previous ones, it is important to know which version of the reference genome has been used for mapping. It is highly recommended to map all related samples to the same build version. To map our sequences, we used Burrows-Wheeler Aligner (BWA) [55]. A big advantage of BWA is that both short and long reads are supported by using different algorithms [55]. The reference genomes against which the reads are mapped has to be indexed using the following command:

```
bwa index ref.fa      (1)
```

With: `bwa` being the executable file, `index` being the first argument telling BWA the task to be executed and, `ref.fa` being the input file in the FASTA format. By running this command, BWA creates a BWT index file that will be used by other tasks to speed up the process. The next step is to find the coordinates of all reads. The alignments in suffix array

(SA) coordinates are then generated with the `aln` command. The resulting file contains ALL the alignments found by BWA. The `samse/sampe` command converts SA coordinates to chromosomal coordinates. If we assume that we have a paired-end sequence, a `bwa` command to align the sequence is

```
bwa aln ref.fa read1.fa >read1.sai     (2)
bwa aln ref.fa read2.fa >read2.sai     (3)
bwa sampe ref.fa read1.sai read2.sai read1.fa read2.fa >read.sam     (4)
```

By executing Commands 2 and 3, an index file for each pair is created. By adding these files to Command 4, bwa generates alignment in the SAM format (Figure 16). Each line in a SAM file consists of the following mandatory fields as listed in Table 3.

Table 3: SAM file - mandatory fields. Each line has 11 mandatory fields.

Col	Field	Description
1	QNAME	Query (pair) NAME
2	FLAG	bitwise FLAG
3	RNAME	Reference sequence NAME
4	POS	1-based leftmost POSition/coordinate of clipped sequence
5	MAPQ	MAPping Quality (Phred-scaled)
6	CIAGR	extended CIGAR string
7	MRNM	Mate Reference sequence NaMe ('=' if same as RNAME)
8	MPOS	1-based Mate POSistion
9	ISIZE	Inferred insert SIZE
10	SEQ	query SEQuence on the same strand as the reference
11	QUAL	query QUALity (ASCII-33 gives the Phred base quality)
12	OPT	variable OPTional fields in the format TAG:VTYPE:VALUE

2.1.4.4. Compression, sorting and indexing

One of the characteristics of NGS is its high throughput. Therefore, a key question is the amount of disk storage needed to store the data. Enough disk storage has to be ready not only for saving the sequencing files but also to store intermediate files generated during the analysis. A lack of disk storage will cause the pipeline to fail. Assuming that the human genome has about 3 billion base-pairs, a file with string of letters (A, T, C and G) will take 750 Megabytes of disk storage (assuming 2 bits per base).

In reality, sequencing a whole human genome or exome with Illumina produces a FASTQ file with millions of short reads plus their quality scores. The quality symbols are not standard characters, but bits counted as bytes (1 byte = 8 bits). Therefore, assuming that the human genome consists of three billion letters, sequencing it at an average depth of coverage of 30X would produce 90 billion letters, roughly occupying 90 gigabytes of disk space. Consequently, the size of a regular FASTQ file would be around 200 Gigabytes. We should keep in mind that this calculation is just an estimation because read length is variable and depends on the application (*e.g.*, CNV detection or SNV calling). As the consequence, the size of a SAM file would roughly be the same as both reads and quality scores are stored. Accordingly, a research center with a large number of studies involving many human genomes will require very significant and costly amounts of disk storage. We thus need to reduce somehow the size of the files and use compression algorithms to save up some disk space. The BAM file contains the binary version of the information of the

SAM file. Converting a SAM file to a BAM file saves about 25% of disk storage. [59, 60]. This conversion can be performed by SAMtools [61]:

```
samtools view –bT ref.fs read.sam -o read.bam     (5)
```

In this command, the `view` tool will print all alignments of the specified input file to the standard output and the `-b` option tells the executor that the output format should be a BAM file whose name is given after the `-o` option. "The reference file is defined by the `T` option. As soon as the BAM file gets ready, the SAM file can be removed to save some disk space. The FASTQ file can also be deleted but it is recommended to keep these files, at least till the last task of the pipeline is performed. We could also say that if the original FASTQ file can be retrieved from the BAM file, it is safe to remove it [62].

The BAM is now ready, but the reads are not sorted by genomic coordinates. Indexing the BAM file will speed up subsequent queries. Both sorting and indexing can be done with SAM tools as shown in Commands 6 and 7.

```
samtools sort -T /tmp/aln.sort  read.bam  -o   read.sorted.bam     (6)
              samtools index read.sorted.bam     (7)
```

A BAM file always has to be sorted before indexing. The `index` command will create an index file with the same name as the BAM file but with the extension `bai`.

Various procedures may use BAM files as input. As explained before, we focus on variant calling and all aspects around this topic.

2.1.4.5. Variant calling and genotyping

One of the main applications of the NGS technologies is SNV mining [63]. Reads from different individuals are generated and by comparing them with the reference genome, a list of SNVs can be produced. Conceptually, looking for SNVs is not a difficult task. Every position of the reference genome is checked and all aligned bases are compared to the reference base. A SNV will be detected if enough bases from overlapping reads aligned at the same single-base position differ from the reference one. Unfortunately, several factors can affect the detection of SNVs [64] such as:

- **The error rate associated with the sequence reads**
  Illumina sequencing is error prone and sequencing errors may lead to the detection of false positive variants, especially since Illumina sequencing errors are not random. As mentioned before, all bases of a read are assigned a Phred score. This score represents the probability of a base being a sequencing error. Distinguishing real variants from sequencing errors is important. However, in some cases, the sequencing error rate predicted by the sequencing machine is not entirely accurate. This issue can be circumvented by recalibrating the quality scores [65]. To do so, known SNPs are used, bases differing from known SNPs are expected to be for the most part sequencing errors, and a model is built to recalibrate the quality scores. The Genome Analysis Toolkit (GATK) implements this process [66]. This means that we should do the variant calling in three steps. First, call the SNVs, second do the recalibration, and finally call all the SNVs again.
- **The error rate associated with the mapping**
  The mapping tools estimate a probability of the correctness of the mapping of a read [67] at that position (*i.e.*, the probability that the read belongs to this portion

of the genomic reference and not to a homologous region). This probability depends on the length of the alignment, the number of mismatches, and the uniqueness of the aligned region on the reference genome. It is important to distinguish real SNVs from false positive variants occurring from mapping errors.

- **The reliability of the reference genome**
- **The unbalanced allelic ratio**
  In theory, we expect that for homozygous variants 100% of the aligned bases will support the alternative allele. However, this is not always true, mainly due to sequencing and mapping errors. Similarly, heterozygous variants should have 50% of the reads supporting the reference allele and 50% of the reads supporting the alternative allele. This should be true if the coverage is high enough but the deviation of the 50% ratios are expected at lower depths.

Up to this point, we considered all the issues around variant calling. By taking into account the sequencing and mapping problems, we can use SNV calling software to look for the SNVs. Three pieces of SNV calling software will be described in more details (SAMtools mpileup [61], FreeBayes [68], and GATK [65]). These SNV callers use different algorithms and make different assumptions about the reference genome and the reads. Some SNV callers are based on counting the number of reads for each allele once appropriate thresholds for the sequencing and mapping qualities have been applied. Other methods are based on more advanced statistics. These methods often perform better, especially with low coverage data, and make certain assumptions to create Bayesian models. Most of them assume diploid individuals and some even take into account Hardy-Weinberg equilibrium [69] and Linkage Disequilibrium [70] information, as well as previous information about the known SNPs present in the species and their allele frequencies. After a short explanation of these tools, they will be compared on a test dataset.

2.1.4.5.1. SAMtools' mpileup

To call and list the variants with SAMtools, the `mpileup` command has to be executed (Command 8). Input arguments are an indexed reference sequence and a BAM file. SAMtools mpileup can generate three types of files: Pileup, VCF, and BCF.

```
samtools mpileup ref.fa read.bam      (8)
```

The default output of `mpileup` is a pileup format describing the base information at each chromosomal position (Figure 17). This file is further used for SNV calling.

```
1.  seq1 272 T 24  ,.$.....,.,.,...,,,,,..^+.  <<<+;<<<<<<<<<<=<;<;7<&
2.  seq1 273 T 23  ,.....,.,.,...,,,,,..A      <<<;<<<<<<<<3<=<<<;<<+
3.  seq1 274 T 23  ,.$.....,.,.,...,,,,,...    7<7;<;<<<<<<<<<=<;<;<<6
4.  seq1 275 A 23  ,$.....,.,.,...,,,,,...^l.  <+;9*<<<<<<<<<=<<:;<<<<
5.  seq1 276 G 22  ...T,,.,.,.,...,,,,,....    33;+<<7=7<<7<&<<1;<<6<
6.  seq1 277 T 22  .....,,,,,.C.,,,,,,..G.     +7<;<<<<<<<&<=<<:;<<&<
7.  seq1 278 G 23  .....,.,.,.,...,,,,,....^k. %38*<<;<7<<7<=<<<;<<<<<
8.  seq1 279 C 23  A..T,,.,.,.,...,,,,,....    ;75&<<<<<<<<<=<<<9<<:<<
```

Figure 17: Example of Pileup file. A Pileup file describes the base-pair information at each chromosomal position. The pileup file consists of six columns: chromosome, position, reference base, the number of reads covering the site (coverage), read bases, and base qualities. In the read base column, a dot stands for a match to the reference base on the forward strand, a comma for a match on the reverse strand, "`ACGTN`" for a mismatch on the forward strand, and "acgtn" for a mismatch on the reverse strand. Most of the variant callers detect short length insertion and deletions (InDels) as well.

Another type of output is the Variant Call Format (VCF). A VCF file consists of two sections: the header section and the data section (Table 4). Lines in the header section start with "##". The header contains the meta-data information mostly in the "key=value" format. The data section is tab limited, and the name of the column is described in the line starting with "#". By default, a VCF data section contains nine mandatory fields (Table 4). The BCF format or Binary Variant Call Format is the binary version of the VCF file.

Table 4: VCF data section - mandatory fields and VCF example. A VCF is a text file used to represent different types of variations. Each file is composed of two sections headers followed by records. The header lines starts with "#" and contains the variant calling information. The records are the actual variants called by the variant caller.

Col	Field	Description
1	CHROM	Chromosome name
2	POS	1-based position. For an indel, this is the position preceding the indel.
3	ID	Variant identifier. Usually the dbSNP rsID.
4	REF	Reference sequence at POS involved in the variant. For a SNP, it is a single base.
5	ALT	Comma delimited list of alternative sequence(s).
6	QUAL	Phred-scaled probability of all samples being homozygous reference.
7	FILTER	Semicolon delimited list of filters that the variant fails to pass.
8	INFO	Semicolon delimited list of variant information.
9	Sample(s)	Individual genotype information defined by FORMAT.

```
##fileformat=VCFv4.0
##INFO=<ID=DP,Number=1,Type=Integer,Description="Total Depth">
##INFO=<ID=AF,Number=.,Type=Float,Description="Allele Frequency">
##FORMAT=<ID=GT,Number=1,Type=String,Description="Genotype">
##FORMAT=<ID=GQ,Number=1,Type=Integer,Description="Genotype Quality">
##FORMAT=<ID=DP,Number=1,Type=Integer,Description="Read Depth">
#CHROM  POS     ID          REF ALT     QUAL    FILTER  INFO                            FORMAT      NA00001
20      14370   rs6054257   G   A       29      PASS    NS=3;DP=14;AF=0.5;DB;H2         GT:GQ:DP:HQ 0|0:48:1:51,51
20      1110696 rs6040355   A   T       67      PASS    NS=2;DP=10;AF=0.333             GT:GQ:DP:HQ 1|2:21:6:23,27
20      1230237 .           T   .       47      PASS    NS=3;DP=13;AA=T                 GT:GQ:DP:HQ 0|0:54:7:56,60
```

2.1.4.5.2. FreeBayes

FreeBayes is a haplotype-based Bayesian variant caller [68]. FreeBayes requires two inputs: a FASTA reference sequence, and a BAM file sorted by reference position. It uses short-read alignments for any number of individuals from a population and a reference genome to determine the most-likely combination of genotypes for the population at each position in a reference genome. It returns a list of positions that are more likely polymorphic than monomorphic in a VCF file. It can also use an input set of variants (VCF) as a source of prior information, and a copy number variant map (BED) [71] to define non-uniform ploidy variation across the samples under analysis. By default, FreeBayes will consider variants supported by at least two observations in a single sample and at least 20% of the reads from a single sample. These settings are suitable for high depth sequencing in haploid and diploid samples, but in some cases, such as the processing of low coverage genomes, these parameters have to be adjusted. Calling variants assuming a diploid sample will be achieved by running the following command:

```
freebayes -f ref.fa read.bam>read.vcf      (9)
```

We use FreeBayes because it allows the calling of variants on many individuals from the same population. It relies on an alignment of sequencing reads to the genome, but rather than just looking at individual bases, it calls variants based on haplotypes up to the length of sequencing reads. It takes reads from a single or multiple samples and determines the

number of different haplotypes that appear in a local region. It then gives a probability for the presence of each of the haplotypes in each of the samples.

2.1.4.5.3. GATK

GATK is one of the most famous variant callers [66]. Indeed, it is well documented and well supported by its development team. It supports a broad range of sequencing technologies and species, and it is an open source project. GATK has two types of variant callers: UnifiedGenotyper and HaplotypeCaller. Concerning the GATK documentation as of GATK version 3.3, it is highly recommended to use HaplotypeCaller, regardless of the type of sequence. Variant calling can be done by the following command:

```
Java -jar GenomeAnalysisTK.jar  -T HaplotypeCaller  -R  ref.fa
-I read.bam --dbsnp dbSNP.vcf –stand_call_conf  30
–stand_emit_conf 10 – o res.vcf
```

where `-dbsnp` is used to compare dbSNP reference IDs to the variants to show whether or not a variant has been reported before. The `-stand_call_conf` value is the minimum phred-scaled confidence threshold at which variants should be called and the `-stand_emit_conf` value is the minimum phred-scaled confidence threshold at which variants should be emitted. Many more parameters are available to customize the variant calling procedure and annotate the results. GATK outputs a VCF file. There is a small difference in the sub-column of some of the VCF fields (FILTER, INFO, and Sample), which is exactly related to the type of variant caller and annotation.

The SNV calling determines which positions have at least one base differing from the bases mapped at the same position in the reference genome. The genotype of the mutations for each individual also has to be determined. Genotyping SNVs has become extremely important to researchers to understand and treat disease. Variants genotype can help researchers have a better overview of the heredity and development of an organism. A simple method to call a genotype is to count the number of times each allele is observed at a single position. If the proportion of the non-reference allele is between 20% and 80%, then the genotype is heterozygous otherwise the genotype is homozygous. This method can be used for sequencing data with the depth higher than 30X. For normal and low coverage sequencing a better estimation is needed to call the genotype. Therefore, in some methods, genotypes are calculated based on the quality scores of each read. GATK uses the quality score likelihood to calculate the genotype. Five fields are computed and represented by GATK for each SNP:

- GT: Genotype; GT represents two alleles in the format of 0 and 1, 0 for the reference allele (REF) and 1 for the alternative allele (ALT). GT can be
    - 0/0: Homozygous reference
    - 0/1: Heterozygous
    - 1/1: Homozygous alternative
- AD: Allele Depth; number of reads supporting the reported alleles.
- DP: Depth of coverage; number of filtered reads related to the reported alleles. Only the reads that pass the variant caller filter are taken into account.
- PL: Normalized Phred-scaled likelihoods of the possible genotype; the PL field contains three values related to each of the three possible genotypes.
- GQ: Quality of the assigned genotype.

Table 5: Variant caller statistics. Three variant callers are compared to see how much their results differ.

	GATK	FreeBayes	SAMtools' mpileup
Avg. Calculation time (min)	100 min	280	230
Number of Variants	25,803,326	11,042,596	15,947,800
Variants reported in dbSNP	3,306,102	3,465,013	3,037,110

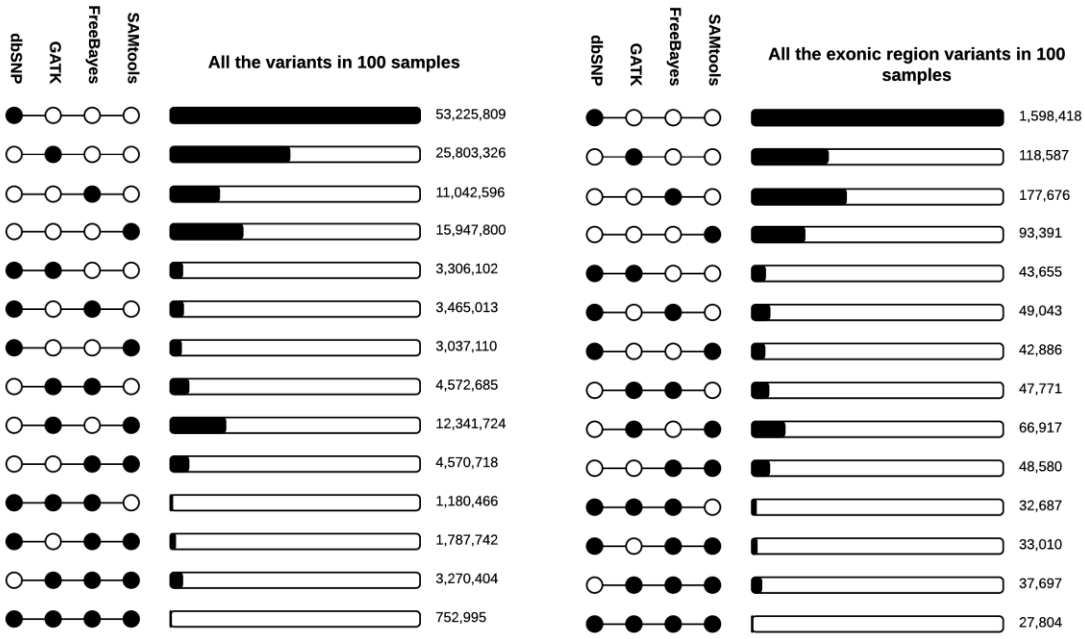


Figure 18: Comparison between variant calling tools and SNPs reported in dbSNP. Results of three variant calling tools compared to each other and data from dbSNP. The number of variants called is totally different because these tools use different algorithm to call the variants.

2.1.4.5.4. Variant caller comparison

Before explaining the variant annotation step, we run a small test to see how much these SNP callers differ from each other. As a use case, the BAM files of 100 individuals from the 1K-genome project were downloaded. These individuals are Utah residents with Northern and Western European ancestry (CEU). The CEU data set consists of 51 females and 49 males. The total disk storage needed to save these BAM files is 976.6 GB. Some statistics on variant calling on this dataset are listed in Table 5. This benchmark was run on the Tier-2 cluster at the Vlaams Supercomputer Centrum (VSC). To be fair, we assigned one node with maximum five cores and 4 GB of memory to each of the variant callers. On average, the run time per sample of GATK was 100 minutes, but the other tools took more time and thus spent more computing resources to call the variants. The minimum coverage to call a variant was set to 30 to ensure that all called variants are of good quality. As shown in Table 5, GATK called more variants than FreeBayes and Samtools mpileup. However, the number of SNPs described in dbSNP is more or less the same for all tools. We also compared the list of called variants to each other's (Figure 18). UpSet [72] is used to display the results of the comparisons to have a better view of the data compared to traditional Venn diagrams [72]. First, all the variants called were compared to each other. GATK and SAMtools mpileup have more positions in common than GATK and FreeBayes, both in genomic and exonic regions. Exonic positions were selected based on the annotation provided by ANNOVAR (see next section). Most of the intersections of

exonic positions gave the same results as those based on genomic positions, except FreeBayes called more variants in the exonic region than GATK.

These results are highly related to the sequencing quality and the mapping algorithm used to create the BAM files. Since the data was downloaded from the 1K Genome project, we are sure that the sequencing quality was approved by the project team. Different results may be obtained if another algorithm is used for the mapping but the order of magnitude for the number of called variants is expected to remain the same.

2.1.4.6. SNV annotation

By definition annotation is the act of adding a comment, explanation or any other type of information to something. Annotation is sometimes also called metadata. Annotating SNVs is the process of predicting the effect, function and type of an individual SNV. It is a challenging and complex process, and many different tools have been developed to add genomic information to the SNV annotation. Variant annotation tools can be classified into several categories:

- Gene-based annotation: The surrounding genomic elements of SNVs are used to interpret the functional effect of that SNV (*i.e.*, the effect of the variant at the protein level) [73].
- Knowledge-based annotation: In this type of annotation, more emphasis is given to the genetic variation of the protein function, protein-protein interaction, and their pathways [74].
- Functional annotation: This method mainly focuses on whether the variant is located in a known functional region that harbors genomic or epigenomic signals [75].
- Transcriptional gene regulation: These studies check the relation between a variant and a phenotype using information related to chromatin states, nucleosome positioning, or transcription factor binding [76].
- Alternative splicing: Modified splicing may have a significant effect on the phenotype that is relevant to disease or drug metabolism [77].
- Protein function: Prediction of the functional consequence of non-synonymous substitution [78].
- Variant frequencies: The frequency of a variant is often used to distinguish known polymorphisms from rare variants.
- Conservation scores: Based on multiple alignments of proteins conserved across species, conservation scores are used to predict whether a variant is pathogenic or not (SIFT and Polyphen).

Nowadays, many annotation tools and web-servers are available. Many of them rely on public databases as they allow the addition of relevant data to our SNVs. To annotate our variants, we have used two applications: ANNOVAR and eXtasy. ANNOVAR can perform gene-based, region-based, and filter-based annotation, while eXtasy focuses on the relation between phenotypes and variants.

2.1.4.6.1. ANNOVAR

Variant lists are important but often long and not easy to evaluate exhaustively. Variants are thus generally selected based on their location. Indeed, variants located within a gene are more susceptible to disrupt a protein than intronic variants. ANNOVAR takes as input

a list of variants with the following fields: chromosome, start position, end position, reference nucleotide and observed nucleotide. ANNOVAR has the ability to perform the following tasks:

- Gene-based annotation: Identify whether SNPs cause protein coding changes and amino acids that are affected.
- Region-based annotation: Identify variants in specific genomic regions.
- Filter-based annotation: Identify variants that are listed in specific databases.

By executing the following command

```
annotate_variation.pl -out  path_to_results_folder
        –build hg19  positionlist.bed
```

ANNOVAR generates four outputs: the `log` file contains the process information; the `invalid_input` file lists the positions that ANNOVAR could not annotate either because the reference build differs from the provided information or because the format of the variant is not correct; the `exonic_variant_function` file only contains that the variants located in the exonic region of the genome (Figure 19); the `variant_function` file lists of the positions from the input file (Figure 20).

synonymous SNV	SMARCA2:NM_001289396:exon4:c.G483T:p.P161P,SMARCA2:NM_139045:exon4:c.G483T:p.P161P,SMARCA2:NM_001289397:exon4:c.G483T:p.P161P,SMARCA2:NM_003070:exon4:c.G483T:p.P161P,	9	2039593	2039593	G	T
frameshift substitution	SMARCA2:NM_001289396:exon4:c.775_775delinsTG,SMARCA2:NM_139045:exon4:c.775_775delinsTG,SMARCA2:NM_001289397:exon4:c.775_775delinsTG,SMARCA2:NM_003070:exon4:c.775_775delinsTG,	9	2039885	2039885	T	TG
nonsynonymous SNV	SMARCA2:NM_001289396:exon4:c.A779G:p.N260S,SMARCA2:NM_139045:exon4:c.A779G:p.N260S,SMARCA2:NM_001289397:exon4:c.A779G:p.N260S,SMARCA2:NM_003070:exon4:c.A779G:p.N260S,	9	2039889	2039889	A	G

Figure 19: Example of ANNOVAR exonic variant annotation. The first column describes the type of exonic variant (synonymous, non-synonymous, frameshift, etc.). The second indicates the reference sequences available as gene models. The third gives chromosome. The fourth and the fifth columns give the start and end positions, while the sixth and seventh provide the reference and variant alleles.

intronic	SMARCA2	9	2035303	2035303	A	T
intronic	SMARCA2	9	2035304	2035304	C	G
intronic	SMARCA2	9	2035323	2035323	A	G
intronic	SMARCA2	9	2038766	2038766	A	T
exonic	SMARCA2	9	2039593	2039593	G	T
exonic	SMARCA2	9	2039885	2039885	T	TG
exonic	SMARCA2	9	2039889	2039889	A	G
intronic	SMARCA2	9	2039983	2039983	A	C

Figure 20: Example of ANNOVAR variant annotation. The first column describes the type of genomic region (exonic, intronic, etc.) where the variant if found. The second gives the name of the associated gene. The third column gives chromosome. The fourth and the fifth columns give the start and end positions, while the sixth and seventh provide the reference and variant alleles.

2.1.4.6.2. eXtasy

One of the main question in variant annotation is to what extent genotypes are related to the phenotype under investigation. The primary focus of eXtasy is to prioritize variant-phenotype relationships in human disorders. It ranks non-synonymous SNVs against a specific phenotype. Inputs are a VCF file and a list of Human Phenotype Ontology (HPO)

terms. HPO is an ontology of human phenotypes. For each HPO term, eXtasy precomputes a gene prioritization file. These files are then used to rank the variants. eXtasy outputs a tab limited text file containing variant information (chromosome, position, reference base and alternative base), the name of the gene containing that variant, two eXtasy scores (eXtasy_complete and eXtasy_impute) and all the features used to compute these scores. A variant with missing values in other features does not have a complete score. The complete score estimates the chance of a variant causing a disease or not. In Chapter three we will see how we used these annotation tools in our systems.

2.1.4.7. SNV filtering

SNVs are generally filtered according to their confidence score. A VCF file is similar to a matrix with the one row per variant and the sample (*e.g.*, individual) genotype listed in columns. Both variants (rows) and samples' genotypes (columns) can be filtered. Two different kinds of analyzes are commonly referred to as variant filtering. The filtered variants can either be removed from the file or annotated by adding a tag to the filter column in the VCF file. Some of the most famous and important filters applied to variants are [79]:

- Quality: SNP callers usually assign a quality score to each variant. SNVs can be filtered out if their quality is lower than a threshold.
- Number of alleles: It is possible to remove monomorphic SNVs (*i.e.*, SNVs for which a single allele is identified in the population of interest) or SNVs that are not biallelic (*i.e.*, variants for which more than two alleles are observed in the population of interest).
- Minor Allele Frequency (MAF): The frequency at which the less abundant (or minor) allele of a variant is present in a population can also be used to remove false positive variants due to sequencing errors. In a population study the following formula calculates the MAF for each biallelic variant:

$$MAF = \frac{\sum Heterozygotes \ + 2 * \sum Homozygotes\ Alternate}{\sum All\ Individuals * 2}$$

- Observed heterozygosity: Heterozygous variants are expected to have a heterozygosity close to 0.5. One common source of false positive variants with high heterozygosity rates is the presence of pseudogenes, repetitive elements or regions duplicated in the sample but unique in the reference genome. In the latter case variants result from the reads from the two copies piled in the single copy found in the reference genome. These cases cannot be avoided by filtering the reads according to the mapping quality since only one copy of the duplication is found in the reference genome. Another way to spot these false positives is to look for variants with high coverage.
- High coverage: Having regions with too many SNPs is also a sign that reads are piled at repeated regions. The variants located in such highly variable regions can also be filtered out. This analysis is usually done by counting the number of SNVs in a window of a certain length.

2.1.4.8. Interpretation

Genetic interpretation is beyond the scope of this project but to cover our pipeline some key features of this step are explained.

Before transferring the data to the clinic, the data has to be translated in a way that improves diagnostics and treatment decision-making. To interpret genetic results, clinicians should also take into account the person's medical history and his or her family history. Bioinformatics tools have been developed to predict the behavior of variations. All of them use public databases. Ideally, the more databases are used, the better. To combine information from different sources, lots of difficulties have to be tackled, such as data integrity, privacy and ethical concerns, data discovery, *etc.* Text mining algorithms and tools have been developed to ease data extraction from public sources. These sources can be divided into three groups [80]: population databases, disease databases, and sequence databases. A list of some of these databases can be found in Table 6. To use these additional resources, the correct sources have to be chosen. For example, if the reads have been aligned against the human genome reference version 19 (hg19) all the data used for interpretation has to be annotated against the same version. Also, the databases used should be regularly updated. It is highly recommended to check the results from at least two different sources to validate our results before making any decision. Typically, a genetic report must contain a summary of the genetic findings, an overview of clinical indicators and a full report about the relation between genetic findings and the clinical indicators. To achieve this, a clear workflow is needed to collect the data we need and some statistical models have to be applied to the data.

Table 6: List of public databases. Public databases can be divided into three groups: population, disease, and sequence databases.

Type	Name	Description
Population databases	Exome Aggregation consortium	Database of variants found during exome sequencing of 61,486 unrelated individuals
	Exome Variant server	Database of variants found during exome sequencing of individuals from European and African American ancestry
	1000 genome project	Database of variants from 26 population
	dbSNP	Database of short genetic variations
	dbVar	Database of structural variations
Disease databases	ClinVar	Database of assertions about the clinical significance and phenotype relationship of human variations.
	OMIM	Database of human genes and genetic conditions
	Human Gene Mutation Database	Database of variant annotations published in the literature
	Human genome variation society	The HGVS site developed a list of thousands of databases that provide variant annotations
Sequence database	NCBI Genome	Source of full human genome reference sequences
	RefSeqGene	Medically relevant gene reference sequence resource
	dbGaP	The database of Genotypes and Phenotypes
	EGA	The European Genome-Phenome Archive

There are many difficulties to interpret the clinical significance of a genetic variant. Eventually, all the evidence needs to be evaluated to find the potential role of a given variant, in a given disease or symptom, in a given patient. In general, final results will be either positive or negative. A positive result means that the laboratory found one or several changes in the patient genome indicating that a person is a carrier of a particular causative mutation. In the case of a negative test result, there are two possibilities: the laboratory did

not find any variant meaning that the person is not affected by a particular disorder, or no variant was found because of a lack of information.

2.2. Access Control List (ACL)

As seen in the previous section (2.1.4.1. Storage), the data has to be stored, and the storage can be organized in various ways. In large and well-organized data structures, samples are grouped based on characteristics, such as the owner of the sample (PI), the project, or the type of sample. In our model, samples are stored per PI since each PI works on his or her own research topic. As illustrated in Figure 13, one parent directory holds all the sequencing files. This folder contains one subfolder per PI with one folder per sample named as the sample-id. The most important questions are who has access to these folders and who can provide and manage these accesses.

2.2.1. Data access committee

A Data Access Committee (DAC) is a group of individuals within a research institute who is responsible for checking and handling the data access requests. To prevent conflict or misuse of the data, the task of the DAC is to decide whether to grant access to a particular sample or set of samples to one or several groups of researchers. The DAC must also determine how to provide access to each group and at which level. Therefore, for every data access request, the DAC must check the following:

- The purpose and duration of the project
- The proposed data usage practically and scientifically
- The existence of a valid procedure for analyzing data or a clear and well-defined research plan to analyze the data
- The confidentiality of individual research subjects must be guaranteed
- The name and affiliation of the users who will have access to the data and the coordinator of the project
- The confidentially and/or anonymity of the data
- The need for an external data set
- The need for an external collaboration

This process becomes complex in big research groups where the number of samples and researchers is large. Therefore, there must be a mechanism/protocol to make this process easier, faster and traceable.

2.2.2. Data access management

Data is the most valuable resource for every project. It must be classified to ensure that only authorized persons can access them. Various authorization levels are possible. Some people may have full control of the data, while others may have limited access only. A Data Access Management (DAM) system covers such a complex structure. It is a necessary module for all institutions no matter their size. A DAM is a piece of software that controls all the relations between users, data and their access levels (Figure 21).

NGS data is generally produced to prevent, detect, diagnose, or and treat the condition of the patients. However, NGS data can also be used for research purposes. The samples are generally tagged as diagnostic or research based on the first purpose of their generation for user access categorization (Figure 22).

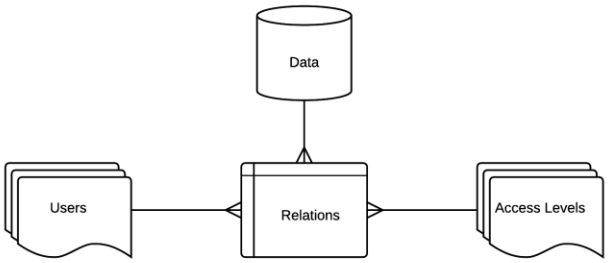


Figure 21: Main modules of data access management systems. Many-to-many relations connect data and users. The type of access levels can also be different for different users. For example, one user may have the full access to a piece of data, while another user only has read access.

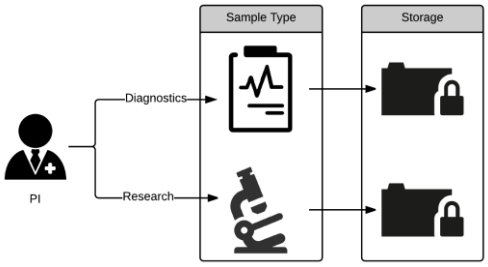


Figure 22: Organization of samples from PIs into two groups: Diagnostics and Research. At the clinical level, we often have two categories of samples: diagnostics and research. Because different types of informed consent has typically been obtained for research and diagnostic samples, it is better to store these samples separately.

Many samples are somehow linked together: they may belong to the same family (father, mother, and child) or carry similar phenotypes. In our system, samples can be grouped in a dataset (*i.e.*, a virtual object in our DAM model). Each PI may have several datasets based on his or her own projects. Samples may belong to one or more datasets. In some cases, one sample can be accessed by more than one PI (Figure 23-A).

By defining datasets, we can control user access in an easier and more organized way. Users can be assigned to one or more datasets and can ask access to other datasets and samples from other PIs. To manage user data access and reduce redundant tasks another virtual object named ‘group’ is defined. Certain data access levels can be defined for each group and users are assigned to these groups. This allows a better overview of all the data accesses without any direct link between users and samples (Figure 23-B).

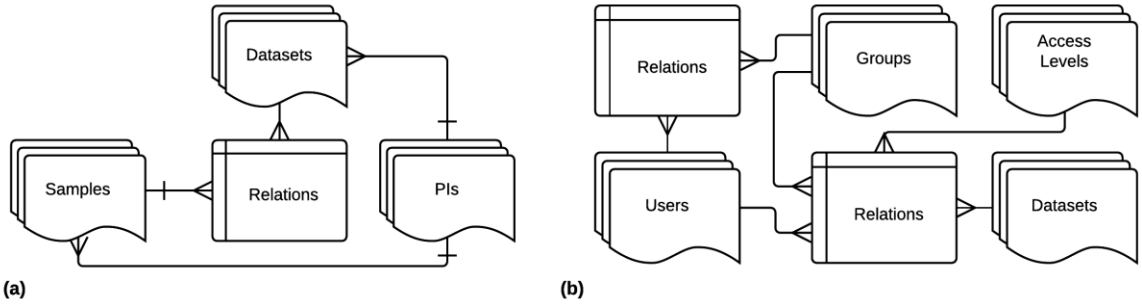


Figure 23: Relation between DAM entities. By defining a flexible data structure, we have the possibility to adapt the database easily to new requirements, while covering all aspects of the data flow. (a) Relation between datasets, PIs and samples. On the left side, each PI controls several samples and at the same time each sample can be part of one or more datasets. (b) Relation between datasets, users, and groups. The relation between entities on the right side allows us to easily manage the user access levels across samples and datasets.

The best solution to store and arrange these relations and links is a rich interface software package. The data is stored in a relational database model. Microsoft SQL Server (MSSQL) is used as the engine of our relational database management system (RDBMS) while Java is used for developing the interface and a multitier architecture is used for the programming methodology. In general, multitier application consists of three tiers: data, business, and presentation (Figure 24).

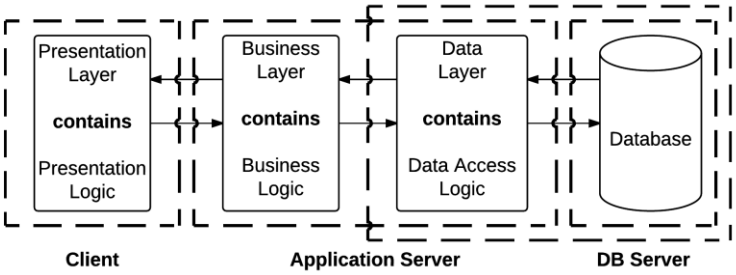


Figure 24: Multi-tier application architecture. A three-tier application separates the data and the presentation layers and link them to each other through the business layer. This model makes it easier for developers to update the application by limiting the impact of most changes to a single layer.

2.2.2.1 Data layer

The data layer contains data, information and eventually all functions necessary either for storing data in the database or fetching data from the database and sending it to the business layer. In MSSQL, several objects allow organizing data professionally:

- Table: This the place where the data is stored in the format of columns and rows.
- View: This is a virtual table allowing us to join several tables or apply filters on the data to retrieve it faster.
- Procedure: This is a stored program to which parameters can be passed to retrieve data or run some calculation on the data.
- Function: This object is similar to the procedure object; it can be used to retrieve data or perform a calculation. The main advantage of a function is that it can be called in another SQL syntax, and it uses row values as the input parameter.
- Index: Indexes are used to speed up the queries. One of the main advantages of MSSQL is that there is no limitation on defining indexes on top of each table so that indexes can be defined for Views.
- Trigger: This is a special kind of procedure that is automatically executed when an event occurs in a table (insert, update and delete).

Our database has been designed by taking into account all DAM objects, as well as their relations (Figure 25). At the time of writing this thesis, this model is also used to manage the data access level of our data-sharing platform "*NGS-Logistics*" (see Chapter 3). Of course, such a model is never complete. During the development and maintenance of any system, its database schema can be updated. Therefore, our database was designed in a way that the costs of adaptation will be reduced. By defining the data access logics in the data layer, the database can be adapted without making many changes in the business and presentation layers.

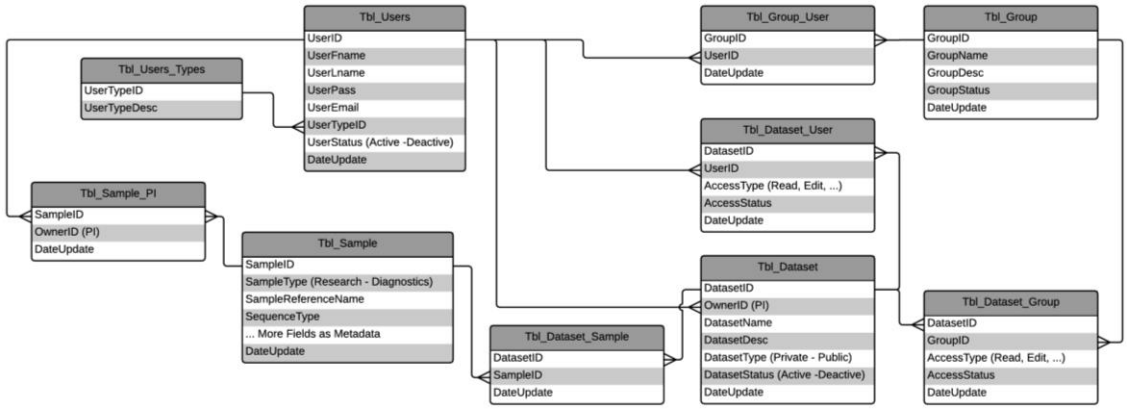


Figure 25: Database entity-relationship model for data access management. In our application, the data access management entities are Users, Samples, Groups, and Datasets. Users are categorized by their types: System admins, Local admins, PIs, and Users. Each type of entity plays a different role. PI users are the controlers of data: each PI controls one or more samples. Based on the characteristics of the samples, samples are grouped by datasets. Each dataset is assigned to one or more users. To handle permissions more efficiently, we add the group entity to the system. In this way, users with the same access can be assigned to the relevant group so as to minimize the effort need to manage user access.

2.2.2.2 Business layer

The business layer is the middle part of any *n*-tier architecture application. All communication between the data and presentation layers passes through this layer. It contains all the necessary code allowing the presentation layer to connect and communicate to and with the data layer. In general, this layer carries two types of functions: the communication between the presentation and data layers and the processing of the request from the presentation layer. Normally, the processing of requests contains some general rules and roles that need to be checked to control user activity.

- Java:

```
1.  Public static Connection CreatConnSec(){
2.  Private static String dbIpAddress = "Server name";
3.  private static String dbName="Database name";
4.  private static String dbUsername = "User name";
5.  private static String dbPass = "User Password";
6.  DriverManager.registerDriver (new SQLServerDriver());
7.  FCon  = DriverManager.getConnection( "jdbc:sqlserver://" + dbIpAddress + ":1433;
        DatabaseName=" + dbName, dbUsername, dbPass);
8.      return FCon;   }
9.    public static void CheckNewReq() {
10.     con = StaticValues.CreatConnSec();
11.     Statement stm = con.createStatement();
12.     ResultSet rslt = stm.executeQuery("Select * from Table1");   }
```

- VB.Net

```
1.  <connectionStrings>
2.      <add name="MyDbConn" connectionString="Data Source=servername;Initial Catalo
    g=dbname;Persist Security Info=True;User ID=username ;Password=userpass" provide
    rName="System.Data.SqlClient" />
3.  </connectionStrings>
4.  Private Function GetData() As DataTable
5.      Dim CnnStr As String = System.Configuration.ConfigurationManager.ConnectionS
                               trings("MyDbConn").ConnectionString
6.      Dim adapter As New SqlDataAdapter("SELECT * from Table1", CnnStr)
7.      Dim data As New DataTable()
8.         adapter.Fill(data)
9.      Return data
10. End Function
```

It may also contain calculation and data processing that do not require any communication with the data layer. Various programming languages can implement this layer differently, but the behavior of the objects is common to all. Objects can be static or variable, public or private, depending on where and how they are used. For example, a database can be assigned the connection manager function (*i.e.*, it will create a secure connection between application and database engine). Two examples of this function are presented in the code fragments denoted as “Java” and “VB.Net”. Both connect to the MSSQL server, one is written in Java, the other in VB.Net, as both programming languages were used in our development (Java to develop backend tools and VB.Net for our web interface applications). The nature of web-based and stand-alone applications is different, but this shows how two different programming languages can be similar.

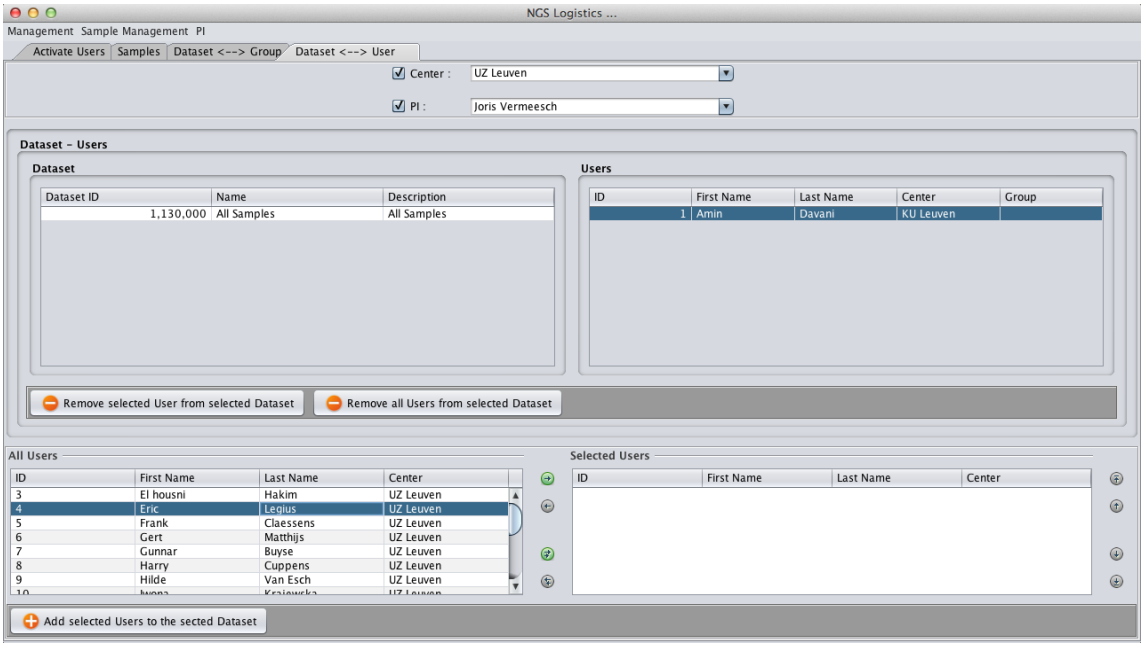


Figure 26: Screenshot of the NGS-Logistics ACL management tool. A user-friendly interface is need for the ACL management tool to hide the complexity of managing user access levels.

2.2.2.3 Presentation layer

This layer is the one with which users interact by providing input parameters, changing values, or removing conditions by clicking or typing commands. User requests are first transferred to the business layer. If data has to be retrieved or added, requests are transferred to the data layer. The complexity of interactions between layers is transparent to the users, except when the processing time causes the completion of an action to be delayed. To hide this complexity, developers do their best to create and develop a user-friendly interface (*i.e.*, an interface that is as simple as possible). The interaction between user and application must be simple, and at the same time comprehensible. Indeed, users always like to get their job done while providing a minimum amount of input. Robust applications, besides the quality of performing complex tasks, also provide user-friendly interfaces. Several integrated development environments (IDEs) exist to help designers to create their application, test it, debug it and find errors. In this thesis, we mostly used NetBeans to develop a Java based application and Visual Studio .Net for our web applications. These IDEs support different operating systems so that developers can test

their application on different platforms to make sure that users will not experience any issue with the interface and application. We thus developed an easy to use interface to help system administrators provide access to users and control their activity (Figure 26).

2.3. Collaboration

Initially, samples are investigated for one type of question. In practice, however, samples can be reused to address a variety of questions. For example, in population studies having access to different samples allows drawing a better and more accurate picture of the population. Approximately 99.9% percent of the 3 billion bases of the human genome is shared between individuals. Still this 0.1% difference is responsible for a huge variability between individuals, such as hair, eye, skin color and many other characteristics that make individuals unique and different from other human beings. In the case of a patient with a genetic disorder, comparing his or her genome to the one of a healthy person results in about 20,000 differences in coding regions. This is a real challenge for researchers to recognize and find the single or group of nucleotide changes related to the patient's condition. Using the genome of related family members (such as parents and (un)affected siblings) reduces the number of variants to consider to thousands of variants. Researchers usually gather data from as many individuals with the same condition as possible to find the culprit variant and collect confirmatory evidence for their findings. In the case of rare diseases, patients are spread all around the world, and it might not be easy to identify and gather all affected individuals. Also, other groups may have studied individuals with the same condition. Therefore, it is obvious that to do our research we need access to the data. But we first need to answer two questions: how to find the data we need? And how to get access to this data? The data that we need can be obtained either by sequencing more individuals with the same condition or by reading literature and searching public databases to identify groups working on the same condition.

2.3.1. Locating and getting access to the data

Finding patients with the same condition as the one we are interested in might be tedious and time consuming. Besides, there is no guarantee that a match will be found. Another option is to search public databases for groups who may have the type of data that we need. As mentioned before in the interpretation section comparing our results to public databases can filter out many common variants, regarded as non-causative in the case of rare diseases. After finding the data, we must apply to have access to it, and we may face various issues depending on the type of data. If the data that we need is available in our center, then we simply have to ask our colleagues to provide us the necessary information and apply to DAC for getting access to the data. Getting access to data sequenced outside of our organization is more complex and time-consuming. In general, samples sequenced outside our organizations can be classified into two categories: those identified thanks to our collaborations and those stored in one of the public repositories, such as dbGap and EGA.

Samples are often deposited to these secured repositories because journals require it upon publication and project funding agencies encourage it. Both dbGap and EGA have well-defined data access policy to provide access to the data as summarized in Figure 27:

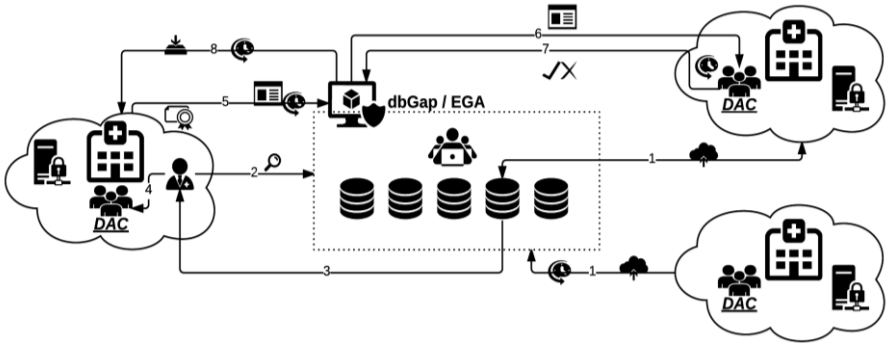


Figure 27: Applying for data through dbGaP/EGA process. Multiple actions, some of them involving bureaucratic decisions, must be taken to obtain access to the a dataset. As a result, the procedure is cumbersome and time consuming.

1. Samples belong to a study with their full annotation uploaded to the data center.
2. Researchers from other centers can search the database to see which kind of data is available.
3. Researchers find the most relevant dataset.
4. To apply for the dataset, researchers must contact the local DAC. The reason behind this is that by definition dbGaP and EGA ask each institute to provide a safe and clear protocol to protect data. Therefore, a global user handling all the communications must be created for each institute. Several forms and agreements must be prepared and signed by all parties. These documents contain the aim of the project, the name and affiliation of the users who will work with the data, the contact information of the person(s) responsible for storage and safety of the data in IT and finally an agreement about the data itself.
5. The local DAC submits the application to the dbGap/EGA for a first control.
6. The request is then sent to the DAC of the organization that controls the data since the decision of granting access or not belongs to the data controlling DAC.
7. The data controlling DAC checks the request and returns its decision back to dbGap/EGA.
8. If the application is accepted, the data becomes available for download.

Creating a user in dbGaP and EGA can take up to a month. The whole process of finding a dataset and getting access to it can also take 5 to 6 weeks. When downloading a dataset, it is important to prepare and provide enough disk space and computing power to use the data. Moreover, the data has to be kept safely, a yearly report on the usage of the dataset has to be provided and the application has to be renewed annually or closed. Despite this tedious process, there is no guarantee that the data we applied for and downloaded will be useful for our study. We thus need a better way to get the information we need. We need a solution that at least helps us to apply for and download a dataset that we are sure is highly related to and useful for our study.

2.3.2. Data access solution

Since researchers often lack data, and since locating and getting access to data is a difficult and time-consuming task, we had to find a way to help researchers find the right data in a shorter amount of time. Most of the times access to the entire dataset is not needed to gain relevant information. For example, by answering a question such “What is the frequency of mutation X in the Y population?”, we can already answer many useful questions. For instance, if when looking for the cause of a certain phenotype we have several candidate

mutations, finding the frequency of appearance of each candidate mutation helps to filter our candidate list.

A unique solution for this kind of problems is data sharing. In research by definition, data sharing is the act of making data available to another investigator. There are three mechanisms to make data accessible: Centralized, Federated, and P2P (Figure 28).

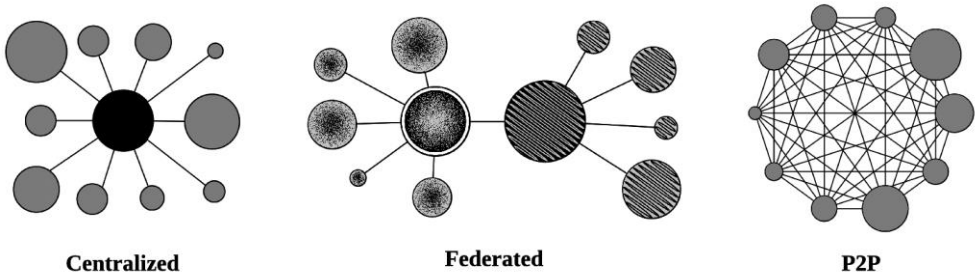


Figure 28: Data sharing mechanisms. Databases such as dbSNP and dbGaP are centralized databases, where all the data is transfered to and stored in one location. In a federated model, the data does not leave the place where it is generated, but there is a central database to store partial results and other basic information. In the peer-to-peer model, all interactions are managed by the peers directly and there is no need for any central database.

2.3.2.1 Centralized databases

A Centralized DataBase (CDB) is a database that is located, stored and maintained in a single location. In a centralized model, each node creates a copy of the data that needs to be shared and transfers it to a location outside of their network. dbGaP, EGA, 1K genome, NCBI, dbSNP, etc., are examples of CDBs. Advantages and disadvantages of a CDB model are listed below:

Advantages	Disadvantages
• Data integrity (increased) • Data redundancy (minimized) • Easier to use by end user • Easier database administration • More cost effective • Easy data update • Easy development	• In case of data loss, recovering data is difficult • Data must be converted into a single format • Bottlenecks can occur as a result of high network traffic • Data ownership is with the source with shared data stewardship with the centralized data warehouse entity.

In the centralized model, the owner of the data does not have full control over the data after upload. This is of course highly related to the agreement between the center and the database organizers. Centers who have data related to patients usually prefer to not ship their data outside of their network. CDBs are divided into two groups: public and registered access. In public models, such as 1K Genome and dbSNP, users have access to the data without any condition and may not even need registration. Databases such as dbGaP and EGA protect data from untrusted access, and users must register and have to fill in the necessary applications as explained above. This model is thus not suitable to provide information and data to other researchers.

2.3.2.2 Federated databases

We can classify databases into two groups: (1) logically and physically centralized; (2) logically centralized and physically decentralized. A Federated DataBase System (FDBS) is a system in which several databases physically located in different places functionally

work as a single entity. A federated dataset is a collection of data that can be viewed and manipulated as if they were a single resource. Each component of the system (node) keeps its data and has its own autonomy and integrity. A federation model consists of n nodes ($n \geq 2$) and a single federal interface interconnecting all nodes via a computer network (Figure 29).

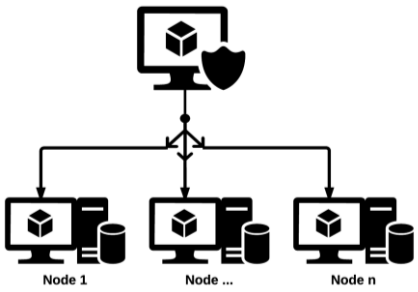


Figure 29: Federated system and its nodes. In a federated database, the data flow is controled by a central database. Users does not have a direct access to the data and instead they submit queries through the central node.

A FDBS has several advantages compared to other systems:

- Transparency: This ability provides an insulating layer to the application, relieving both the application and its developer from having to know all the various technology normally required to access data from different vendors and different data source types.
- Heterogeneity: In practice, users can manipulate a database without having to worry about the distribution of data, the diversity of local systems and database languages and without modifying the original sources. A FDBS system creates a virtual unified schema allowing users and developers to query the data in a single format.
- Extensibility: A Federated system can be extended to almost any data source.
- Autonomy: Data sources and nodes can remain autonomous from the influence of other parties.
- Performance: Queries are optimized for a distributed environment by understanding the capabilities of the back-end sources of each node to take advantage of processor capability.

Like other mechanisms FDBS models have their own disadvantages as well:

- Requires development and maintenance of multiple data sharing policies.
- Data linked every time a dataset is generated.
- Investment and support of intermediary interface by each of the participants.

It is impossible to hide the complexity of the structure to the users without a unique interface. An interface allows users and clients to communicate with each node without taking into account the type and configuration of that particular node. The act and technique of managing complexity in computer science is called abstraction. This complexity is the result of heterogeneity. Diversity in databases may be due to several factors, such as differences in structure, semantics of data logic and query languages. A concrete schema, known as Five Level Schema Architecture [81] is used to handle the heterogeneity (Figure 30). It contains:

1. **Local schema**: This is the conceptual schema of the nodes. Different local schemas may be expressed in different data models since they might be designed based on the various DBMS systems and architectures.

2. **Component schema**: This is made by translating the local schemas into a model called the common data model of the FDBS.
3. **Export schema**: Not all the data of a node may be available to the FDBS and its users. Therefore, each node can specify the shareable object to the other members of the system. This schema also contains the list of processes and operations that can be performed and submitted from other nodes. The export schema is managed and controlled by the local administrator of each node.
4. **Federated schema**: This is the integration of several export schemas. FDBS is responsible for updating, monitoring and managing the federated schema. Normally to reduce human errors, programmers develop modules to automate the schema update.
5. **External schema**: For each node, this is the place where queries submitted to the system are transferred and processed. Data produced at different nodes is merged into a single dataset and returned back to the federated system.

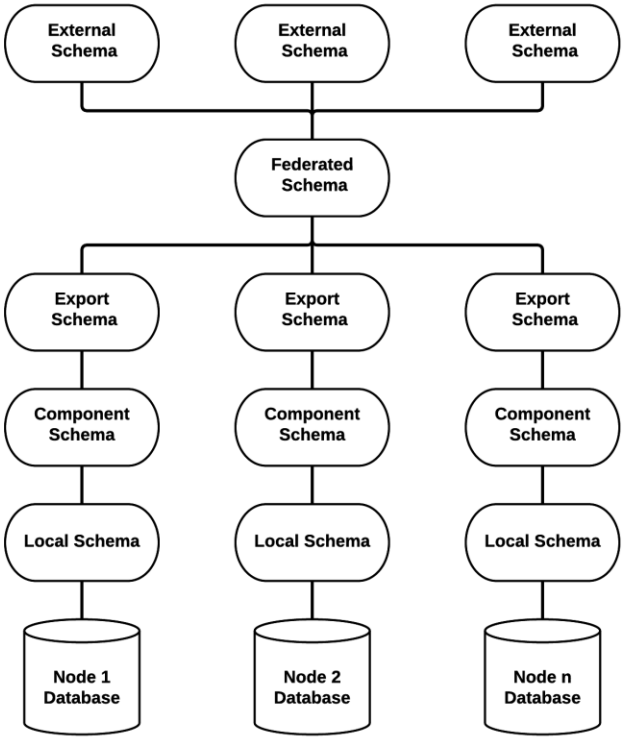


Figure 30: Five-level schema architecture of an FDBS. Every node has a local schema that will be translated into a predefined and standard schema called the component schema. The export schema is responsible for controling which data is shared with users. The federated schema is obtained through the aggregation of all the export schema. On top of these layers, the external schema manage the views that users have of the data.

By defining and integrating all these schemas, the local systems must be able to continue their operations while participating in the federated system without any issue and with a minimum amount of changes. However, having a successful solution should not be at the cost of reducing the security and user access level has to be controlled. Each of the local partners has its own security policy describing access control information. These policies are stored in the local database and transferred to the federated access control system so that the system can have the full control over user activities. An example of such a system has been described in Section 2.2. Access Control List (ACL).

The characteristics of a federated system (transparency, heterogeneity, extensibility, autonomy and performance) make it one of the best structures for genomics data analysis. We thus decided to build a structure based on the federated model to allow researchers to query NGS data and implemented *NGS-Logistics*, which is the unique example of this kind of platform at the time of writing this thesis. In the next chapter, we briefly explain how the system works.

2.3.2.3. Peer to Peer (P2P)

Both centralized and federated models need a central module to control the activity of the users and other partners. This means that some time has to be spent to check the system (centralized/federated) in parallel with the in-house systems. Any change in one center may affect the other systems and keeping the whole structure up to date is difficult. We therefore need a solution to reduce the dependency of components to each other structures (*i.e.*, we need to somehow connect the nodes without a need for a central database. This type of system is known as Peer to Peer (P2P). A P2P network is created when two or more PCs, servers or networks are connected directly to each other and when their resources are shared. The main difference between a P2P system and a centralized or federated model is that the main node does not exist. In this model, each node defines its own rules and roles. The only thing that members of the network should agree on is the protocol that allows them to communicate with each other. Currently P2P networks are mainly used as a medium for downloading and sharing files. P2P networks have grown to a tremendous level with many networks having more than millions of users to share software's, music files, videos, programs, *etc.* In a P2P model, everyone becomes a server as the server role is distributed to users. Instead of only taking files, peer to peer makes it a two-way street. One of the most famous P2P network is BitTorrent. Although BitTorrent is notorious for its use in illicit file sharing, the protocol can also be used legitimately to serve our community better and faster. BitTorrent is an open protocol whereby users create a "descriptor" file (a .torrent file) containing information about the download, but not the actual download. A tracker is needed to store these descriptors, along with information on who currently holds that file. Therefore, a Distributed Hash Table (DHT) is implemented to save the information on the content of each node. A DHT is made of key-value pairs that provide a lookup service for decentralized distributed systems. Nodes are responsible for updating and managing the key-value pairs. In this way, the amount of necessary changes at the time of updating a module in one of the nodes is reduced. DHTs remove the need for a central coordination system, the network becomes reliable even when nodes are joining, leaving or failing. By taking into account the characteristics of DHTs, it is obvious that P2P networks can form an infrastructure that can be used to build more complex systems. So far no one uses this method in genomics and implementing a network based on these methods constitutes a future challenge. We believe that by connecting and sharing data directly between hospitals and research centers, we could bring the questions around health to another level. Of course, security and privacy are the real challenges in this kind of solutions. However, it is clear that by giving the possibility to each center to organize their own data and control who can have access to what we will have more flexibility.

2.4. Complexity

Identifying the relation between the properties of software programs and the difficulty of performing programming tasks is always challenging. Measuring these relationships is important since a good estimation reduces both programming and maintenance costs. In general software, can be categorized as simple or complex systems. Simple software has typically been developed to cover one task only without the need for heavy computational resources or a large amount of maintenance. For example, a simple calculator can be labeled as a simple system. The complexity of a system is based on the logical structure of a program, the heterogeneity of data, the number of events and the number of inputs and outputs from and to other systems [82]. As software design complexity is difficult to assess, we need complexity metrics and measurements algorithms. Most of these metrics focus on the calculation of complexity of algorithms, such as the Halstead software difficulty metric or the Mc Cabe cyclometric complexity metrics [83]. These algorithms are code-based meaning that the calculation of the complexity is done once a code has been written. If the complexity is too high, it has to be reduced to speed up the calculation.

2.4.1. Functional complexity

Another important aspect of calculating the complexity of software is based on the measurements of factors that affect the cost of developing and maintaining the software. In software engineering, measuring the size of the software is known as functional complexity. Several factors must be checked to determine software size [84] (Figure 31).

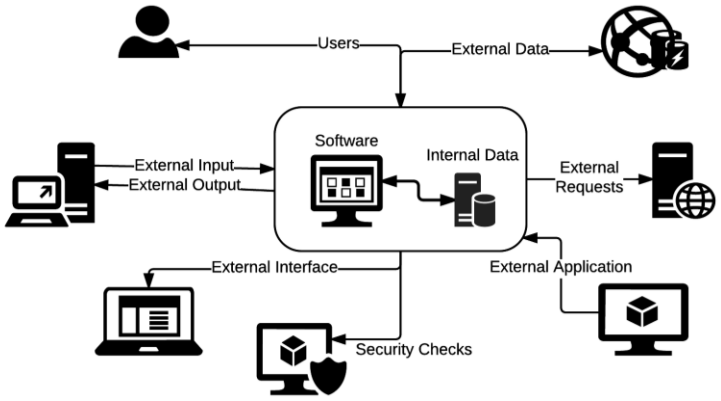


Figure 31: Functional complexity factors. Several factors can affect the complexity of a solution. The effect of each factor should be analyzed separately and across the whole system to limit the risk that it becomes a bottleneck for the functionality of the whole system.

Each of these factors in the system is given weight (between 1 and 10) according to its complexity. Scores are named *raw function points*. Every system is described using the following different characteristics:

- Data communications
- Distributed processing
- Performance objectives
- End user efficiency
- Transaction rate
- Online data entry
- Online update
- Complex processing logic
- Re-usability
- Installation ease
- Operational ease
- Multiple sites

These characteristics factors are then rated from 0 to 5:

No influence	Incidental	Moderate	Average	Significant	Essential
0	1	2	3	4	5

All ratings are then summed up as *N*, the value of *N* ranging from 0 to 70 (14*5). *N* is then used to calculate the Complexity Adjustment Factor (CAF) with the following formula:

$$CAF = 0.65 + 0.01N$$

and

$$Functional\ Complexity = CAF \times RawFP$$

The Functional Complexity (FC) score can be used in various metrics, such as

- Cost = $ / FC
- Quality = Errors / FC
- Productivity = FC / Person-month

It is now clear how much functional complexity can affect project costs, quality of material and programmers' ability to understand and tackle the problems. In general, functional complexity is divided into three categories: problem complexity, system design complexity, and procedural complexity [85, 86].

2.4.1.1. Problem complexity

First, the relation between problems, algorithms and programs should be explained (Figure 32). By definition, a problem is anything we are trying to solve computationally. Problems typically specify an input and the desired output, which will both be finite for most models of computation. Any statement of the form, "Given X: compute, evaluate, find, determine, decide, etc." is a problem. Therefore, an algorithm is a step-by-step procedure for calculation. To solve a problem, we need to break it into steps that can be coded. Consequently, a program is an implementation of an algorithm. As explained previously, the algorithm complexity is defined by analyzing an algorithm. To see how complex a problem is the algorithm has to be evaluated in two ways: first, how much the algorithm can be improved by minimizing the algorithm complexity (upper-bound) and second, a threshold must be defined to prevent the condition of an algorithm becoming worse (lower-bound). Accordingly, the upper bound is the best algorithm solution that has been found for a problem; and the lower bound is the best solution that is theoretically possible.

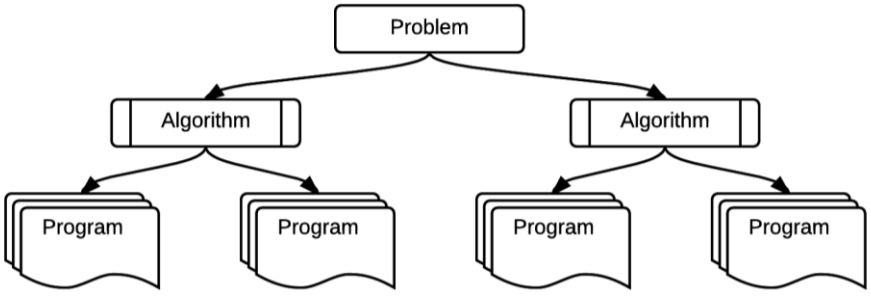


Figure 32: Relation between problems, algorithms, and programs. Here, a problem is a question we try to solve by running a computational procedure. A procedure consists of one or more algorithms. A program is then a particular implementation of an algorithm.

To illustrate this, let us take the example of what we have done in *NGS-Logistics*. In *NGS-Logistics*, one of our goals is to provide an environment for users to submit their queries and find the frequency of a certain mutation among the population. The problem is "Find a mutation X at the coordinate of Chromosome:Position". To do so we have two options (algorithm):

1. Ask each center to prepare the list of its mutations with their frequencies.
2. Ask each center to give us the chance to query directly their BAM files.

Each of these algorithms has its own advantages and disadvantages. By asking each center to prepare the list of their variants, we only need to query their dataset to find the position of interest and return the results back. This process is fast but has some limitations.

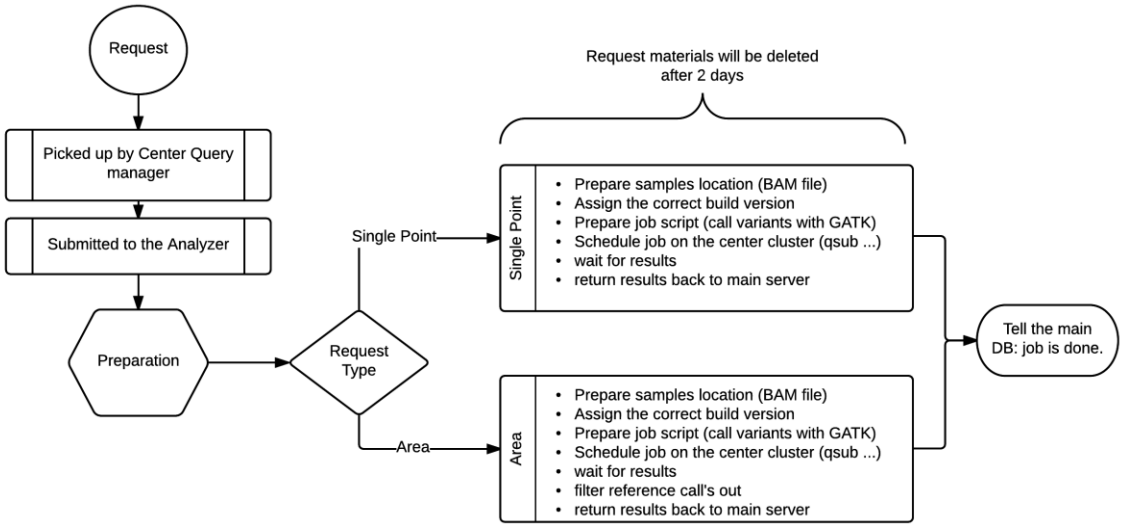


Figure 33: NGS-Logistics: variants calling workflow. The query manager is responsible for the requests submitted by the users. Different tasks will be executed by the submitted request. Once the result are ready, the query manager transfers the partial results to the main database and informs the main system that the task is completed.

First, we do not know which tools have been used by each center to call the variants. As already discussed, variant callers can make different decisions and return different results. Another limitation is that users are not given the possibility to define their own parameters and thresholds for calling the variants. Finally, when looking only at variant positions, it is impossible to estimate a real variant frequency across all individuals as only variant positions are taken into account. The other option is to ask each center to allow us to query directly their BAM files. The big disadvantage of this algorithm is the time needed to get the results. However, the queries can be standardized among all centers, and users have the option to define the parameters to call the variants and a real frequency, taking into account reference genotypes, can be computed (Figure 33).

2.4.1.2. System design complexity

Two major reasons make the design of the software complicated:

- Data richness: many interconnections and relationships.
- Hardware infrastructure: different types of servers and methodologies.

These two reasons are known as data complexity and infrastructure complexity respectively. The software engineer should consider both data and infrastructure complexity and should understand their implications before proceeding into the

construction stage of software development. Any data structure is designed to organize data for a specific purpose. Thus, developers choose the best format to organize and store the data depending on the type of application. The format could be a static array, a file, or a row and a table in relational databases. The data structure used by an algorithm can significantly affect the algorithm's performance. A data structure can be considered complex if it is:

- Multi-format: The data is represented in different formats (database, text, image).
- Multi-structure: The data is structured diversely (relational structure, XML, JASON)
- Multi-source: The data originates from different sources (distributed databases).
- Multi-modal: The data is described through several channels (data expressed in different scales).
- Multi-version: The data changes through time in term of definition or value.

The design complexity depends on the infrastructure complexity that determines how each component of a system talks to and works with each other. The effective management of an infrastructure requires an integrated approach for monitoring and administrating hardware and software elements. Organizations are often challenged to achieve these requirements because of infrastructure complexity. Most modern IT infrastructures are composed of five main elements: computer hardware, operating system, software applications, data management and storage, networking, and internet. These components highly rely on each other and failure of one of them can affect the whole system. Effective IT management is thus critical to achieving a high level of service. The more heterogeneous a server environment is, the more complex the required administrative process. As infrastructure complexity is hard to avoid, it is important that organizations be aware of the problem and prevent the consequences of an overly complex structure. To establish and maintain control over the development and maintenance of a software product, software developers must evaluate software requirements, such as disk space, processing power, memory, and tools (database, web servers). In conclusion, system design complexity addresses the mapping of a problem into a given representation by taking into account both data and infrastructure conditions. It is beyond the scope of this thesis to explicitly define how the criteria can be measured in practice. However, two systems (eXtasy and *GALAHAD*) are compared to illustrate how data and infrastructure can affect the complexity of the design of a system. Both tools were developed in a way that allows researchers to analyze their data by running a standalone application (i.e., users can download the tools and run them on their personal computers or institute servers). To provide a better service, a web-based version was offered for both of them. Therefore, users without extensive knowledge of information technologies can also run the tools and use its results for their research. In continue we are going to see how we implement these two systems.

As explained before (2.1.4.6.2. eXtasy), eXtasy is used to rank non-synonymous mutations against a specific phenotype or set of phenotypes. Two inputs have to be provided: a list of variations in VCF format and a list of phenotypes (comma separated list). Phenotypes are based on the Human Phenotype Ontology (HPO). eXtasy runs the process and creates a tab separated file containing all the scores related to the study. Users can use this result file to check which variant is highly correlated to the phenotype of interest.

The overall software architecture and hardware infrastructure of the eXtasy web application is illustrated in Figure 34. The steps are

1. Users browse the web page, upload a VCF file and define HPO term(s).
2. After submitting the request is transferred to the web server.

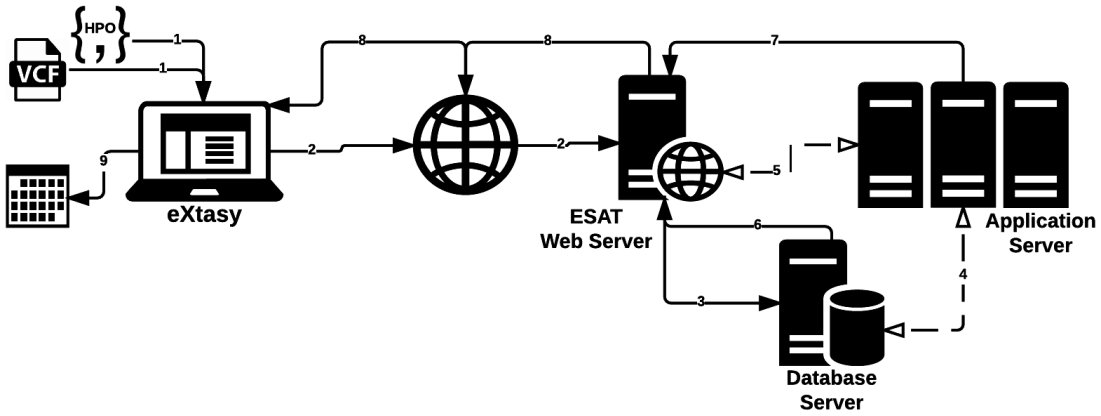


Figure 34: eXtasy web application structure. The user submits a VCF file and the phenotype(s) of interest to the system. The web server transfers the request to the application server, where the actual process takes place. Once the results are ready, the status of the request is changed in the database. Once this change is registered, the web server transfers the result back to the user.

3. The request is stored in the database (Request status = 0).
4. The eXtasy application checks the database for new requests and it then
    a. Picks the new request (Request status = 1).
    b. Keeps track of the status of the request.
    c. Runs the request (Request status = 2).
    d. Changes the request status once it gets complete (Request status = 3).
    e. Changes the request status if any error occurred (Request status = 4).
5. Users files are stored on the web server and eXtasy runs the analysis on the files without moving them.
6. If the request finished successfully, the database server tells the web application that the request is ready.
7. Results are transferred to the web server.
8. The eXtasy web application notifies the user of the results.
9. Users can download the result as a tab separated text file.

In eXtasy, users just upload files and wait for the server response. There is no interface for visualizing the results. The requested information is stored in the database to know how many researchers used the system. By the time of writing this thesis, 2,563 unique requests have been submitted to the eXtasy web application. Another and more important reason to store the requests in a database is that since running eXtasy is a time-consuming process, we could not provide this service as a standard web service. The duration of the process is strongly dependent on the size of the VCF file (number of variants on the VCF) and the number of phenotypes of interest, as well as to the workload of the application server. For web servers and web applications, a session is a unique connection between the client and the server. Each session has a unique identifier that web developers use to track user activity. However, sessions are limited in time. After a while sessions are killed by web servers for security reasons. Because the eXtasy execution process can be longer than the

session time, the session will sometimes get killed by the web server before users receive their results. In such a situation, we cannot use session identifiers and instead we need to track requests by storing them in a database with a unique identifier and keeping track of their status.

GALAHAD implements an algorithm that uses gene expression data and integrate it with protein interactions networks to find possible targets of a drug or chemical compound [87]. Drug development is a high-risk and expensive business. Attrition rates have risen sharply, especially in late-phase clinical trials. With safety and efficacy being the main bottlenecks, a better knowledge of candidate drug's mode of action and its off-target effects could be of substantial value to drug development. DNA microarray technology enables a genome-wide analysis of the transcriptional response to a compound treatment, and thus can provide valuable information for identifying compound-protein interactions and resulting effects prior to clinical trials. In addition, this information may also be useful for already marketed drugs, in the light of drug repositioning [88]. *GALAHAD* provides

- Preprocessing of gene expression data from Affymetrix arrays.
- Quality assessment and exploratory analysis.
- Differential expression analysis.
- Genome-wide drug target prioritization.
- Prediction of pathways involved in the drug's mode of effect.
- Identification of associated disease phenotypes enabling side effect prediction and drug repositioning.

Users can define several parameters, such as the type of expression array. Then they need to upload two zipped files containing raw Affymetrix data for both control and treatment (= drug response) samples. Users can also choose the type of analysis to run on their data.

The basic aim of this infrastructure (Figure 35) is to increase the effectiveness and efficiency of the tools and report the results in the most convenient way. In *GALAHAD* and eXtasy both web server and database server are highly active and responsive to user requests. Interactions between components of the systems are:

1. The user defines the expression data, array type and the list of analyses to run on the data; and control and treated samples files are uploaded to the web server.
2. The request files and parameters are transferred to web server.
3. Each request is stored in the database with a status equal to 0.
4. The database triggers a function to run the *GALAHAD* analysis.
5. The data is transferred to the application server where the actual job is scheduled and run. The algorithm is coded in R, and a small package connects the R environment to the database and reads and writes data directly to the database.
6. Results are written to the database. Since different analyzes can be run on the data, a complex data structure is needed to support the system (Figure 36). Results are stored in two formats: records and images. Since visualizing some of the results online slowed down the web application, images are computed beforehand and returned to users. Other results are stored in tables.
7. After transferring all the results (records and images) to the database and web server, the database triggers another function to let the web application know that the request has been completed successfully.
8. Requests results are returned back to the user.

9. Users connect to the result page where all the results can be visualized in an interactive way.
10. Users can filter, export, and re-run some parts of the analysis based on new parameters.
11. New user requests are transferred to the web server through the interactive page.
12. The web server communicates with the data layer through the business layer of the application and gives the right feedback to the request.

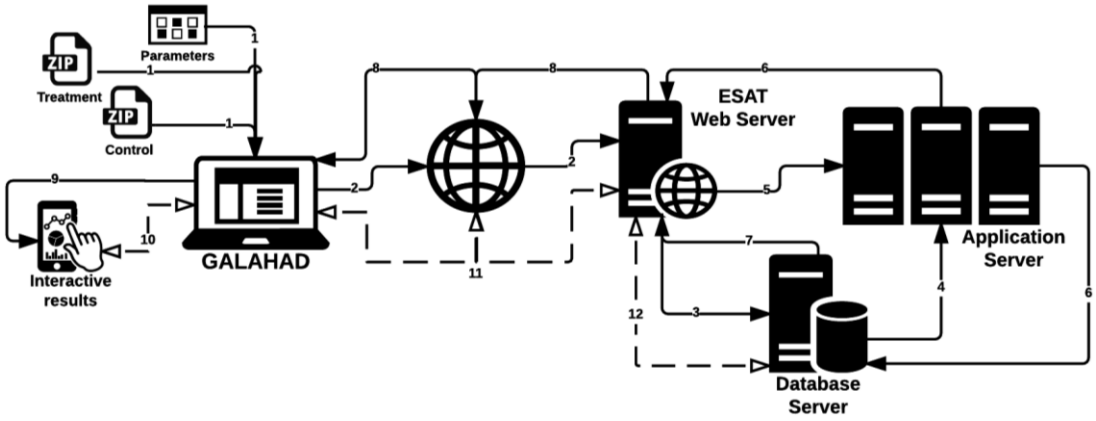


Figure 35: *GALAHAD* web application structure. The user submits the expression data, as well as some parameters to the system. The parameters are stored in the database and the files transfered to the application server, where the main application runs the *GALAHAD* script in R. We developed an R package to read and write from the database and interact with the main R script running the *GALAHAD* algorithm. Several graphs are generated by the R script. Once the analysis is finished, the results are transfered to the database and the images are transfered to the web server. Finally, users view the results through a rich interactive web interface.

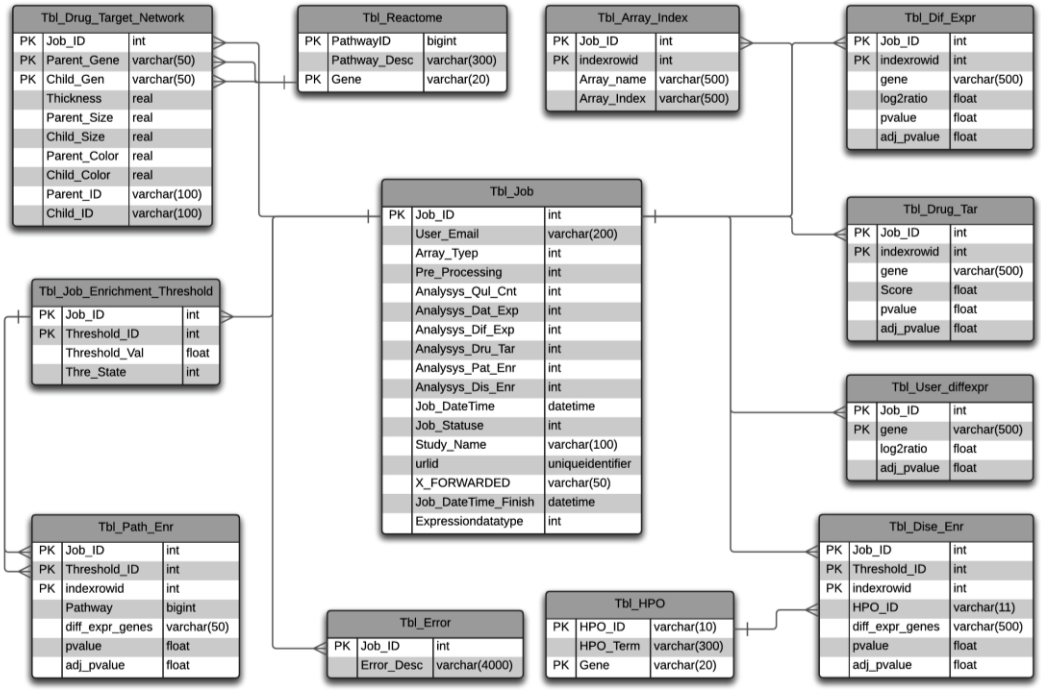


Figure 36: *GALAHAD* data structure ER diagram. Submitted parameters are stored in the Job table. A unique job identifier is assigned to the request. It will be the main reference to track the request status. All the results are stored in different tables. This data structure allows users to interactively query and view the results. Furthermore, if any change is needed in the system, the developer can easily adapt this structure to the new requirements.

2.4.1.3. Procedural complexity

Most of the time, procedural complexity is linked to structural programming and is computed based on the length of the program, the number of logical constraints, decisions or loops. The procedural complexity of software is thus often associated with the complexity of the logical structure of a program [86]. However, the logical structure is considered from another point of view here. We focus on the reasons that make the development of a software and platform necessary and we design our logic based on these reasons. Designers frequently employ a top-down design model, in which the overall program structure is mapped out into separate subsections. Each subsection becomes a module of the application. The basic idea underlying modular design is to organize a large program as a set of distinct components that can be developed independently and then plugged together. These modules work separately and the malfunctioning of any module can affect the whole system. One of the big advantages of developing tools in modular fashion is that the modules can be tested independently. The most important step in this kind of solutions is thus to decide which parts of the system must be divided into components and how these components are going to be linked together. Normally each module should encapsulate information that is not available to the rest of a program. A module will hide the complexity of the information it manages and only provides an overview or subset of this information to the other modules. While modular designs can in principle be implemented in any programming language, implementation is easier if the language supports information hiding by permitting the encapsulation of code and data structures. In addition, there must be support for automating tasks and communication between modules so as to maintain the software and data easily. To ensure the success of a modular design we need to

- Clearly define and identify modules.
- Hide implementation details from other modules.
- Give each module a sufficiently flexible interface to communicate with other modules.
- Avoid that different modules replicate functionalities from other modules.
- Have the possibility to test modules independently.
- Ensure that updates or changes in one module have the smallest possible effect on other modules.

The design of *NGS-Logistics* is used to illustrate how modularity can influence the design. The ultimate goal of *NGS-Logistics* is to allow users to query through a single interface NGS data located in different geographical locations without directly having access to the data. We thus first listed all necessary components, such as (1) an interface to submit a query, track the queries and visualize the results, (2) a database to store information, (3) an access control list to check who has access to what, and (4) a program to run the query at each center.

The *NGS-Logistics* platform structure contains three main modules: a query manager installed in each center, a web application and an access control list manager (Figure 37).

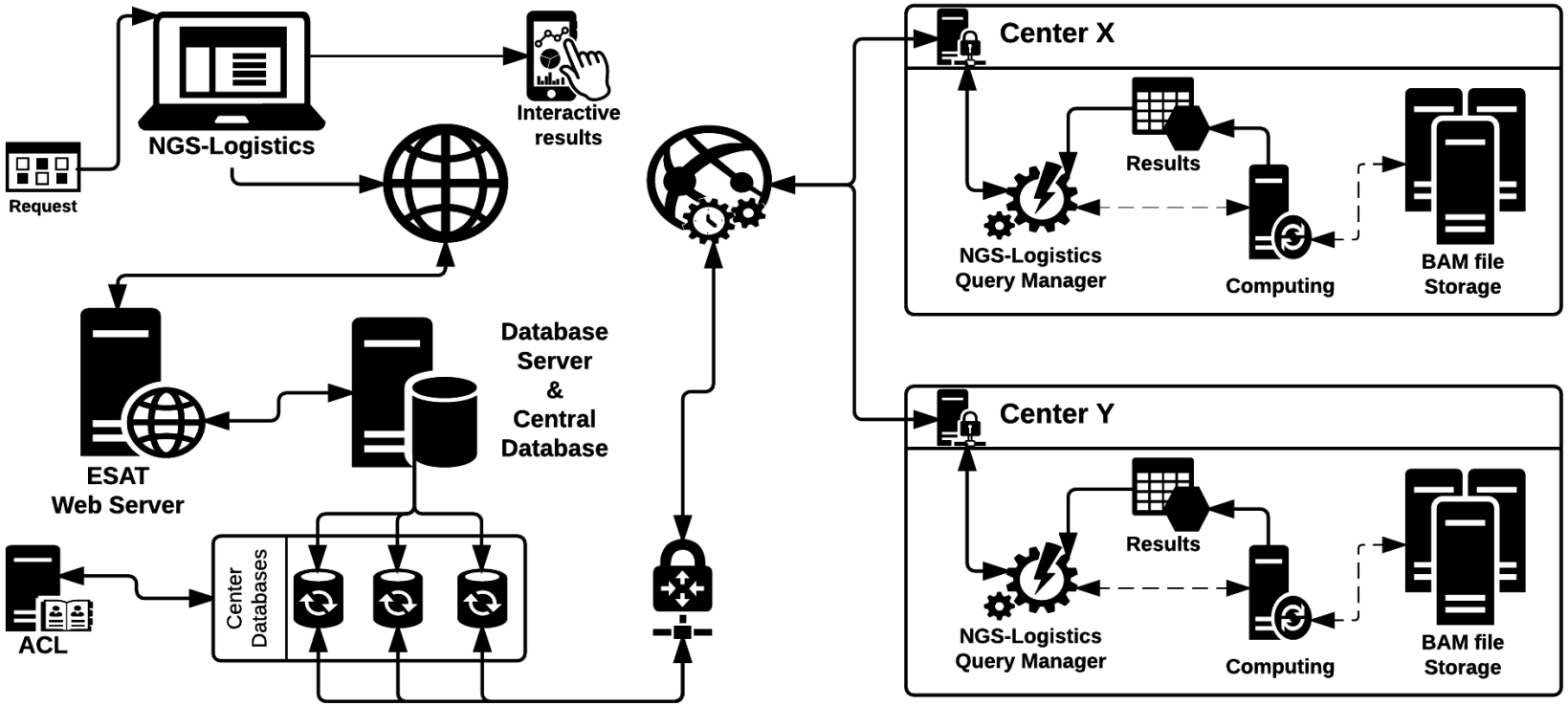


Figure 37: Structure and components of the NGS-Logistics platform. Users pass their queries from the NGS-Logistics web interface to the clients. Requests are stored and scheduled in the main database. Each center has one database and it is the only mean of communication between centers and the main system. The databases of the centers are connected to the main database through a secured connection. Only valid and trusted IPs are allowed to connect via this system. The query manager is responsible for tracking and running the request, as well as collecting and returning the results to the main system.

In this section, we discussed all the major aspects that make a system complex and by giving some examples, we showed which solutions we chose to control this complexity when designing and developing our software. However, these definitions cannot be exhaustive and fixed, and each project will require a specifically tailored solution.

2.5. Big Data

Big data is a term that describes any comprehensive amount of structured and unstructured data that has can be mined for information. Big data can be characterized by the "three Vs" [89]:

- Volume: Organizations collect data from a variety of sources.
- Velocity: Data streams in at an unprecedented speed and must be dealt with a timely manner.
- Variety: Data comes in all types of formats.

The three Vs together describe a set of data and a set of analysis conditions that clearly define the concept of Big Data. More recently, additional dimensions have been proposed to the model, including

- Variability: In addition to the increasing velocities and varieties of data, data flows can be highly inconsistent with predictable or unpredictable peaks.
- Complexity: Data comes from multiple sources and it is necessary to connect, match, clean, and transform data across systems.

Because Big Data takes a lot of time to load into traditional relational databases for analysis, new approaches relying less on data schema and data quality have emerged to store and analyze data. Big Data analytics is often associated with large-scale computing infrastructures because the analysis of large data sets in real-time requires a special platform to store large data sets and new methods to coordinate, combine, and process data faster. Until recently, most efforts have been made for developing algorithms or utilizing supercomputer to process these types of data. Supercomputers help for computation. Small research institutes and individual researchers have to use a scale-out architecture [44] (*i.e.*, a system or network that can be expanded as needed). The idea is to aggregate commodity PCs to increase the computational power and storage capacity. While traditional systems or networks have to be reinitialized or formatted after adding new devices, a scale-out architecture can be expanded horizontally without harming the old system or data storage. Researchers can build the infrastructure based on what they need and increase the capacity as their demand grows. Many methods allow the implementation of computation tasks in such a scale-out network for parallel processing. Recently, many researchers chose to use Big Data solutions, such as those from the Apache Hadoop and HBase projects because they are open source and support scale-out networks very well [90]. Hadoop enables network or cluster management and parallel computation, while HBase is a database model for distributed data storage. Hadoop has been integrated with sequencing data analysis in tools such as CloudAligner and Hadoop-BAM, both using a distributed MapReduce process to speed up sequence alignment [91, 92]. They also support cloud computing to avoid the difficulties of building local computer clusters. Another example, SeqWare uses HBase as the big sequencing data storage platform to provide users high-speed response data extraction [93]. The Hadoop and HBase projects already have a substantial user base

in bioinformatics [90], and the number of instances and cases are growing as well, thanks to their cost-effectiveness. A lot of network service providers now provide online computer clusters for Apache projects implementation at a relatively low cost. For example, to test our Hadoop based application, a cluster from Amazon Web Services has been rented at an average cost of $0.285 per hour ($50 budget for a usage of approximately 175 hours). Nevertheless, most bioinformatics Hadoop-HBase cases focus on sequencing data assembly and practical instances in other bioinformatics fields remain limited. In our work, we have explored the possibility to expand the Apache projects into these new areas of application of bioinformatics.

2.5.1. Apache Hadoop

Apache™ Hadoop® is a high-profile open source project launched by the Apache foundation and aiming for reliable, scalable, distributed data computing [94]. The Hadoop library enables developers to build a cluster of commodity computers for distributed big data processing using simple programming. The cluster is able to scale out from one single node to thousands of nodes to aggregate computational power; each node offering local computation, as well as storage. Hadoop also tracks jobs and nodes status, detects any failure, and manages any rescheduling needed. With this project, we can simulate a supercomputer by just adding normal PCs together. The architecture of a Hadoop cluster is a typical tree structure (Figure 38). The master node of the cluster distributes the data and jobs to each data node, and keeps track of their working status. If one data node fails for a certain job, the master node will reschedule another job to this data node and move the failed job to another other data node. If a node fails to respond entirely, it will be marked dead by the master node and will not be given tasks any more. This mechanism helps the Hadoop cluster to overcome any system failure and makes sure that the computation generates a result. The master node also collects the task results and combines them into a single output.

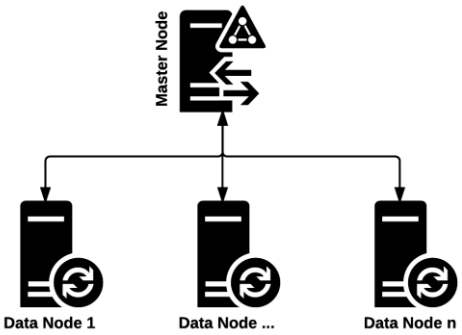


Figure 38: A simplified Hadoop cluster. A Hadoop cluster consists of one master node and several data nodes. The master node controls the whole system. For example, if one node failed on a task, the master node will send that task to another node.

There are two key concepts in the Hadoop project: The MapReduce process and the Hadoop Distributed File System (HDFS™).

2.5.1.1. MapReduce

MapReduce is a data process model that can be managed by programming. It is the core data process method used by a Hadoop cluster. It is a model inherited from parallel computation and is thus a suitable for users who need to handle massive data and have enough machines at disposal [95].The MapReduce process always handles data in "key(k), value(v)" pairs and consists of two major sections. The first one is the *Map* task. The computation is split into small sub-problems and distributed as jobs to all the nodes. Each data node solves the given small problems and returns the answer to the master node. The master node then executes the *Reduce* task to collect all the answers and form a single output. Hence, the whole problem asked to the cluster is solved in parallel:

$$map: (k_1, v_1) \Rightarrow list(k_2, v_2)$$
$$reduce: \left(k_2, list(v_2)\right) \Rightarrow list(k_3, v_3)$$

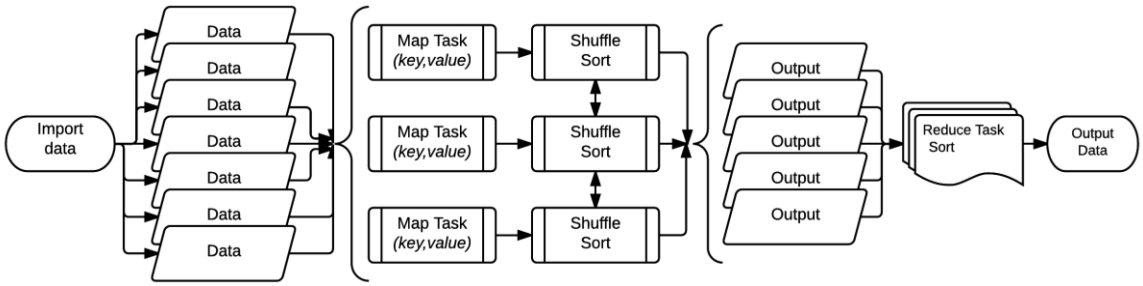


Figure 39: A MapReduce process. Imported data is split into several chunks, Map tasks are run on the key-value pairs,. The results are then sorted and the Reduce task aggregates all the partial outputs into a single output.

As illustrated in the MapReduce process workflow (Figure 39), records in a dataset are split into subsets given to data nodes. Each record is a key-value pair. The computation result(s) in each node is shuffled or sorted as an output. The output of each node is collected by the master node, sorted and combined into a single output data file. As the MapReduce process perfectly matches the architecture of the Hadoop cluster, it is the primary choice for Hadoop data processes.

2.5.1.2. HDFS

HDFS is the storage structure for all Hadoop data storage, including the input and output data. It is designed to fit with the MapReduce process and avoid data loss.

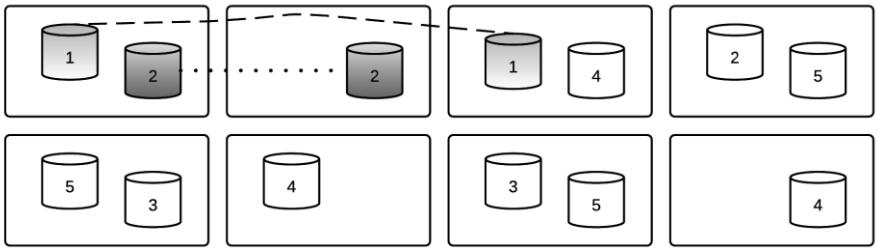


Figure 40: HDFS data nodes. More than one copy of each data chunk is transferred to each data node. In this way, if one node crashes, the master node can request another node to run the failed process.

After the data split, all the data chunks are transferred to data nodes. As showed by dotted lines in Figure 40, Chunk 1 and Chunk 2 are replicated in different data nodes. Indeed, during the transfer each data chunk is replicated, the copies are sent to different nodes, and

the meta-information of the copies, such as tracking information, is stored in the master node. Therefore, system failure will not harm data integrity, and the master node can schedule the computation of the task in another node if one data chunk is lost. The Hadoop MapReduce process and HDFS allow computational power to be aggregated with simple PCs. It is also failure tolerant. When Hadoop is considered for a data processing project, a powerful database model is also needed to handle data storage.

2.5.1.3. Apache HBase

HBase is an open-source distributed database model built on top of Hadoop HDFS. It is column oriented. The prototype of HBase is Google's BigTable. HBase can be considered as the Hadoop complementary project for real-time read and write random access to massive datasets [95]. To implement HBase for data management, a computer cluster consisting of commodity hardware is used. Three major components are needed by HBase: a master node like in a Hadoop cluster, an arbitrary number of region servers similar to the data nodes in Hadoop, and a client for data import and query. The master server keeps the meta-information about the datasets, such as data structure and their tracks. The region servers keep the actual data in regions. One region can only be stored in one server and contains data with contiguous row keys. However, a region server can have more than one region. Thus, the number of region servers of the cluster can be scaled by just adding or removing region servers. HBase can be deployed in multiple file systems, but HDFS is the most straight forward choice because it is also scalable, failure safe, and offers block replication for data integrity. The two Apache projects both fully support the MapReduce data process and HDFS is therefore the most suitable file system for HBase.

As most of the big data solutions, HBase provides to the users a concept of column family (Figure 41). A column family is a prefixed field that has to be defined when the data table is created. Each column family can have multiple sub-columns identified by their column qualifiers. The qualifiers are not fixed in each row and do not have to be defined at table creation. Users do not have to know the data structure in advance or during data import, making the data storage straightforward. The row schema in HBase is column oriented. For each row key, users can access the value of the cell with the combination of column family and qualifier. Several versions of the value can be stored in the same cell. Each version can be distinguished by its timestamp (*i.e.*, the time at which the value has been inserted in the cell). By default, HBase gives the latest version of the value, but an older version of the value can be accessed with the correct timestamp.

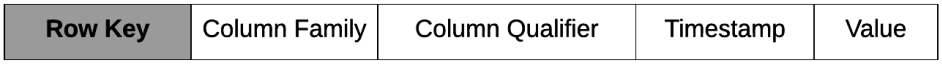


Figure 41: A schema of a data cell in HBase. In HBase, each data cell has a unique identifier which acts as the primary key. Then the type of data is identified by the column family, which could have subcolumns as well. Subcolumns are identified by the column qualifier. The timestamp can be used in case of tracking the changes over the data. Finally, the record contains the actual value of the data.

As a distributed system, the most efficient method to manage the data in HBase is the MapReduce process. However, building a MapReduce task requires strong programming skills. Also, the MapReduce task has to be modified to adapt the specific purpose of each operation. This is a bottleneck for searching, altering, querying and retrieving data in a database interactively. To overcome this disadvantage, HBase provides a CLI (Command

Line Interface) named HBase shell. In the shell, user can use HBase built-in commands to access the database, and all the operations are implicitly translated into MapReduce jobs, so that the distributed system can execute the jobs in parallel. While the shell brings a lot of convenience for daily database usage, advanced users can still access HBase by using API and MapReduce job programmatically for customized purposes. In our work, Hadoop and HBase were used for VCF file process and storage.

2.5.1.4. Apache Hive

Apache Hive is a data warehouse model built for Hadoop and originally developed by Facebook [95]. It has multiple features: (1) it is built for extremely large datasets, (2) it is mainly used for log analysis and ad hoc querying, (3) it supports multiple data types, and (4) it is very extensible. Its most important feature and the reason for which Facebook developed Hive is that it uses the SQL query syntax:

```
SELECT miller.row_key
FROM miller
JOIN test
ON miller.chrom = test.chrom
WHERE miller.qual > 500
ORDER BY miller.row_key
```

```
SELECT * FROM
(Select table1.row_key AS row_key
FROM test table1 WHERE table1.filter = "PASS"
UNION ALL
Select table2.row_key AS row_key
FROM miller table2 WHERE table2.qual > 500
AND table2.chrom = "20") tmp
ORDER BY tmp.row_key
```

It is possible for people with an SQL background to analyze data stored in HDFS without any further programming skills. Hive translates SQL queries into MapReduce jobs that Hadoop subsequently executes. Hive is thus a good model for Hadoop analysis since SQL is still the most popular scripting language for database manipulation. As a result, creating a complex data extraction job from HDFS only takes ten minutes. Hive can integrate with HBase as well. With the integration, Hive can create an external table as a linkage to the real HBase data table. The operation provides several advantages to users. First, the data is physically separated from the users. All transactions in Hive are local, only carried out in the Hive server against the linkages. Thus, the original data in HBase remains untouched. For this reason, the data cannot be damaged by any query. Second, Hive queries are translated into MapReduce jobs implicitly. Thus, users do not need to worry about coding MapReduce jobs. Instead, they only need to focus on constructing their query statements. Although simple queries can be implemented with the HBase shell to avoid programming, HBase uses a NoSQL model, which still requires most users to learn extra commands to implement their operations. Hive is more user-friendly because it supports the well-known and easy to understand SQL syntax almost completely. Analyzing data in HBase with Hadoop MapReduce process thus becomes possible for researchers with intermediate levels of IT skills.

As discussed before, VCF files can be extremely large and have a complex structure, so it is a good choice to integrate Hadoop, HBase, and Hive together instead of using them one by one. Hadoop is the base for distributed data processing, HBase for data storage, and Hive for data retrieval. The integration architecture for these three Apache projects is shown in Figure 42.

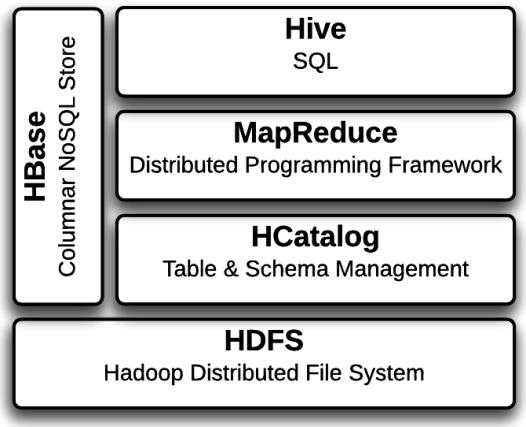


Figure 42: The architecture of the Apache integration. The Hadoop distributed file system is the lowest level of the system. Actual data is stored at this level. HBase is used to store data in columnar format. Tables and schema models are managed at the HCatalog layer. The MapReduce layer handles the actual computation. Hive allows users to query data through the SQL syntax, even though the data is not actually stored in a relational database.

So far we discussed the fundamental concepts of Apache Hadoop, HBase, and Hive. In the next chapter, we show how these tools are integrated to create a model to store, process, and analyze NGS data.

2.6. Project descriptions

In Table 7, we list the project descriptions. The purpose of this section is to provide a high-level description of each of the software programs that will be described in more details in the following chapters. In short, the following table is meant to equip the reader with a solid understanding of the systems. While software requirement specifications were discussed in previous sections, the current section offers a summary of the projects, including goals and objectives, general details, and some major challenges associated with the platforms.

Table 7: Project descriptions

Name	Description	Key terms	Computation problems	Programming language and database	Expected Output
NGS-Logistics	Federated analysis of NGS sequence variants across multiple locations	Data sharing, variant discovery, BAM, VCF, ACL, privacy, data protection, federated database	Processing time, storage, communication between nodes and central database, pipeline standardization, ACL management	Java, C#, JavaScript, MSSQL	Rare variant and variant frequency discovery
GALAHAD	A web server for drug effect analysis from gene expression	Gene expression, Affymetrix arrays, drug target prioritization, drug effect prediction, drug repositioning, disease – phenotype – drug association	Genome-wide drug target prioritization, database structure, result visualization, user request tracking without need for user registration	R, Java, C#, JavaScript, MSSQL	Drug repositioning
SNIPTS	Big Data solutions for variant discovery from low coverage sequencing data by integration of Hadoop, HBase and Hive.	NIPT, NoSQL, variant discovery, SNV calling, Big Data, low coverage sequencing	Processing time, huge amount of data, parallel computing, database management, low quality sequencing data	Java, Hadoop, HBase, Hive	Variant reference database
Endeavour	A web tool for candidate gene prioritization	Gene–disease association, disease causing genes prediction, gene prioritization, training, scoring, fusion	Processing time, query builder, database structure and management, results visualization, heavy computing	Java, C#, JavaScript, MSSQL	Disease-causing gene discovery

Chapter 3

NGS-Logistics

- Title:
  NGS-Logistics: federated analysis of NGS sequence variants across multiple locations
- Published:
  Genome Medicine, 17 September 2014
- Publication link:
  http://www.genomemedicine.com/content/6/9/71
- Authors:
  Amin Ardeshirdavani [1,2], Erika Souche [3], Luc Dehaspe [3], Jeroen Van Houdt [3], Joris Robert Vermeesch [3], Yves Moreau [1,2].

  1. KU Leuven, Department of Electrical Engineering (ESAT), STADIUS Center for Dynamical Systems, Signal Processing and Data Analytics, Kasteelpark Arenberg 10, Leuven, 3001, Belgium
  2. iMinds Medical IT Department, Kasteelpark Arenberg 10, Leuven, 3001, Belgium
  3. KU Leuven, Center of Human Genetics Gasthuisberg, O&N I Herestraat 49, Leuven, 3000, Belgium
- Author contribution:
  The PhD candidate designed, developed, implemented, and maintains the *NGS-Logistics* platform. The PhD candidate coordinated the validation of the software. The PhD candidate drafted the manuscript.

3.1. Abstract

As many personal genomes are being sequenced, collaborative analysis of those genomes has become essential. However, analysis of personal genomic data raises important privacy and confidentiality issues. We propose a methodology for federated analysis of sequence variants from personal genomes. Specific base-pair positions and/or regions are queried for samples to which the user has access but also for the whole population. The statistics results do not breach data confidentiality but allow further exploration of the data; researchers can negotiate access to relevant samples through pseudonymous identifiers. This approach minimizes the impact on data confidentiality while enabling powerful data analysis by gaining access to important rare samples. Our methodology is implemented in an open source tool called *NGS-Logistics*, freely available at https://ngsl.esat.kuleuven.be.

3.2. Background

NGS is a key tool in genomics, in particular to study inherited and acquired human genetic disorders [9]. Multiple projects now aim at mapping the human genetic variation on a large scale, such as the 1,000 Genomes Project [19], the UK 100 k Genome Project [20], or the Genome of the Netherlands project [21]. Meanwhile, with the dramatic decrease in price and turnaround time, large amounts of human sequencing data have been generated over the past decade [17]. As of August 2014, about 2,555 sequencers were spread over 920 centers across the world [18]. As a result, about 100,000 human exomes have been sequenced so far [22].

Crucially, the speed at which NGS data are produced greatly surpasses Moore's law [17] and challenges our ability to conveniently store, exchange, and analyze the data. Data processing is needed to extract reliable information from sequencing data and it can be divided into two major steps: primary analysis (image analysis and base calling) and secondary analysis. When looking for variation in the human genome, secondary analysis consists of aligning/mapping the reads against the reference genome and scanning the alignment for variation. Both raw data (that is, the reads produced by the sequencer as well as their quality values, generally FASTQ files) and mapped reads (usually BAM files [61]) are large files occupying significant disk storage space. The collection of files resulting from the analysis of a single whole genome study can take up to 50 GB of disk space [33]. This raises significant issues in terms of computing and data storage and transfer, with off-site data transfer currently being a key bottleneck.

Moreover, the analysis of NGS data also raises the major challenge of how to reconcile federated analysis of personal genomic data and confidentiality of data to protect privacy. In many situations, the analysis of data from a single study alone will be much less powerful than if it can be correlated with other studies. In particular, when investigating a mutation of interest, it is extremely useful to obtain data about other patients or controls sharing similar mutations. However, personal genome data (whole genome, exome, transcriptome data, and so on) are sensitive personal data. Confidentiality of these data must be guaranteed at all times and only duly authorized researchers should access such personal data. Moreover, important issues around informed consent also arise. What are acceptable uses of personal genomic data, in particular for data that have been collected primarily for a clinical purpose rather than for a research study? These questions are particularly hard because the legal landscape surrounding them and the interpretation of

existing regulation varies from jurisdiction to jurisdiction, and is quickly evolving over time. We cannot resolve those questions here. What we do in this article is show what can be done from a technical perspective to balance data confidentiality and powerful data analysis for the benefit of patients and the community.

3.2.1. Data sharing

Biology and especially genomics is a data-rich environment. Tools and methods are constantly being improved; each method having different features compared with previous versions. Since the output format of NGS sequencers has converged to standard formats over the past years, this provides a great opportunity for bioinformaticians to reconstruct their pipeline based on new methods and software to get better results. The reusability of data makes researchers eager to access each other's data, but creates the additional challenge of providing this access at a sufficiently detailed level. We therefore aim at a solution that allows querying the data down to the level of aligned reads (BAM/SAM files) rather than only at the level of called variants (Variant Call Format (VCF) files) because those may vary significantly as calling algorithms evolve.

In addition, the number of similar cases needed to confirm a clinical or biological hypothesis has always been an issue. Gathering similar observations can lead to more accurate results and conclusions. In most situations, a change in one single nucleotide can affect the whole mechanism of a gene and consequently corrupt the behavior of the pathways involved, which could cause a disorder [25]. Therefore, comparison across multiple patients and controls is a key feature in studying single-nucleotide variants (SNVs) and in genomics research in general. In practice, smaller data sets and data from routine clinical work are often not shared, although they also have the potential to add information that may be useful for the analysis of other research projects. Another example is that of building and sharing a database of variant frequencies in unaffected individuals, which will help in excluding common variants from further analysis. This is especially important since common variants are highly variable between populations and thus regional databases will play a key role in interpreting variants for research and clinical purposes. Thus, some form of data sharing is highly desirable.

3.2.2. Privacy

The current European legal framework [96] on data privacy is the main influence on our work, but similar issues arise in the US legal framework and elsewhere. Moreover, in a setting of international collaboration, any solution will have to fulfill all legal requirements and thus focusing on the slightly more conservative EU framework may be advantageous.

A discussion of privacy legislation is beyond the scope of this article, but a brief overview of some key concepts is useful to understand the motivation for our work.

Personal data [97] comprises any information relating to a person who can be identified, directly or indirectly, in particular by reference to an identification number or to one or more factors specific to his physical, physiological, mental, economic, cultural or social identity. It consists of any data for which someone can link the information to the person it originated from, even if the person holding the data cannot make this link.

Personal data can only be processed under the three key principles of transparency, legitimacy, and proportionality. Transparency implies that individuals have a right to be

informed when and how their personal data are being processed. This includes the need for appropriate consent for processing of the data, and the need for identifying all people processing the data. Legitimacy implies that personal data can only be processed for specified explicit and legitimate purposes and may not be processed further in a way incompatible with those purposes. Proportionality implies that personal data may be processed only insofar as it is adequate, relevant and not excessive in relation to the purposes for which they are collected or further processed. This includes the right to correct or withdraw information. By contrast, and importantly, data that are not personal - which means data that cannot be linked back to a specific individual - are not subject to these requirements.

Personal genomic data comprises any substantive amount of genomic sequence data (for example, human whole genome, exome, transcriptome, or epigenomic sequencing) that are deemed personal by way of being unique to each person and fixed. Importantly, this implies that removing classical identifiers from the data (names, social security identifiers, patient identifiers, or pseudonymized versions of those) is insufficient to keep the data from being personal.

3.2.3. Data management

Data control is an important and complex issue when dealing with personal data. It is therefore necessary to determine and know "who has access to what". This corresponds to the key principle of "transparency" in the management of personal data. In most research institutes, it is the responsibility of the principal investigator (PI), the head of a group, to supervise the access of researchers to the data of a study, after approval by an institutional review board. In a simple scenario, the PI defines the study and assigns the researcher to it, and only those researchers need to access the data. However, there is often a need or desire to share data with other researchers from the team, and it can also often be useful to share the data and results with other groups within or outside the institution. Privacy principles do not require that the number of people accessing the data be minimized at all cost, but rather that the process be managed transparently, that the data be used for legitimate purposes, and that only people with a legitimate interest access the data.

A potential solution to manage access to personal genomic data is the use of access control lists (ACLs). An ACL can be defined and assigned to a corresponding study, and researchers can then be assigned to this ACL. This feature allows control of which researcher has access to which study. Many operating systems support ACLs, although this is not yet a fully standard feature. For a center with several groups and several studies, however, it can quickly become difficult to maintain all these access permissions as the number of samples and studies increases. There is therefore a need for a data management system that can handle these permissions efficiently.

3.2.4. Data storage and processing

Monitoring of pipelines shows that there are several bottlenecks in terms of data storage and processing. Processor speed, virtual memory, and disk storage are the most frequently observed bottlenecks. To analyze large amounts of data, organizations will increase their computational power by upgrading their existing servers (scale up) and nowadays by adding computing and storage servers (scale out) to their hardware facilities [44]. Therefore, a well-designed infrastructure supported by powerful computational resources

and organized storage servers is essential. As mentioned before, by having access to similar cases and by comparing different studies, researchers can make more accurate conclusions. Consequently, having access to other data across studies is necessary. However, it is often far from optimal to store copies of additional data sets next to the user's own data because of the significant costs involved, as well as the time necessary to obtain the data and to transfer it locally. Being able to investigate other data sets without actually moving the data could increase research efficiency.

3.3. Aims

Our ultimate goal is to provide a way for biomedical researchers to ask high-level questions, such as "what is the frequency of appearance of a single mutation across the large population" or "how many patients with disease D have a mutation in gene G?" transparently. Our system also allows detailed anonymized analysis of genome sequencing data at single positions in the genome, for example, to distinguish rare mutations from highly variable or poorly sequenced positions in the genome. To address all the issues described above, we have developed a data infrastructure, *NGS-Logistics*, that fulfills all requirements of a successful application that can process data inclusively and comprehensively from multiple sources while guaranteeing privacy and security by avoiding the exchange of raw data. *NGS-Logistics* is a web-based application providing a data structure to analyze NGS data in a distributed way. The data can be located in any data center, anywhere in the world. *NGS-Logistics* provides an environment in which researchers do not need to worry about the physical location of the data. With respect to user's rights, queries will be sent to each remote server. The host will process the request and return the results back to the main server where all the privacy limitations are controlled for the data. Once the results are ready, the end user can see the desired information (Figure 43). In the next section, we provide a detailed description of functionalities of the system and its implementation.

3.4. Implementation

The *NGS-Logistics* application package consists of three modules:

1. Administration
2. Query manager
3. Primary user interface

Figure 43 shows the components of *NGS-Logistics*. *NGS-Logistics* is designed as a multi-tier architecture that allows the development of a flexible and reusable application. It consists of three tiers: presentation, application processing, and data management. Different software packages are used depending on the nature of the component. Currently, Java is used for the administration and query manager modules, while ASP.NET is used for our user interface. Microsoft SQL Server is used as our database management system. These are local design choices and the application could be redeployed fairly easily using other programming technologies. Each one of these components and their characteristics is further explained below.

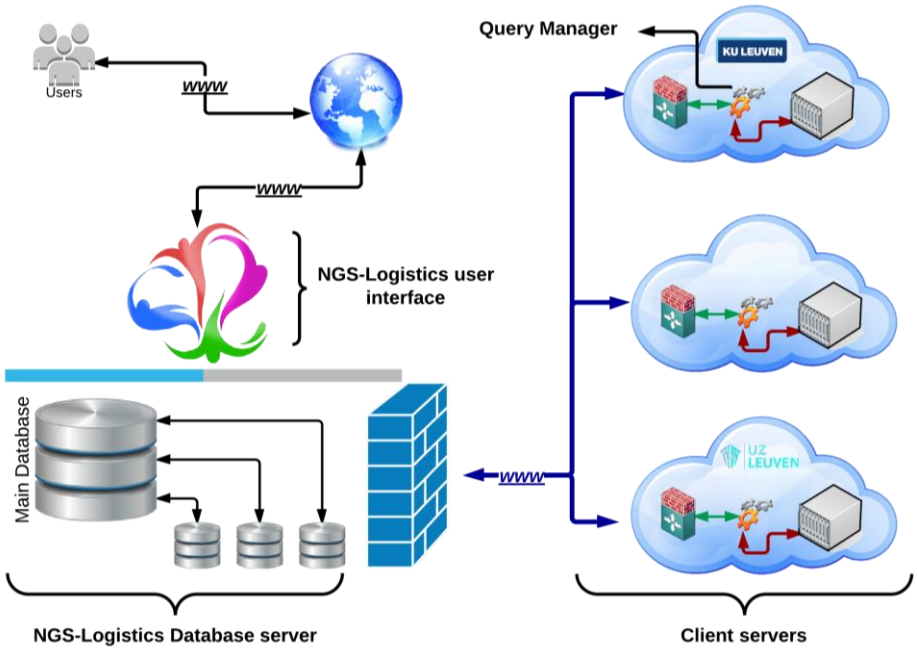
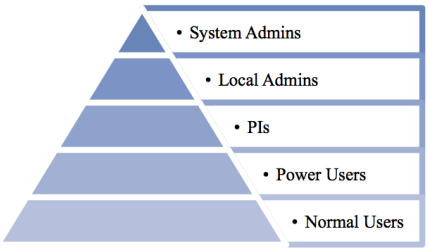


Figure 43: NGS-Logistics components. Users pass their queries from the NGS-Logistics web interface to the clients. Requests are stored and scheduled in the main database. Each center has one database, being the only way of communication between centers and the main system. Centers and their databases are connected through a secured connection, to which only valid and trusted IPs are allowed to connect. The query manager is responsible for tracking and running the request, as well as collecting and returning the results to the main system.

3.4.1. Administration

The administration module mainly focuses on controlling the access rights of the users. It is the place where the administrator sets up access control among users, so that confidentiality of personal genomic data can be guaranteed. The system administrator can also control the type and volume of queries that the different users are allowed to submit. The user interface is designed in such a way that the complexity of users' access controls is hidden. For finer control over users, users can be assigned roles at different levels (Figure 44).



- System Admins: control the whole data structure
- Local Admins: define and manage data sets for a center
- PIs: create data sets and assign users to data sets
- Power Users: manage data set access for a PI
- Normal Users: submit queries and view results

Figure 44: NGS-Logistics user types and their access levels.

The system administrator defines the centers and their local admins. Local admins are responsible for ensuring that the sample list is kept updated and the samples are assigned to the authorized PIs. They also manage the users' accounts, assign samples to data sets, and allocate data sets to users or groups of users. Data sets can be private or public (that is, accessible to all the users of the system) depending on the characteristics of the samples that they contain.

3.4.1.1. Sample list

The sample list is the list of all files that each center makes available through *NGS-Logistics*. The sample list can be updated either automatically or manually by the local admin. Several fields such as the "Sample Local Name", "PI Name", "Sample Type", and "Sample Reference version" are compulsory for each sample. At the time of insertion, *NGS-Logistics* automatically assigns an internal Sample ID to each sample to pseudonymize the actual name (Sample Local Name) given to the sample by the owner from the rest of the users. (Note that pseudonymization does not stop the data from being personal because the genome or exome sequence is unique to each individual). Then it uses the "PI Name" and "Sample Type" fields for recognition of the type of sample (Research or Diagnostics) and assigns them to the owner of the sample (PI Name). The "Sample Reference Version" shows the version of the human genome build to which the sample has been aligned. Since samples can be aligned to different builds of the human genome, the system will use this information to aggregate query results. To do so, we use the UCSC LiftOver database [98].

3.4.2. Query manager

The query manager is a component that picks up the request at each center, runs the task, gathers the results, and returns them back to the main system. Since there are different infrastructures and software packages in different centers, we have tried to keep the query manager as simple as possible. To establish a connection between the server and clients, we only need to configure the query manager by changing a small list of parameters, such as the address of the executable applications and of the data (BAM files). As we need to integrate all results from different sources, we query BAM files with GATK [66]. The output looks like a standard VCF file [99], which is exported from the different centers to the main system. Standard GATK variant caller output has the following fields: Chromosome, Position, dbSNP ID, Ref, Alt, Qual, Filter, Info Format, and Format Values.

3.4.3. User interface

The user interface plays an important role in the success of an application. Well-designed user-friendly interfaces have a great impact on the users' activities. Therefore, we have tried to make the interface as simple and intuitive as possible. The *NGS-Logistics* interface allows the users to submit their queries, track them, and visualize the results. To further explain this, we divide the interface into three categories: user settings, query builder, and results.

3.4.3.1. User settings

User settings allow users to alter their information, create their own data sets, assign samples (to which they have authorized access) to these personal data sets, and activate their data sets. Users can choose the data set they want to investigate from the list of those to which they have authorized access. This procedure is called Data Set Activation. Activating a data set is important: if no data set is activated, no detailed results will be available (in the results section, we will discuss what kinds of detailed information can be investigated). This feature is part of our security system and privacy agreement. In this way, we guarantee that users can only see information they are authorized to see. Besides detailed information, some summary values (such as the total number of appearance of a

variant or average coverage) are displayed. Users can also ask for access to samples and data sets from other groups. In this case, the owner of the data must review the request and grant access to the data or not.

3.4.3.2. Query builder

Two types of queries are available: point and area statistics. GATK is used to call variants for both types of queries but results and statistics are shown slightly differently for each type of query. Since GATK provides two variant callers, users can choose between HaplotypeCaller and UnifiedGenotyper. In point queries, users must fill in the chromosome and position and have the possibility to genotype reference sites (allSitePLs); whereas in area queries, they have to provide the chromosome number and start and end positions, or a gene name. After submitting the query, a unique request ID is assigned to the request. In the background, the system automatically sends the query to the centers and waits for results. Meanwhile, through the request tracking option, users can see the status of the request. Since the process really depends on the facilities and workload of each center, we cannot estimate the duration of each process but display the progress by center.

3.5. Results and discussion

All functionalities of *NGS-Logistics* will be illustrated by querying one gene. Another example will demonstrate how *NGS-Logistics* can help in interpreting variants. A demonstration run was performed and successfully analyzed samples located at two centers (KU Leuven and UZ Leuven) with approximately 1,500 exome samples in total.

3.5.1. Use Case 1

We use the example of SMARCA2, a gene located on chromosome 9 whose heterozygous mutation causes Nicolaides-Baraitser syndrome [100]. We start with the area query. An area can be queried using a gene name or a region (defined by chromosome number and start and end positions). The area query results section is divided into three sections: “Summary”, “Sample to Position”, and “Position to Sample”. The top part of each section describes the query and includes, if possible, links to well-known online databases, such as the UCSC and Ensembl genome browsers. The “Summary” section displays the total number of samples queried in each center and the total number of samples to which they have access. Results can be investigated by checking the list of available mutations per sample (“Sample to Position” section) or by checking the list of all mutations for the selected region, how frequently they occur, and which genotypes are observed in all samples from all centers (“Position to Sample” section) (Figure 45). We refer to all samples from all centers as the “Whole Population”. Note that the “Whole Population” contains the samples from the active data set. One of the SNVs reported in SMARCA2 has the dbSNP id rs281875187 (chr9:2115841, build 19). As shown in Figure 3, this SNV is not observed in any sample from the active data set, but in the whole population, one sample is heterozygous. If other samples were found with this mutation, it would be of direct interest to a researcher to contact the PI responsible for this sample, for example, to investigate whether or not this patient is affected by Nicolaides-Baraitser syndrome to confirm or weaken the hypothesis of association between Nicolaides-Baraitser syndrome and SMARCA2 mutations, or to collect biological samples from patients for further biological studies.

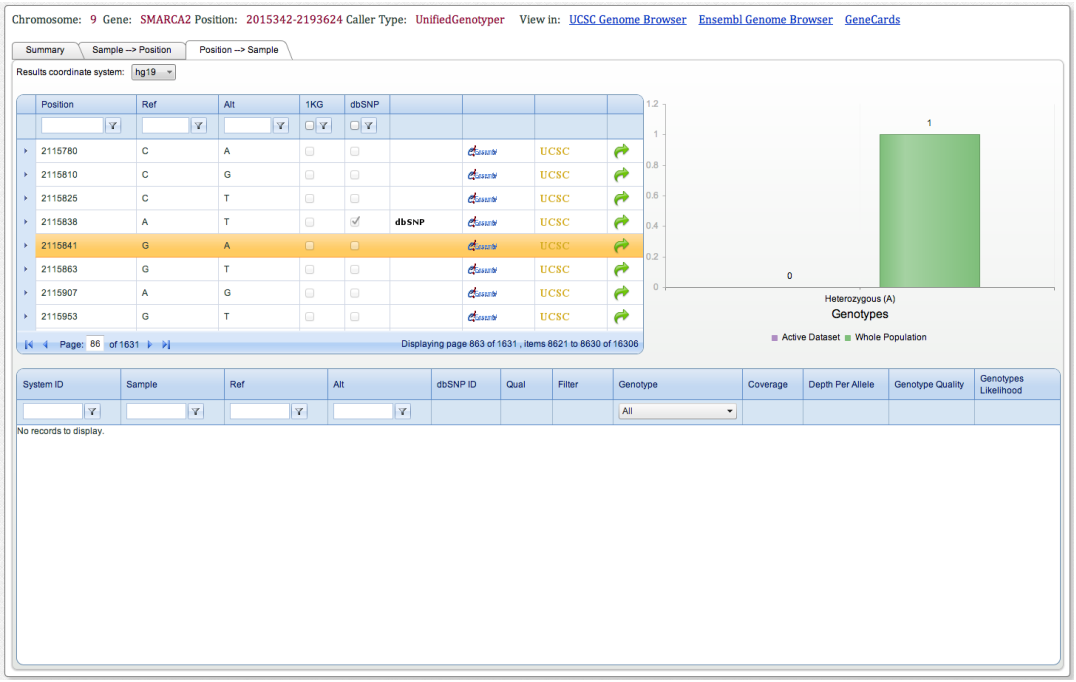


Figure 45: NGS-Logistics area query results page (Position to Sample section) for SMARCA2. Results demonstrate that at the selected position (chr9:2115841, build 19) a G to A mutation can be observed with heterozygous genotypes. The graph shows the differences between the active data set and the whole population.

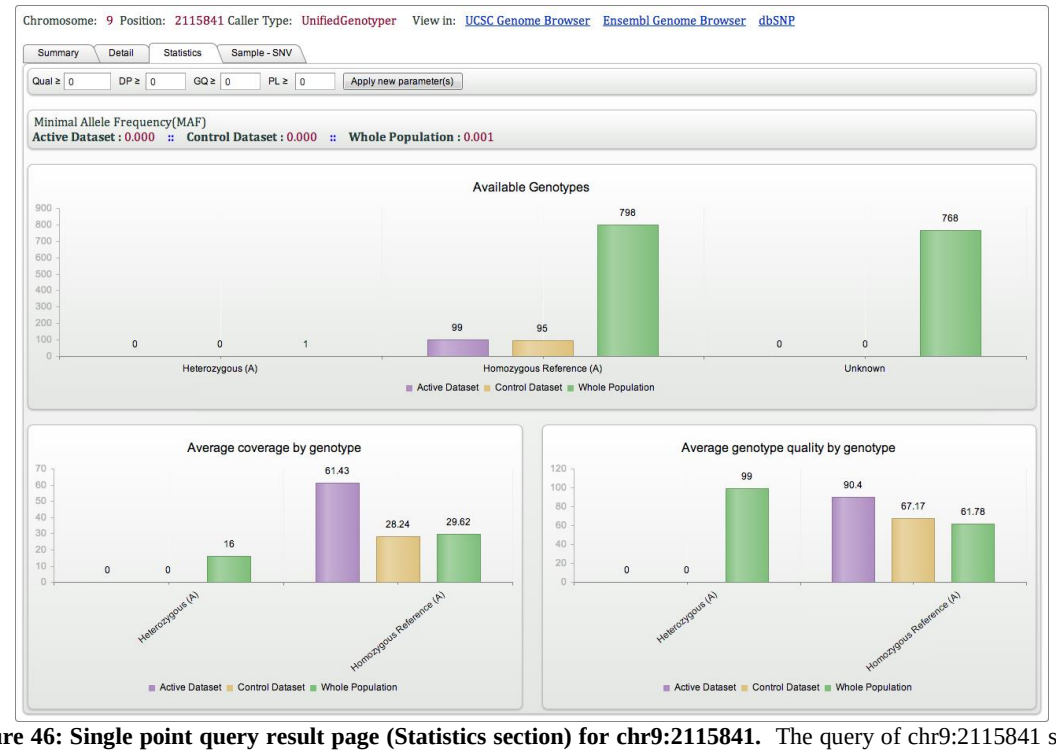


Figure 46: Single point query result page (Statistics section) for chr9:2115841. The query of chr9:2115841 shows that only one sample is polymorphic at this position. All samples that can be genotyped at this position from the active data set, the control data set and the whole population are homozygous reference. The MAF of this variant in each data set is thus very low.

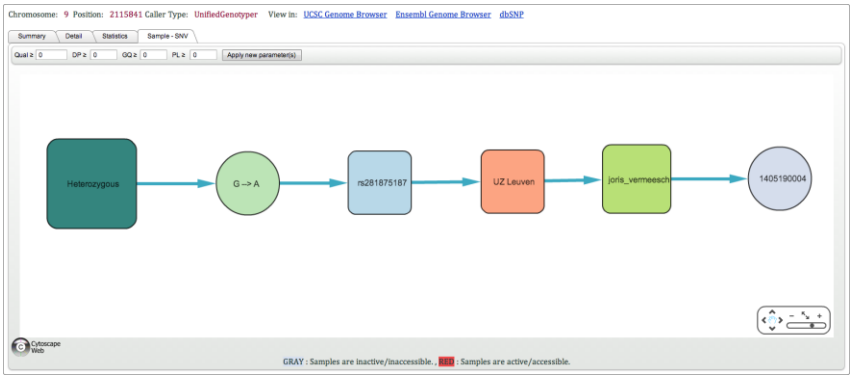


Figure 47: Single point query results page (Sample to SNV section) for chr9:2115841. Those samples that have a mutation are clustered based on their genotypes, dbSNP ID, Center, and Sample Owner name.

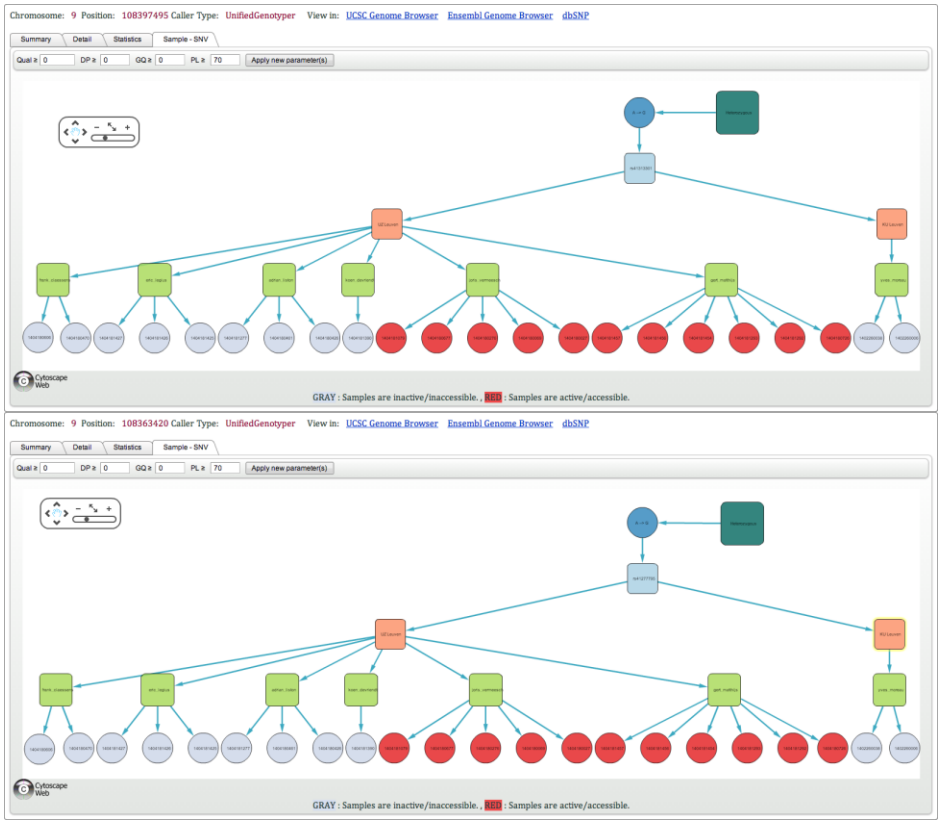


Figure 48: Single point query results page (Sample to SNV section) for chr9:108363420 and chr9:108397495.

The single point query allows users to investigate a single position. The single point query results page is divided into four sections: “Summary”, “Detail”, “Statistics”, and “Sample to SNV”. Similarly, to the area query, the top part of each section contains information related to the query, as well as links to dbSNP and the UCSC and Ensembl genome browsers. The “Summary” section displays information on the total number of samples queried in each center and summary results (number of samples genotyped as the same as or different to the homozygous reference, minimum, maximum and average quality scores). The “Detail” section contains the GATK output for the samples of the active data set. Results are available for download. The “Statistics” section provides an overview of the different genotypes and their quality values. Graphs are divided into three sets of data: the active data set, the control data set and the whole population (Figure 46). The control

data set includes unaffected, unrelated individuals. Usually these individuals are either unaffected parents or unaffected siblings of affected patients. As such they might carry one pathogenic mutation. Each center can provide a list of these individuals. The minimal allele frequency (MAF) is calculated for each data set. A call is assigned to each sample and by default no quality filtering is performed by *NGS-Logistics*. However, users are free to select good quality variants according to quality, depth, genotype quality and/or PL (phred-scaled genotype likelihood) values.

Finally, the section "Sample to SNV" is the demonstration of the relation between available genotypes and samples (Figure 47). Samples are clustered based on their genotype, dbSNP ID, center, and PI name. Every sample is assigned a pseudonymized System Sample ID, different from the real sample ID. This identifier can be used to request access to the sample details. Samples for which the user is not allowed to see the details are colored in grey. This system allows the quick identification of the person to contact to get access to a relevant new sample.

3.5.2. Use Case 2

The variant analysis of three patients with a congenital disorder of glycosylation revealed two good quality heterozygous variants (chr9:108363420 and chr9:108397495, build 19) in the fukutin gene, responsible for muscular dystrophy-dystroglycanopathy. Since fukutin is a recessive gene disorder, two heterozygous variants on different alleles or one homozygous variant could cause the disease. The use of *NGS-Logistics* showed that both variants were reliably genotyped as heterozygous in 22 individuals and had a MAF of 0.05 and 0.03 in the control data set and the whole population, respectively. The "Sample to SNV" graphs (Figure 48) show that the same 22 individuals carried both variants in a heterozygous state, indicating that the variants are on the same allele. Consequently, the combination of the two variants found in three of our patients cannot cause the disease.

3.6. Conclusions

There is currently no comprehensive solution for personal genomic data sharing. The Database of Genotypes and Phenotypes (dbGaP) [101] and the European Genome-phenome Archive (EGA) allow the exchange of personal genomic data. However, some important limitations remain. The process of securing approval for data access is time consuming and involves significant administrative overhead because the process involves legal contracting between the requesting institution, dbGaP or EGA, and the institution providing the data. dbGaP or EGA acts as broker between institutions and maintains the data sets under escrow until approval of the requests. Data transfer to the receiving party, reprocessing of the data, and maintenance of the data locally represent significant hurdles for non-specialists and require significant computational resources. The need for reporting and project closure adds further to the administrative overhead. As a result, there is a major barrier to using those resources and the available data are not being exploited to the full extent. It also slows down the adoption of those systems by institutions that could be providing significantly more data than is happening in practice.

Another solution for data sharing is to rely on vendor solutions, such as Illumina BaseSpace, or more recently Google Genomics. Well-integrated vendor solutions are tempting because they could allow easy access to data sets from collaborators spread across the world through cloud-based solutions. However, it is unclear how such solutions will

support a privacy-enhancing framework beyond leaving access decisions entirely at the responsibility of individual researchers. Moreover, such solutions could present a significant risk of network effect. If significantly more and more data become available through a given vendor, this solution will acquire a major advantage over the competition. Given broadly similar features, the network with the most data is by far the most valuable because it allows more research questions to be answered. A dominant network could possibly wipeout all significant competition, and gain a permanent lock-in over significant amounts of personal genomic data worldwide. Therefore, issues of platform interoperability and data freedom are essential when considering vendor solutions for personal genomic data sharing.

Further access to the samples by providing a researcher with direct access to given samples after a request is subject to the standard process. Does the consent provided by the patient or guardian allow for legitimate access by the third party? Or is additional consent or approval by the institutional review board necessary? This process cannot be automated at this point because the answer depends on the specific details of the request and on the legal framework imposed by the jurisdiction or institution. However, the fact that all users and requests are tracked provides a high level of transparency, which should ease this process greatly.

A key design choice in *NGS-Logistics* has been to provide data analysis at the level of aligned reads (BAM files) rather than called variants (VCF files). While deploying a system handling only VCF files might be easier, we believe that for detailed analysis of a single nucleotide position it is desirable to use a uniform assessment of variants across all samples, which will be difficult to guarantee for legacy VCF files produced at different times using different pipelines. Besides querying BAM files, *NGS-Logistics* allows the genotyping of reference calls, which are often not included in standard VCF files. Knowing whether a sample is homozygous reference at one position or whether the position is not covered is important since it allows a better assessment of the MAF.

One way in which the power of the system can be increased is by making healthy control population sets available through the system. Currently, the “Whole Population” contains all samples available in the system regardless of clinical diagnosis. Once a relevant mutation has been identified, samples can be tracked down, and clinical information can be gained about these samples. However, the “Whole Population” will contain samples that can be considered “cases” as well as “controls” without distinction. This prevents us from implementing any form of association scoring in *NGS-Logistics*. So far, we have one control population, but it would be useful to add phenotypic information in the next phase, so that association scores can be calculated for mutations between the active data set and a control data set with respect to their phenotype. Promising mutations can then be evaluated in detail in the system. Similarly, another difficulty for any association analysis is that samples from the same patient could be processed at different sites (that is, seen by physicians at different clinical centers) with no communication that these samples are identical. This would skew any statistics of the presence of variants in the population. Using a hash reference derived from the genome of the patient, which would identify each patient uniquely and thus allow recognition of the fact that the same patient is present at two centers, would alleviate this problem. However, it is unclear how such a hash reference could be derived from noisy genome information and also such an approach goes against

the idea of minimizing the amount of personal information shared because it provides a unique identifier for each patient. Our approach allows drilling down through the data to the lowest level and contacting PIs responsible for different patients and therefore eventually identifying that different records correspond to the same patient.

Finally, the main direction for further development of our system is that it currently does not support clinical information. Important technical issues make handling clinical information challenging. Clinical information is often managed and transferred separately from sequencing data. Researchers may often have sequencing data, but little clinical information beyond a few overly broad labels (for example, tumors versus controls), which makes query by others difficult. In practice, clinical information in genomic research projects is often managed in an unstandardized way without reliance on structured information, such as ontologies. Even if data are described in a structured way, there are currently only limited options to cover all clinical domains beyond Unified Medical Language System (UMLS) or Systematized NOmenclature of MEDicine Clinical Terms (SNOMED), which have important limitations. Extracting clinical information automatically from hospital information systems remains extremely challenging and will be done differently in each center. For these reasons, integrating clinical information into *NGS-Logistics* remained beyond the scope of this article. Developing a full clinical information management system is unrealistic and would be too cumbersome for users. A more realistic direction is to provide a simple way to annotate samples with either free text or, better, with Resource Description Framework (RDF) triples (or similar concepts) that allow annotation using properly referenced ontologies. Both functionalities should be present so as not to exclude users that can only provide free text annotation. Text search and SPARQL-like (SPARQL Protocol and RDF Query Language) querying would then allow identification of relevant samples based on clinical information. This will make it possible to carry out queries such as “which mutations are significantly more frequent in clinical population A versus clinical population B across multiple centers?”

The key concept in *NGS-Logistics* is to let users query SNVs across data sets in multiple centers. Users get a full description of the results for the samples to which they have authorized access, while getting relevant summary information for other samples. This summary information is sufficient to carry out key research tasks, such as identifying relevant mutations in a patient population. The footprint on personal genomic data from other patients is kept to a bare minimum. The frequency of mutation at a given position in the “Whole Population” is not personal data and thus not subject to restrictions. The ability to identify a specific pseudonymized sample carrying a specific mutation provides stricto sensu access to personal genomic data. However, the amount of information is kept minimal (one single base-pair position at a time), the identifier is pseudonymized (to avoid any possibility of direct identification), and all users and requests are tracked by the system. The interest of broadly allowing bona fide researchers to identify relevant samples for further research or to support clinical diagnosis fulfills the legitimacy principle of data privacy. The fact that users and requests are tracked at all times contributes to the transparency principle. Finally, the fact that information about samples beyond the authorized ones is kept to non-personal summary statistics, except for a single backlink through a pseudonymous identifier, contributes to the proportionality principle. As such, we believe our system represents a best effort to enforce a high level of data protection while providing a powerful solution for clinical genomic research and diagnosis support.

Chapter 4

GALAHAD

- Title:
  GALAHAD: a web server for drug effect analysis from gene expression
- Published:
  Nucleic Acids Research (NAR) 4 May 2015
- Publication link:
  http://nar.oxfordjournals.org/content/early/2015/05/04/nar.gkv436.full
- Authors:
  Griet Laenen [1,2,*], Amin Ardeshirdavani [1,2,*], Yves Moreau [1,2] and Lieven Thorrez.

  1. KU Leuven, Department of Electrical Engineering (ESAT), STADIUS Center for Dynamical Systems, Signal Processing and Data Analytics, Kasteelpark Arenberg 10, Leuven, 3001, Belgium
  2. iMinds Medical IT Department, Kasteelpark Arenberg 10, Leuven, 3001, Belgium
  3. KU Leuven, Interdisciplinary Research Facilit Life Sciences, Department of Development and Regeneration, Kortrijk, 8500, Belgium

  * These authors contributed equally to the paper as first authors.
- Author contribution:
  The PhD candidate designed, developed, implemented and maintains the *GALAHAD* web-application. The PhD candidate contributed to the validation of the software and to the manuscript.

4.1. Abstract

GALAHAD (https://galahad.esat.kuleuven.be) is a web-based application for analysis of drug effects. It provides an intuitive interface to be used by anybody interested in leveraging microarray data to gain insights into the pharmacological effects of a drug – mainly identification of candidate targets, elucidation of mode of action, and understanding of off-target effects. The core of *GALAHAD* is a network-based analysis method of gene expression. As an input, *GALAHAD* takes raw *Affymetrix* human microarray data from treatment versus control experiments and provides quality control and data exploration tools, as well as computation of differential expression. Alternatively, differential expression values can be uploaded directly. Using these differential expression values, drug target prioritization and both pathway and disease enrichment can be calculated and visualized. Drug target prioritization is based on the integration of the gene expression data with a functional protein association network. The web site is free and open to all and there is no login requirement.

4.2. Introduction

The pharmaceutical industry is facing unprecedented productivity challenges. Only one out of 20 compounds that entered clinical trials in 2006-2008 made it to become a marketed product [102, 103]. Causes of failure change during the course of development. Early in the process, compounds fail primarily for safety reasons. Compounds that successfully navigate Phase 1 increasingly drop out due to lack of efficacy in Phase 2 and 3 [104]. With safety and efficacy being the main bottlenecks, a better knowledge of a candidate drug's mode of action and its off-target effects could be of substantial value to drug development.

DNA microarray technology enables genome-wide analysis of the transcriptional response to a compound treatment, which is frequently employed to study the effects of small molecules on cells [105]. Gene expression patterns that describe the perturbation of a biological system by a drug compound can provide valuable information for identifying compound-protein interactions and resulting effects [106] prior to clinical trials. In addition, this information may also be useful for already marketed drugs, in the light of drug repositioning [107]. This approach has proven to be successful as demonstrated by the many applications of the *Connectivity Map* [108-111].

We have developed an easy-to-use web server called *GALAHAD*, for the in-depth exploration of a drug's mode of effect based on gene expression changes following treatment. Our software provides multiple tools needed for gaining new insights into the biological effects of a drug by combining *Affymetrix* human gene expression data preprocessing, quality assessment and exploratory analysis, genome-wide drug target prioritization, differential expression analysis and pathway, as well as disease phenotype enrichment. Drug target prioritization relies on the integration of the obtained differential expression values with prior knowledge on functional protein associations. By means of a network neighborhood analysis the functional relation between proteins is taken into account, which significantly improves the expression-based prediction of drug-protein interactions.

4.3. Input

The submission page is easy to find and help is available by mouse-over where input is required from the user. The main input for *GALAHAD* are raw DNA microarray data derived from both untreated control samples and samples treated with a drug of interest, with one of the three common *Affymetrix GeneChips* HG-U133A, HG-U133 Plus 2.0, or HuGene 1.0 ST. Both the control and the treatment data need to be uploaded as a zipped file containing ".CEL" files for at least two samples. The total size of both these zip files is currently limited to 250 MB. Once the data are uploaded completely, *GALAHAD* can be launched by clicking the submit button. A typical workflow then starts with the preprocessing of these uploaded expression data, either with the *RMA* [112] or the *MAS5.0* algorithm, as indicated by the user. Next, the desired analyses can be selected; these may include quality assessment and exploratory analysis of the data, differential expression analysis, drug target prioritization, and both pathway and disease enrichment. Drug target prioritization and enrichment analysis can also be performed on differential expression data derived from other platforms through the submission of a plain text file containing *HGNC* gene symbols, $\log_2$ ratios, and corresponding *p*-values. The input interface provides the possibility to enter a study name, as well as an email address to get notified when all computations are finished.

4.4. Methods

An overview of the different analysis steps is depicted in Figure 49.

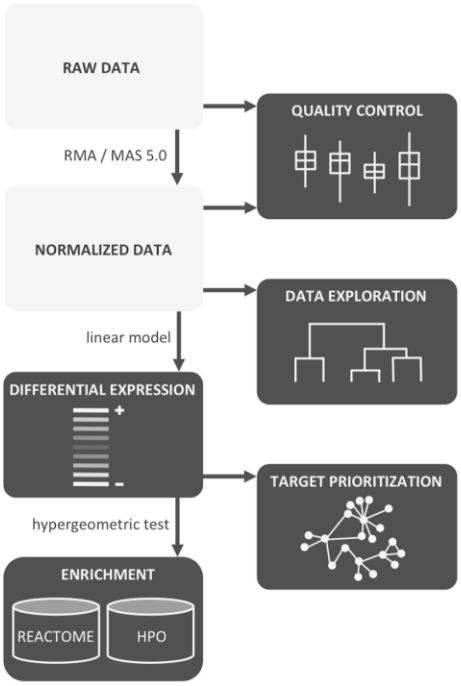


Figure 49: Overview of the different *GALAHAD* analysis steps. Quality control is performed on both the raw and MAS5.0 normalized data. After normalization some further exploratory analysis is performed and differential expression is computed. The obtained differential expression values are then used for both pathway and disease phenotype enrichment as well as for network-based drug target prioritization.

4.4.1. Quality control

To assess the quality of the different arrays within the microarray data set, uncover experimental problems, and help in deciding whether certain arrays need to be considered as outlying, several quality metrics can be generated. This quality control is mainly based on the *Bioconductor ArrayTools* package, but also relies on functionalities from the *affy* [113], *oligo* [114], *oligoClasses*, *genefilter*, *simpleaffy* [115], *affyPLM* [116], and *RColorBrewer* libraries. The following plots are produced: side-by-side boxplots of the log intensities to assess the overall signal comparability, probe level model-based pseudo-images to detect spatial biases and a NUSE and RLE plot to check for probe set homogeneity. In addition, for HG-U133A and HG-U133 Plus 2.0 arrays, the average background, the percentage of genes called present, and the intensities of the spike-in hybridization controls give insight into the hybridization quality. Scale factors provide information on signal comparability and 3'/5' ratios for b-actin and GAPDH that can be used to assess sample quality. Finally, an RNA degradation plot is generated for these array types.

4.4.2. Data exploration

To assess the correlation and grouping of samples three more plots are generated on the preprocessed data: a PCA plot, a hierarchical clustering and an intensity correlation heat map.

4.4.3. Differential expression

Differential expression analysis to determine the significance of gene up- and down-regulation following drug treatment is based on the *Bioconductor limma* [117] package. A linear model is fitted to the expression data for each gene and by computing a global variance estimator using empirical Bayes moderated *t*-statistics and corresponding *p*-values are obtained. These *p*-values are then corrected for multiple testing by the method of Benjamini and Hochberg [118], by controlling the false discovery rate (FDR).

4.4.4. Drug target prioritization

Genome-wide drug target prioritization is performed by means of an in-house developed algorithm for network neighborhood analysis [87] integrating the expression data with functional protein association information derived from *STRING* [119]. More specifically, this approach prioritizes potential targets by diffusing the differential expression signals to neighboring network nodes based on the confidence scores of these functional associations. *P*-values are obtained by random reassignment of the expression values to the network nodes and computation of the corresponding randomized scores for all genes. The method has now been validated on a new test set consisting of 35 publicly available *Affymetrix* gene expression data sets where treatment with a chemical compound was tested against a control. In *ChEMBL* [120], we identified all proteins interacting with an IC_{50} below 10 μM with one of these compounds, resulting in a total of 85 distinct targets. A table with the 35 GEO accession numbers, together with the drugs of treatment and their targets as derived from *ChEMBL*, are provided as supplementary file S1. Based on the positions of these known targets in the protein rankings obtained with our method for each of the interacting

compounds, an AUC of 87% was achieved (Figure 50). In comparison, ranking the genes based on differential expression data only resulted in an AUC of 60%.

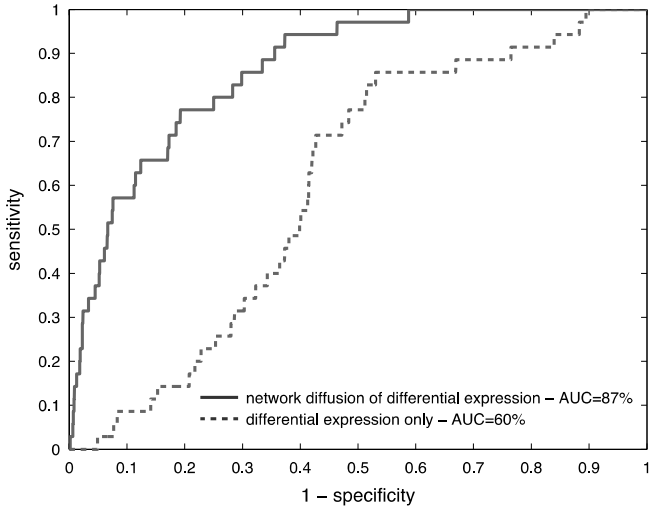


Figure 50: *GALAHAD* diffusion-based drug target prioritization performance. ROC curves comparing the performance of our network diffusion-based drug target prioritization algorithm and prioritization based on differential expression only on a new validation data set consisting of 35 drug treatments corresponding to 85 known targets.

4.4.5. Enrichment

Functional enrichment of differentially expressed genes is performed by means of a hypergeometric test, followed by an FDR-based multiple testing correction of the obtained *p*-values. In this way, *Reactome* [121] pathways involved in the drug's mode of effect are predicted, as well as associated disease phenotypes from the *Human Phenotype Ontology* (HPO) [122] enabling side effect prediction and drug repositioning.

4.5. Output

After submission, the user is automatically redirected to the results page, where the job details are provided. When the job is completed, the results appear on this page, and an email containing a hyperlink to the page is sent to the user if a valid email address was provided. If cookies are enabled, a hyperlink to the result page (study name may help in case of submitting several jobs) is also stored in a personal history for 20 days. The output is displayed in a series of tabs corresponding to the different analyses selected by the user.

4.5.1. Quality control and data exploration

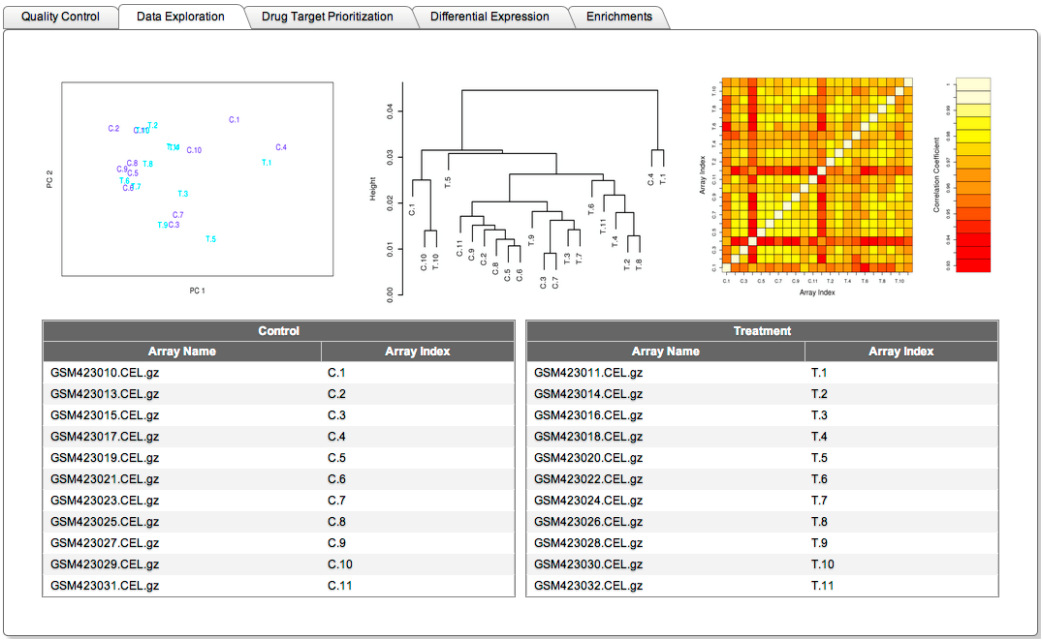
In the Quality Control and Data Explorations tabs (Figure 51, Figure 52), the different plots are displayed along with a table in which each sample is assigned a new index as used in the plots. By clicking on a plot a larger image is shown, together with a short explanation.



Control	
Array Name	**Array Index**
GSM423010.CEL.gz	C.1
GSM423013.CEL.gz	C.2
GSM423015.CEL.gz	C.3
GSM423017.CEL.gz	C.4
GSM423019.CEL.gz	C.5
GSM423021.CEL.gz	C.6
GSM423023.CEL.gz	C.7
GSM423025.CEL.gz	C.8
GSM423027.CEL.gz	C.9
GSM423029.CEL.gz	C.10
GSM423031.CEL.gz	C.11

Treatment	
Array Name	**Array Index**
GSM423011.CEL.gz	T.1
GSM423014.CEL.gz	T.2
GSM423016.CEL.gz	T.3
GSM423018.CEL.gz	T.4
GSM423020.CEL.gz	T.5
GSM423022.CEL.gz	T.6
GSM423024.CEL.gz	T.7
GSM423026.CEL.gz	T.8
GSM423028.CEL.gz	T.9
GSM423030.CEL.gz	T.10
GSM423032.CEL.gz	T.11

Figure 51: Quality control. In the Quality Control tab the different quality assessment plots are displayed along with a table in which each sample is assigned a new index as used in the plots. By clicking on a plot a larger image is shown, together with a short explanation.



Control	
Array Name	**Array Index**
GSM423010.CEL.gz	C.1
GSM423013.CEL.gz	C.2
GSM423015.CEL.gz	C.3
GSM423017.CEL.gz	C.4
GSM423019.CEL.gz	C.5
GSM423021.CEL.gz	C.6
GSM423023.CEL.gz	C.7
GSM423025.CEL.gz	C.8
GSM423027.CEL.gz	C.9
GSM423029.CEL.gz	C.10
GSM423031.CEL.gz	C.11

Treatment	
Array Name	**Array Index**
GSM423011.CEL.gz	T.1
GSM423014.CEL.gz	T.2
GSM423016.CEL.gz	T.3
GSM423018.CEL.gz	T.4
GSM423020.CEL.gz	T.5
GSM423022.CEL.gz	T.6
GSM423024.CEL.gz	T.7
GSM423026.CEL.gz	T.8
GSM423028.CEL.gz	T.9
GSM423030.CEL.gz	T.10
GSM423032.CEL.gz	T.11

Figure 52: Data exploration. The Data Exploration tab is constructed in exactly the same manner as the Quality Control tab.

4.5.2. Differential expression

The Differential Expression tab (Figure 53) contains a table listing all genes together with their $\log_2$ ratios and both the naive and adjusted *p*-values for differential expression. Genes are ranked according to the adjusted *p*-value, but can for example also be ranked according to $\log_2$ ratio by simply clicking on the respective column header. Furthermore, different filters can be applied. The complete table can be exported in comma delimited CSV format, which can be imported by any spreadsheet. By clicking on a gene name, the user is brought to the corresponding *GeneCards* section. To select the differentially expressed genes used for Enrichment Analysis, a *p*-value threshold can be entered at the top of the page. By default, an adjusted *p*-value of 0.01 is used.

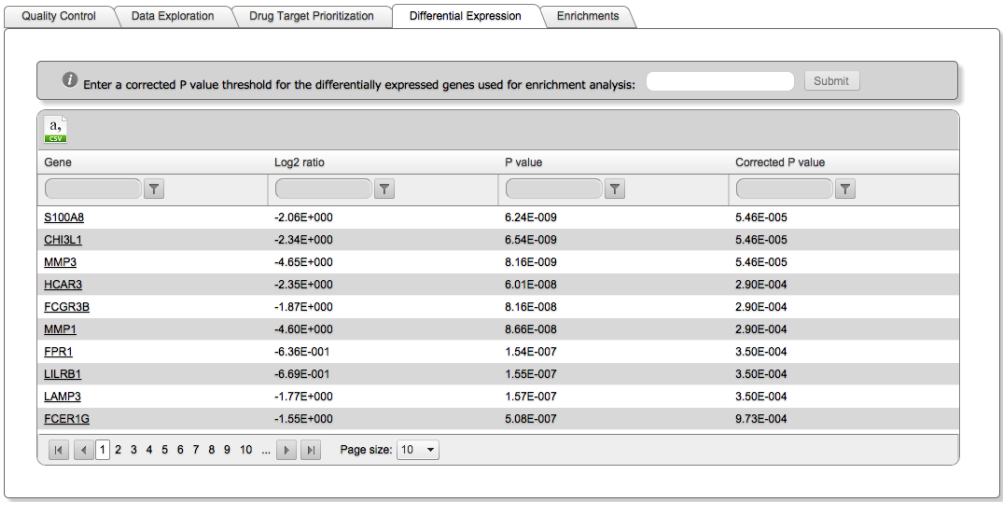


Figure 53: Differential expression.

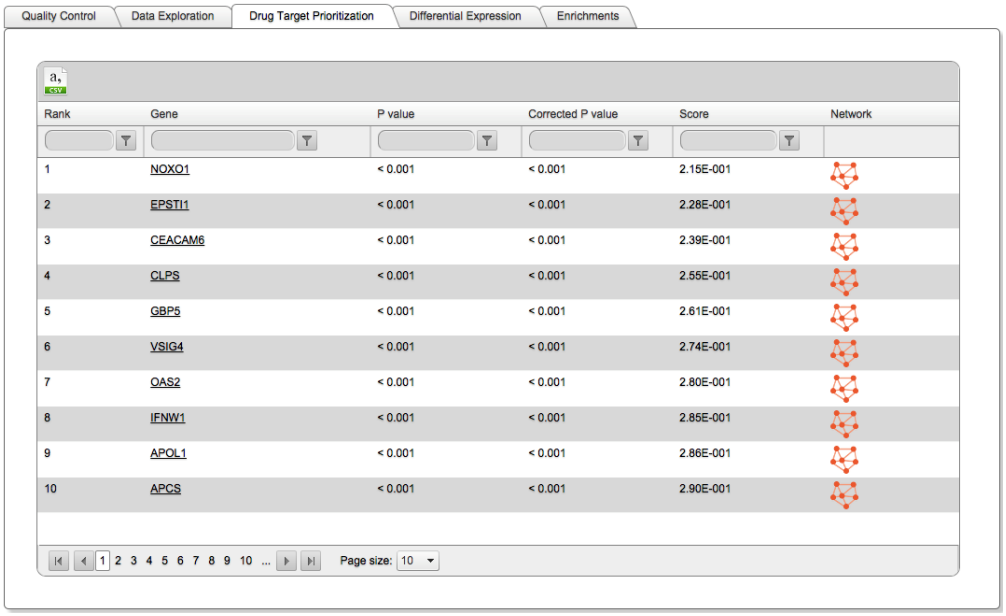


Figure 54: Drug target prioritization.

4.5.3. Drug target prioritization

In the Drug Target Prioritization tab (Figure 54), a ranked list of genes as potential targets of the drug can be found together with the network diffusion-based scores and both the naive and adjusted *p*-values for prioritization. Genes of interest can be detected in this list with the aid of the filter function. The complete table can again be exported as a CSV file and clicking on a gene name opens the corresponding *GeneCards* section. In addition, a network-based visualization is available for each gene, showing the ten interaction partners contributing most to the gene's ranking. Genes in this network are colored according to fold change and sized according to the significance (*t*-statistic) of this differential expression. Clicking on a node or edge opens the corresponding gene or interaction in *STRING* (Figure 55).

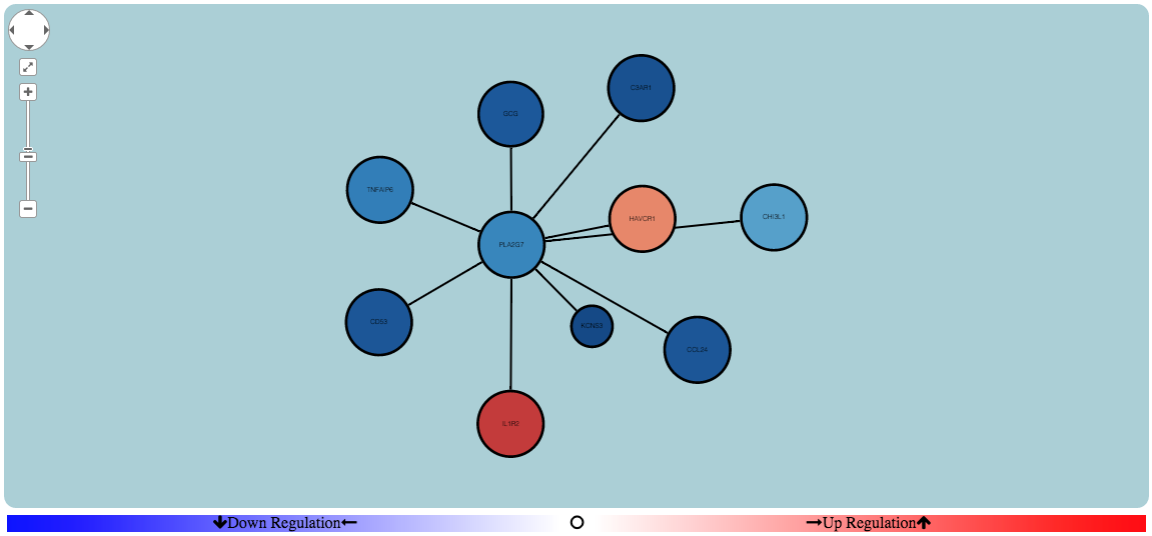


Figure 55: Drug-gene network.

4.5.4. Enrichment

After selecting the differential expression threshold on which the gene set used for enrichment analysis is based, two separate tabs display tables summarizing the enrichment results for Pathway and Disease Enrichment (Figure 56). These tables contain *Reactome* or HPO IDs and names, together with the number of differentially expressed genes in the corresponding gene sets and the accompanying *p*-values and adjusted *p*-values. The differentially expressed genes for each pathway or disease phenotype are accessible through dropdown tables. Clicking on an ID brings the user to the *Reactome* diagram or HPO page. The enrichment tables can again be sorted, filtered, and exported as CSV files. In addition, network graphs are available, consisting of the top ten most significant pathways or disease phenotypes, along with their associated genes colored according to fold change.

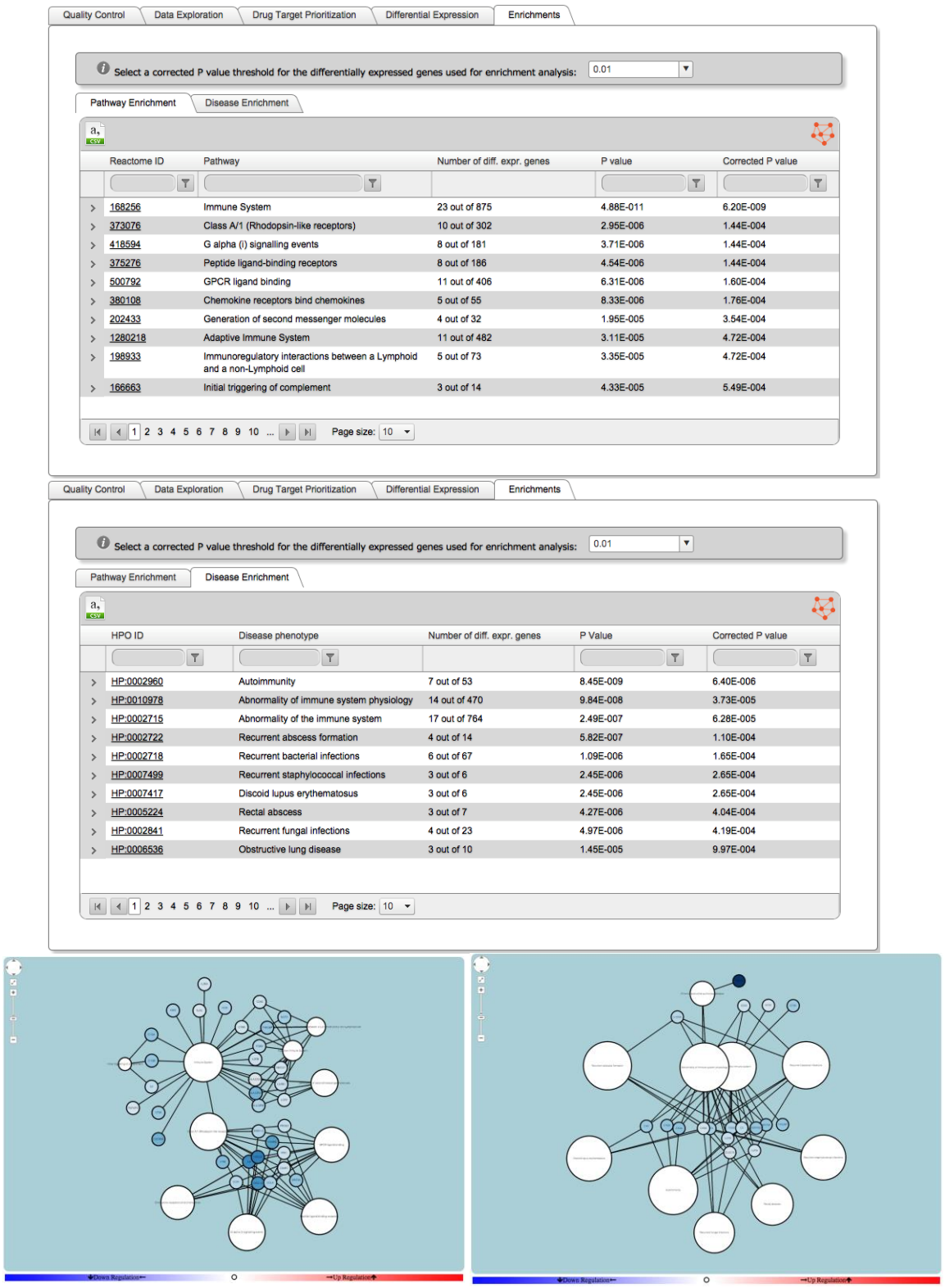


Figure 56: Enrichment tab: Pathway enrichment and disease enrichment.

4.6. Implementation

The *GALAHAD* server has three modules: a user interface, a job controller, and an analyzer. The user interface provides a database-related service based on a three-tier architecture using *Microsoft .Net* technology. At the presentation layer on top, we developed a user-friendly and rich interface. To speed up interaction between the clients and the server, we added *AJAX* functionalities to retrieve large amounts of data as fast as possible. All the procedures for importing data into and retrieving data from the database are located in the data layer. The business layer represents the main link between the presentation layer and the data layer and thus contains all the rules for processing information.

The job controller is a simple *Java* application responsible for communication between the web application and the analyzer. After submission of a new request the job controller transfers the uploaded control and treatment file to a *Linux* server and runs the analyzer to process the request. When the results are ready the job controller transfers them back to the web server and makes them available for the user.

The analyzer is the main module responsible for analysis of the user's request. This module is completely developed in R. The time needed for a job to finish is depending on the data size, the requested analyses, and the workload of our server. Data produced by the analyzer module are stored on a *Microsoft SQL* server. *GALAHAD* can be accessed with all main web browsers. Uploaded data and the results of analysis are kept private and not viewable by other users through the use of a secure server and are deleted after two weeks. The web site is free and open to all and there is no login requirement.

4.7. Example

An example data set is provided on the web site. Results can be directly accessed or input data are provided and can be uploaded. A manual and an example video are also available on the home page.

4.8. Conclusion and outlook

GALAHAD is an open and free platform to analyze microarray data for the characterization of drug effects. It provides an easy interface where in an integrated way quality control, data exploration, differential expression and both pathway and disease enrichment can be calculated and visualized. In addition, drug target prioritization provides a ranked list of target candidates.

Towards the future, the platform can be extended to take data from RNA sequencing experiments and more complicated experimental designs such as multiple groups and time series, as well as data from other organisms such as mouse and rat. We are also in the process of implementing the network analysis tools from *GALAHAD* as a standalone R package.

Chapter 5

SNIPT

- Title:
  Big Data solutions for variant discovery from low coverage sequencing data, by integration of Hadoop, HBase and Hive.
- Publication status:
  In preparation
- Authors:
  Amin Ardeshirdavani [1,2], Erika Souche [3], Jia Yu [4], Martijn Oldenhof [5], Joris Robert Vermeesch [3], Yves Moreau [1,2].

  1. KU Leuven, Department of Electrical Engineering (ESAT), STADIUS Center for Dynamical Systems, Signal Processing and Data Analytics, Kasteelpark Arenberg 10, Leuven, 3001, Belgium
  2. iMinds Medical IT Department, Kasteelpark Arenberg 10, Leuven, 3001, Belgium
  3. KU Leuven, Center of Human Genetics Gasthuisberg, O&N I Herestraat 49, Leuven, 3000, Belgium
  4. Bielefeld university, Faculty of technology, Int. Research Training Group, 33305 Bielefeld, Germany
  5. KU Leuven, Facilities for research, 3001 Leuven, Belgium
- Author contribution:
  The PhD candidate designed, developed, implemented, and maintains the application. The PhD candidate contributed to the data collection and analysis. The PhD candidate contributed to the setup and test of the infrastructure. Part of this project (Hadoop implementation) was carried out as the Master thesis of Jia Yu (Sep. 2014) as designed and supervised by Prof. Yves Moreau and the PhD candidate. The PhD candidate drafted the manuscript.

5.1. Abstract

Next-Generation Sequencing (NGS) allows the sequencing of the whole human genome sequencing (WGS) to, among other things, efficiently study human genetic disorders. However, the large amount sequencing data resulting from WGS requires large computational resources and optimized programming structures for effective data analysis. Many researchers use scale-out networks of commodity hardware as an alternative to classical supercomputers. In several use cases, Apache Hadoop and HBase have been used to respectively coordinate distributed computation and act as a storage platform. However, scale-out networks have rarely been used to handle gene variation data from NGS, except for sequencing read assembly. In our study, we propose a Big Data solution that integrates Apache Hadoop, HBase, and Hive to efficiently analyze NGS output, such as VCF files, a commonly used gene variation data file format for variant discovery.

5.2. Introduction

5.2.1. Variant discovery

Raw sequencing data has to be processed to extract reliable information. NGS data processing can be divided into two major steps: primary analysis (image analysis and base calling) and secondary analysis. When looking for variation in the human genome, secondary analysis consists of aligning/mapping the reads against the reference genome and scanning the alignment for variation. Variant calling procedures depend on the results from the mapping algorithm. In practice, different combinations of mapping algorithms and variant callers unfortunately lead to different variant calling results [123]. A core application of variant discovery is finding DNA sequence variant(s) that could be the cause for constitutional and acquired disorders [124]. One class of variants of interest is that of Single Nucleotide Variants (mutations that are rare in the general population) and Polymorphisms (mutations that are frequent in the general population) (SNP/SNV). SNPs and SNVs are stored in Variant Call Format (VCF) files, a file format introduced by the 1K Genome Project [99]. During the past ten years, researchers all around the world have introduced several variant discovery tools and methods, each one of them having distinct advantages and drawbacks [125]. As a result, today we have access to public databases, such as dbSNP or HGMD [126, 127] holding millions of records that report a variation and definitely or putatively associate them with human diseases. Each research center has either designed and developed its own pipeline of software packages to automate the whole process of the variant discovery or opted for the use of commercial software. The content of the analysis pipeline can vary from one center to another depending on the structure of the computing facilities and sequencing methods used at each place. Considering also the multiple quality control steps needed for each process, it is clear that the whole workflow needed to identify SNPs/SNVs and demonstrate their biological function and involvement in disease is highly complex. As a consequence, heavy computational requirements and long response times become a significant hurdle for researchers.

5.2.2. Sequencing coverage

There are several ways to categorize sequencing methods. One can group them based on the length of reads (*short* and *long*) and depending on the existence of a reference genome,

analysis of sequencing data can be divided into two major categories: mapping against a reference sequence and *de novo* assembly [128]. Raw files first need to be aligned to reconstruct the genome. The alignment enables the count of the number of reads covering each position. Coverage or depth is the average number of reads that provides sequencing data for a given nucleotide. There are different standards for the minimum coverage cutoff to call an allele as a reference base or a variant. Each type of application will require a specific minimum level of average coverage. For example, sequencing coverage recommended for detecting human genome mutations is 10x to 30x depth of coverage [129]. In reality this coverage is not uniform across all positions. Some regions might have higher coverage, while others have lower coverage. To identify rare variants, we will need to have access to sufficiently high coverage data. By contrast, there are analyses for which lower coverage sequencing are sufficient, such as some studies of complex trait genetics or the building of a reference panel [130].

5.2.3. Integration of Apache Hive and HBase

As described in the previous section, Hive and HBase have different characteristics. While HBase is ideal for low latency random access, Hive is built for petabyte scale data. It is not ideal for small data sets or low latency queries. Integrating those two tools together allows benefiting from the advantages of both platforms and has been made possible by contributions from Facebook around 2010 [90].

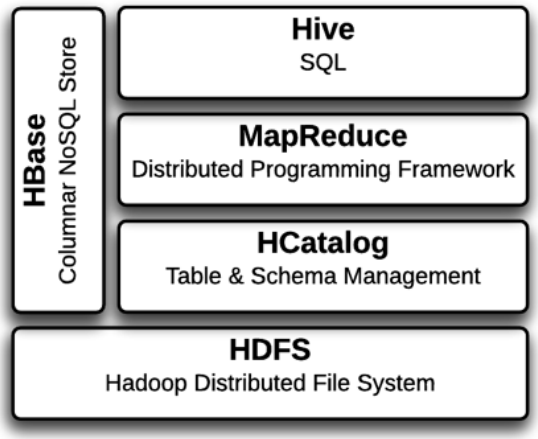


Figure 57: Integration Hive – HBase. Hadoop distributed file system is the lowest level of the system, actual data stored on this level. HBase is used to store data as the columnar format. Tables and schema models are managed at the HCatalog layer. MapReduce layer is responsible for the actual computation part. Hive allow users to query data through SQL syntax.

With the integration (Figure 57), Hive can create an external table as a linkage to the real HBase data table. The operation provides users with several advantages. First, the data is physically separated from all the data manipulation and process. All the operations in Hive, are Hive local, only with the linkage to HBase. Thus the original data in HBase remains untouched. For this reason, the data does not risk to be damaged of any process. Second, Hive queries are all translated into MapReduce jobs explicitly. Thus users do not have to worry about coding MapReduce jobs; instead, they only need to focus on constructing their query statement. Although simple query can be implemented in HBase shell to avoid programming, HBase, as a NoSQL model, still demands users to learn extra commands to implement their operations. Hive is thus more user friendly since it supports well known

and easy to understand SQL syntax completely. This makes driving the whole computer cluster and analyzing data in HBase by Hadoop MapReduce process possible for researchers with intermediate level of IT skills.

5.3. Data use case

Our platform focuses on gene variant analysis. Technically, our application is aimed at parsing and storing VCF files. The application is designed to dynamically adapt to different kinds of VCF files since their structure can diverge significantly. For gene variant analysis, the VCF files of different samples have to be queried and it should be possible to export the results for further downstream analyses. A VCF file for a single sample or a group of samples can be large, so the efficiency of processing is crucial. In our study, we used samples sequenced for Non-Invasive Prenatal Testing (NIPT) from a Belgian population. Our dataset consists of 200 NIPT cases sequenced on an Illumina sequencer. NIPT was based on low coverage (0.1x) whole-genome sequencing of maternal plasma DNA. Low-coverage whole-genome sequencing of maternal plasma DNA was initially used to detect common trisomies in pregnancies. In this work, by contrast, we will use this data to create a reference set of gene variants for the Belgian population.

5.4. Aims

Our goal is to propose an efficient and easy to handle model for parallel processing of VCF files, together with an application that implements this model to solve bioinformatics problems. The new model should meet certain requirements: it should be scalable, fault tolerant, and dependable. Also, data import should be fast and data should occupy the smallest storage possible, be secure, and be easy to query, even remotely. To achieve these requirements, the open source projects Apache Hadoop, HBase, and Hive are integrated into a model and application for efficient NGS data analysis. Our application can thus efficiently deal with different types of VCF files and allows us to easily generate the Belgian variation reference database and keep it up-to-date. Based on its performance, we believe that this application will be helpful to many bioinformaticians.

5.5. Methods

To facilitate and automate data processing, we set up an analysis pipeline. This pipeline was designed specifically to extract and analyze possible mutations. To extract mutations from BAM files, we used the FreeBayes variant caller. Since we are dealing with low coverage data, we needed a variant caller that handles haploid nucleotides, as there is no promise that the sequencer sequences both copies of a single nucleotide. Sequencing and primary analysis such as alignment are all done at the UZ Leuven computational cluster. To do our analysis, we set up a small Hadoop cluster and transferred data to this cluster. The cluster consists of three nodes: one master node and two slave nodes. The whole pipeline is made up out of several steps, as illustrated in Figure 58. One of the key advantages of our pipeline is that we can add samples at any time. Furthermore, there is no hard restriction on the size of the data we can handle.

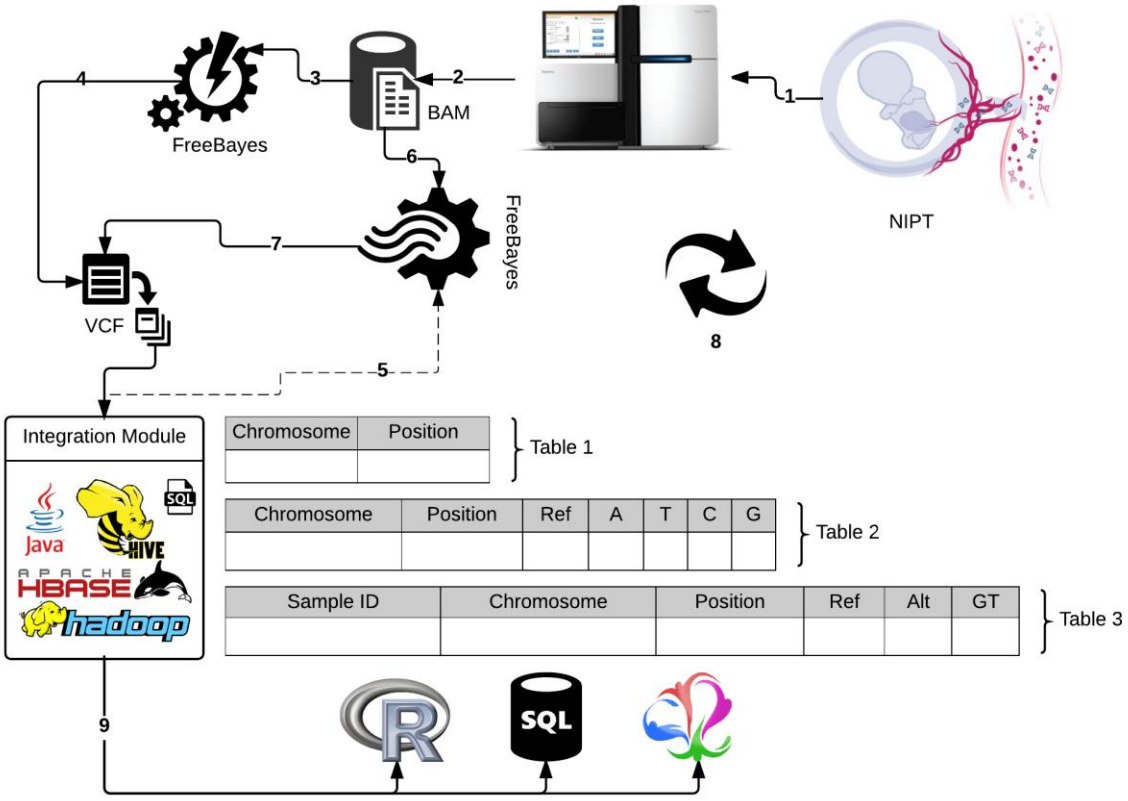


Figure 58: NIPT variant calling pipeline. Our NIPT variant calling pipeline consists of eight steps. It starts with collecting a DNA sample from mother, which is then sequenced by an Illumina sequencer. The raw data is mapped to the reference genome and converted to BAM format. Variants are called with FreeBayes. VCF files are then stored and aggregated in the HBase database to create a unique list of variants that will be used as the reference. Then a second variant calling is performed with Freebayes. Once the second set of VCF files has been generated, we can calculate actual frequency of the observed variation in the population.

The NIPT variant calling pipeline steps are

1. Samples are collected and after preparation sent to the Illumina sequencer.
2. Sequencing results are processed and stored as BAM files on the storage devices.
3. Variant calling analysis is run on each sample with FreeBayes.
4. The result of FreeBayes is transferred to our integration model in VCF format, where in the first phase only Chromosome and Positions are extracted and a unique list of chromosome positions are stored in Table 1.
5. After all new samples are processed and new records added to Table 1, we use this list as reference positions and call variants among all the samples based on this reference. In this way, we can be sure that all the samples are investigated for a unique list of positions and not only based on the variant caller parameters.
6. FreeBayes is run again over all samples and collects the information related to the input reference positions.
7. VCF files are collected and transferred to the Hadoop cluster. In this step, two types of information are extracted from the data. First, we store the chromosome, position, reference base, and the number of alleles appearing in that position in Table 2. With this information, we calculate the minor allele frequency score for that position among all the samples. This score is then used to decide whether a mutation is seen at that position in each sample. Second, we also use MAF scoring

to calculate the genotype of the alleles per position and sample, and store this information in Table 3.

8. As already mentioned, we do not need to rerun the pipeline over all the samples each time a sample is added. Instead, we can simply add samples one by one to the system. When a new sample is sequenced, we can add the sample to the system and all the information regarding the positions existing in the sample will be updated in Table 1. For those positions appearing in the system for the first time, we run the pipeline on all the samples to extract the information regarding the new positions and update the whole database.
9. We can query the data that we need by writing a simple SQL command, which is handled by Hive. We can use the extracted data to run an analysis on the data in R, or we can store this data in a relational database, and use it in another application. We can also use this information in *NGS-Logistics* in the format of population variant frequency and filter variations that we are interested in based on their frequency in the NIPT dataset.

5.6. Results and discussion

At the time of writing of this thesis, the whole study is not finished yet and we cannot provide a final result. However, we did a partial study on the Chromosome 21 of 200 samples, taking into account that the Hadoop cluster to which we have access is a small setup with three nodes. The characteristics of the data are

Number of samples	Total Size of VCFs	Number of unique positions
200	110 GB	33,256,600

From the size of the test dataset, it is clear that we need an efficient solution to store this amount of data. A first potential solution is to focus on an efficient file storage system. Using this format, files are stored as they are generated and researchers analyze data by means of scripting languages or some other programming languages to extract the data they need out of the text files. A second potential solution is to use a relational database. As the SQL database model is still the most popular data management solution and SQL syntax is easy to learn and understand, there are still a lot of researchers using an SQL database to tackle VCF files. Finally, the third potential solution is our integration model, which combines through the use of Hadoop/HBase/Hive the SQL model with the distributed file system and processing.

5.6.1. Performance and comparison

We benchmarked our platform to check whether our integration model can handle VCF files more efficiently and execute queries more quickly than the other solutions. Since three solutions needed to be tested, we used three different environments to run them separately. VCF files were stored on the Leuven University Hospitals cluster. For SQL, we used Microsoft SQL server (MSSQL) as database. We used the standalone version of MSSQL, which means that we only needed a single PC for database deployment, located on the Leuven University Hospitals database servers. Finally, the environment for implementing our integrated solution needed to be a fully distributed PC network. To start a new local PC cluster and deploy all Apache projects, we used the Flemish Supercomputer Center

(VSC) infrastructure. As mentioned before, our setup consists of three nodes: one master node and two slave nodes. We ran each test five times and recorded the performance. We then averaged the performances for each single test and compare the two solutions in Table 8 and Table 9. For both types of tests, we executed the following query:

```
Select*From TblVCFs where che=1 and cov>20
```

Table 8: Performance comparison for three solutions - 1 sample.

	Text Base	MSSQL	Apache Integration
Melting	NO	YES	NO
Data load into database (Second)	0	600	900
Query Time (Second)	30	5	20

Table 9: Performance comparison for three solutions - 200 samples

	Text Base	MSSQL	Apache Integration
Melting	NO	YES	NO
Data load into database (HOUR)	0	6.5 - NA	1.5
Query Time (Second)	350	NA	200

Table 8 shows the performances for the three solutions when tackling a single VCF file. To use MSSQL to manage a VCF file, a melting procedure is needed, meaning that the data must be converted into a relational database data structure. By contrast, for Apache integration and text base format, the data loading is free of melting. One can argue that we can import data into relational databases exactly as they look, but queries will be much slower if we need to query something based on the text format of that record. We can tell from the table that using MSSQL the data loading took an extra approximate 600 seconds, as to melt a 700MB VCF file. However, Apache lost superiority during the actual data loading. With respect to our results, the text-based format is the real winner in a one sample test.

For the second test, we randomly selected 200 samples and repeated the exact same procedure as used for our previous test. The total file size was approximately 110GB. Because of this size, melting the files into the data table took around 6.5 hours and to only import part of the data, after which the MSSQL server engine crashed. We tried other solutions, such as breaking down the data into chunks and then import them, which took a lot of time as well. Loading the data into HBase took much less time and without any issue all the data was imported into the database. We should also mention here that HBase has special data zipping algorithms that helped us to save much more disc storage than anticipated. As the results show, this time the integration model wins the competition on query time. We should also take into account the fact that the text-based procedures run on much faster computing recourses (AVALOK) than our Hadoop cluster.

The tests show clearly that the Apache integration performance is significantly better than SQL and the text-based model when dealing with large amounts of data. Also, for small datasets, the integration performance is acceptable. So, we conclude that Apache integration could be a good solution for this kind of file management. It is fairly safe to assume that with more cluster nodes, the performance of the integration and especially the query efficiency can be further improved (Figure 59).



Figure 59: Hadoop cluster performance. By adding more machines to the Hadoop cluster, we can increase the performance if we need to analyze more data.

5.6.2. System power and results

A rich interface was developed and installed on top of the integration model to help users easily interact with the system. Users can create tables, define fields, import data, query and export results through a user-friendly interface. Since the application is running on a Hadoop cluster, it was convenient to implement a MapReduce process to quickly parse a large VCF file. The mapper function reads the file row by row. The reducer function continuously collects the results of the mapper and sorts them. After the mapper parses the last row of the file, a single parsed result is generated by the reducer at the same time and written into a HBase table. HBase gives several ways of loading data into tables, for instance the Put command or normal APIs, but the most efficient way is to implement bulk loading. It uses a MapReduce job to translate table data into HBase's internal data format, and then directly loads the generated StoreFiles into the HBase table. Using bulk loading will use less computational power than simply using the HBase APIs. To implement bulk loading, we simply alter the output format of the parsing MapReduce process to match the HBase schema, a combination of the row key, column family, column qualifier if applicable, and the value. After the MapReduce process, the bulk loading process only needs a few lines of code. Because we only use Hive to query the database and extract data, we just designed a one-tab pane for the Query menu (Figure 60). The Hive-Query pane will provide the user with all the linkages in Hive. By default, the query field contains `SELECT * from tableName`. This is a simple SQL query asking the database to show all the data in a table. There is a listener in the linkage list. The `tableName` in the query will change once the user chooses another link in this list. If the user submits this query, the data linked in HBase will show up in table form. The user can clear the query field to write a custom query. As long as the syntax is SQL, the query can be submitted and translated into a MapReduce process on the cluster. We also provide the functionality of exporting the query result. The user can specify the output directory and a file name to save the results in a tab delimited text file.

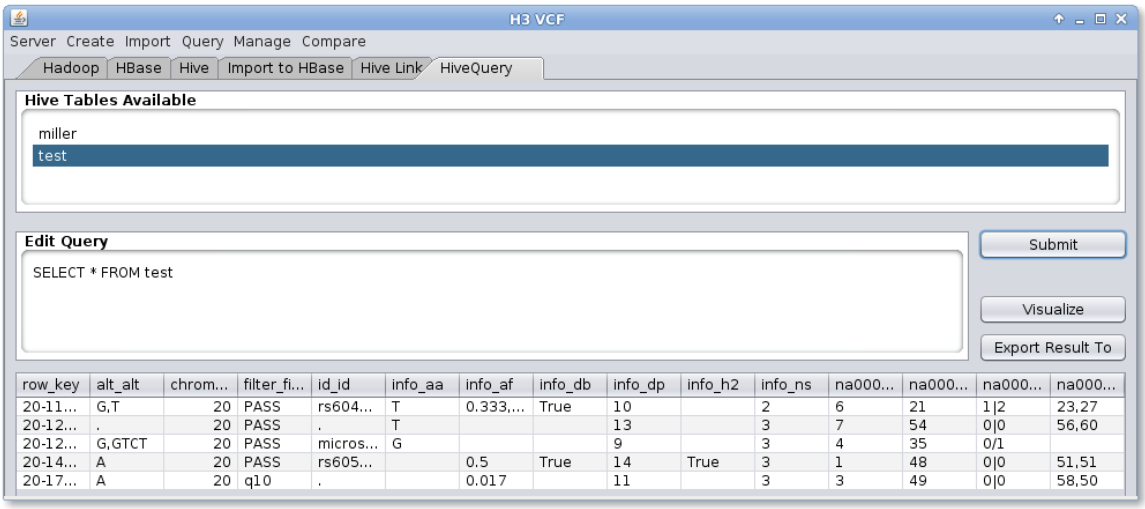


Figure 60: Query pane: Interactive interface on top of the integration model. In the query pane, we also provide the user with a simple visualization. Users can select the query result table to visualize basic statistics. We transfer these results to R so that users can apply any visualization algorithm to the data.

The user can for example select all the positions on chromosome 21 where the sequencing coverage is higher than 50 and then transfer the results to R to draw a histogram of position coverage (Figure 61).

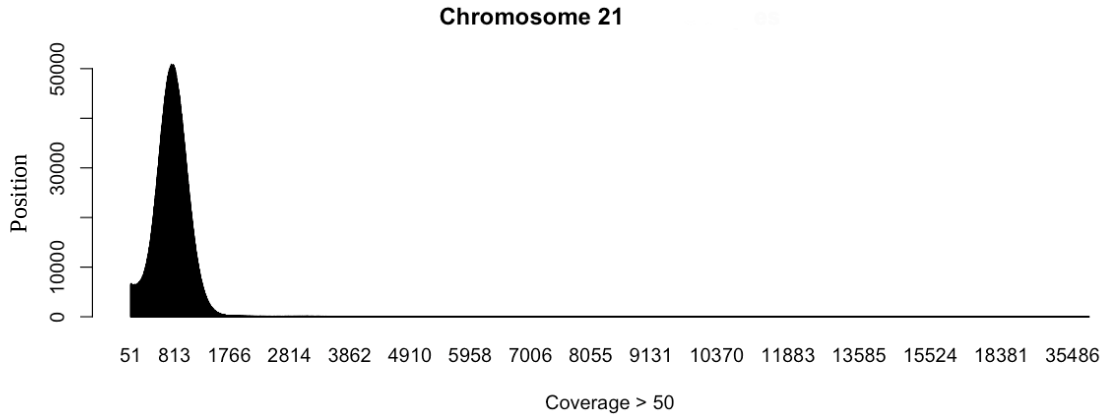


Figure 61: Position vs. coverage relation. Aggregation of the variants of 200 samples shows which regions are well covered by the sequencing and we can use those regions to calculate the frequency of mutations.

This newly developed application with a user-friendly interface is an effective tool for users with limited IT knowledge. With it they can conveniently integrate data across VCF files. Users can either choose to build their own local computer cluster or use Amazon Elastic MapReduce to easily create a cluster with Apache projects. To the best of our knowledge, there are hardly any examples of using Apache integration to analyze with VCF files. We hope our study and application can be a convenient tool for researchers in genomic variation research.

5.7. Future work

Apache Hadoop and MapReduce have been released for ten years. Use cases have changed over time. As more and more users are confronted with big datasets, there are increasing demands for analytics of data streams and recursive computation. Considering that MapReduce needs initialization at each round of computation, and that Hadoop is a batch processing model, efficiency is severely reduced in such situations. To overcome this drawback and still make use of a scale-out network, computer scientists have proposed a new computational framework named Spark. It processes data streams much more efficiently than Hadoop and MapReduce. Spark also supports HBase data storage. It is compatible with Hive and also provides its own tool called Shark to execute SQL queries. It is likely to be a good alternative to Hadoop when the user needs a heavy data streaming process or recursive computations, for instance in the case of machine learning and data mining. Our study shows that the Apache integration performs better when dealing with large volumes and complex structured data. The developed application merely offers researchers a convenient method to utilize this integration. Together they are just an analytic infrastructure. However, researchers are also interested in the useful information hidden behind the data. In NGS and the analysis of genomic variation, special mining methods or algorithms are needed for deeper analysis. Currently this is still lacking from our application. In the future, we can incorporate some useful and common NGS and variation analysis methods in our application. A significant advantage is that the Apache integration can easily be combined with statistical analysis in R, which is a popular biostatistics and bioinformatics tool. R also has a lot of open source packages to extend its use cases. The popularity and the excellent performances of R made it our first choice for incorporation into our application. We will continuously maintain and improve this application, and add more functionalities. We hope this application can help researchers deal with NGS data, especially VCF data files, more easily, so that they can focus more on using the data to answer questions instead of putting too much effort on data management.

Chapter 6

Endeavour

- Title:
  Candidate gene prioritization with Endeavour
- Published:
  Nucleic Acids Research (NAR), 30 April 2016
- Publication link:
  http://nar.oxfordjournals.org/content/early/2016/04/30/nar.gkw365.full
- Authors:
  Léon-Charles Tranchevent [1, *], Amin Ardeshirdavani [2,3, *], Sarah ElShal [2,3], Daniel Alcaide [2,3], Jan Aerts [2,3], Didier Auboeuf [1], and Yves Moreau [2,3].

  1. INSERM U1210, CNRS UMR5239, Laboratoire de Biologie et de Modélisation de la Cellule, Ecole Normale Supérieure de Lyon, Université de Lyon, Lyon, France.
  2. KU Leuven, Department of Electrical Engineering (ESAT), STADIUS Center for Dynamical Systems, Signal Processing and Data Analytics, Kasteelpark Arenberg 10, Leuven, 3001, Belgium
  3. iMinds Medical IT Department, Kasteelpark Arenberg 10, Leuven, 3001, Belgium

  *. These authors contributed equally to the paper as first authors.
- Author contribution:
  The PhD candidate designed, developed, implemented, and maintains the *Endeavour* web application. The PhD candidate contributed to the validation of the software. The PhD candidate contributed to the data visualization. The PhD candidate contributed to the manuscript

6.1. Abstract

Genomic studies and high-throughput experiments often produce large lists of candidate genes among which only a small fraction are truly relevant to the disease, phenotype or biological process of interest. Gene prioritization tackles this problem by ranking candidate genes by profiling candidates across multiple genomic data sources and integrating this heterogeneous information into a global ranking. We describe an extended version of our gene prioritization method, Endeavour, now available for six species and integrating 75 data sources. The performance (Area Under the Curve) of Endeavour on cross-validation benchmarks using 'gold standard' gene sets varies from 88% (for human phenotypes) to 95% (for worm gene function). In addition, we have also validated our approach using a time-stamped benchmark derived from the Human Phenotype Ontology, which provides a setting close to prospective validation. With this benchmark, using 3854 novel gene–phenotype associations, we observe a performance of 82%. Altogether, our results indicate that this extended version of Endeavour efficiently prioritizes candidate genes. The Endeavour web server is freely available at https://endeavour.esat.kuleuven.be/.

6.2. Introduction

Biologists often use a combination of high-throughput methods, to produce large-scale data and generate hypotheses, and of low-throughput methods, to experimentally validate these hypotheses and create biological knowledge. One challenge in current biology is the gap between the large amount of genomic data that are being generated and the pace at which novel knowledge is created from it. This problem is particularly conspicuous in medical genetics, with many human complex traits and Mendelian disorders remaining unexplained despite the availability of huge amounts of genome-scale data. In this situation, computational biology aims at reducing this gap by proposing *in silico* methods that analyze these data to derive hypotheses that can be validated experimentally.

An example in medical genetics is the identification of the genomic factors underlying human Mendelian disorders. Indeed, high-throughput experiments often produce large lists of candidate genes among which only a few are truly relevant to the disease or phenotype under study. It is not always possible to experimentally validate all candidate genes individually, and so there is the need to prioritize these candidate genes as to maximize the efficiency of the downstream validation experiments. More precisely, the principle is to identify the most promising candidate genes and filter out the ones that appear of limited relevance, and then to investigate these promising candidate genes more thoroughly. The study of complex disorders raises a similar problem since there are many genes involved, each gene having a small effect. The objective is therefore to identify these genes among the numerous candidate genes. Several computational methods have been proposed to tackle this prioritization problem and they have been used in practice to help identify disease-causing genes [131, 132].

Some of these methods rely on the guilt-by-association concept as to favor candidate genes that resemble the most what we already know about the disease or phenotype of interest [133, 134]. For instance, if a candidate gene has a protein product that directly physically interacts with a protein that is already known to be involved in the phenotype of interest, then the candidate gene appears promising for this phenotype. This can be extended by considering multiple sets of data beside protein interaction networks, which is often what

prioritization methods do. For instance, the *ToppGene* suite combines an interaction network with functional annotations to favor the candidate genes that interact with the seed genes (which defined the disease of interest) and also have similar functional annotations [135]. Similarly, *GeneDistiller* uses gene-phenotype associations, gene expression patterns and protein–protein interactions among other data to propose candidate gene prioritization [136]. There exist other methods that do not rely on seed genes, and are therefore more suited for diseases of unknown etiology. These methods rely on disease relevant keywords, or on complementary disease specific experimental data sets. For instance, the *Ge´nie* web server ranks genes using a text-mining approach that is fed with user-selected keywords and can be complemented with orthology information [137]. Alternatively, *MetaRanker* integrates multiple disease specific data sets, including genetic association data, copy number variation data and differential expression profiles [138].

In this article, we introduce a much-improved version of *Endeavour*, our gene prioritization method, which is avail- able for six species-now also including zebrafish. *Endeavour* currently integrates 75 data sources that represent different and complementary views about the genes, which is more than twice as many as our previous version [139]. In addition, we have validated *Endeavour* using a time-stamped benchmark, in which predictions are made and only benchmarked later on when enough new data have accumulated (similar to the CAFA strategy [140]).

Other researchers have used *Endeavour* to decipher the mechanisms underlying several human diseases. For instance, Yu *et al.* used *Endeavour* to prioritize candidate genes extracted from a whole-exome sequencing of familial cases of congenital diaphragmatic hernia [141]. After validation, *GATA4* (ranked 3rd) was associated with congenital diaphragmatic hernia. Zielinski *et al.* studied hemifacial microsomia and identified a 1.3 Mb duplication on chromosome 14q22 by analyzing five affected individuals [142]. Their final result suggests a role for *OTX2* (ranked 1st) in human craniofacial development.

6.3. Endeavour methodology

Users can prepare a prioritization in four simple stages (see Figure 62). In the first stage, the species with which to work is selected among the six available species (*H. sapiens*, *M. musculus*, *R. norvegicus*, *D. melanogaster*, *C. elegans*, and *D. rerio*). Then, the set of seed genes (or training genes) is prepared in the second stage. These are the genes that are already associated with the process of interest (*e.g.*, genes already associated with congenital heart defects). Typically, users would like to have at least 5 but no more than 40 seed genes, but larger seed sets can still provide reliable results. The third stage consists in selecting the data sources to use for the prioritization. Users are advised to select the resources that best suit their specific problem and to avoid using conjointly redundant resources. The list of usable data sources depends on the selected species. There is no limit in the number of data sources to select, besides its impact on computing time. Finally, the set of candidate genes is defined in the last stage. The hypothesis is that at least one of these candidate genes is associated with the biological process of interest, but this association is yet unknown. This set can for example be derived from previous experiment (*e.g.*, genes from a deletion frequently observed in patients with congenital heart defects). However, there is no size limit regarding this set, so users who have no *a priori* regarding relevant candidate genes can instead prioritize the whole genome. The single output is an ordered list of the candidate genes with on top the most promising candidate genes. In an ideal

situation, the gene yet unknown associated with the process of interest should be ranked first or at least among the top ranking genes. Each candidate gene is also given a p-value that represents the significance of this combination of rankings. In addition, rankings for each individual data source are also available as to better understand the global ranking (*e.g.*, to identify the sources that contributed the most to prioritize a given gene).

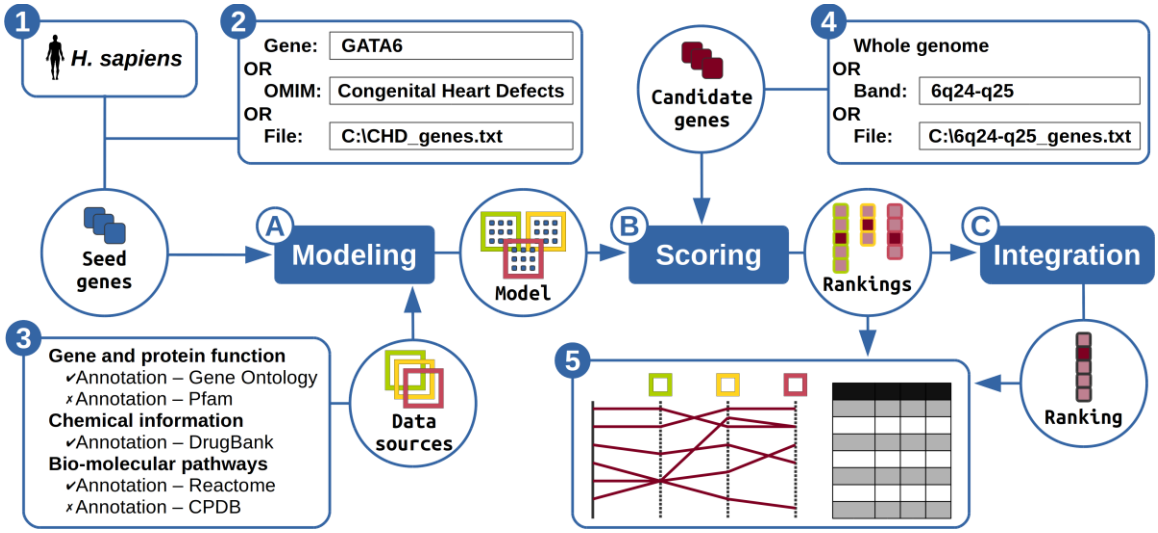
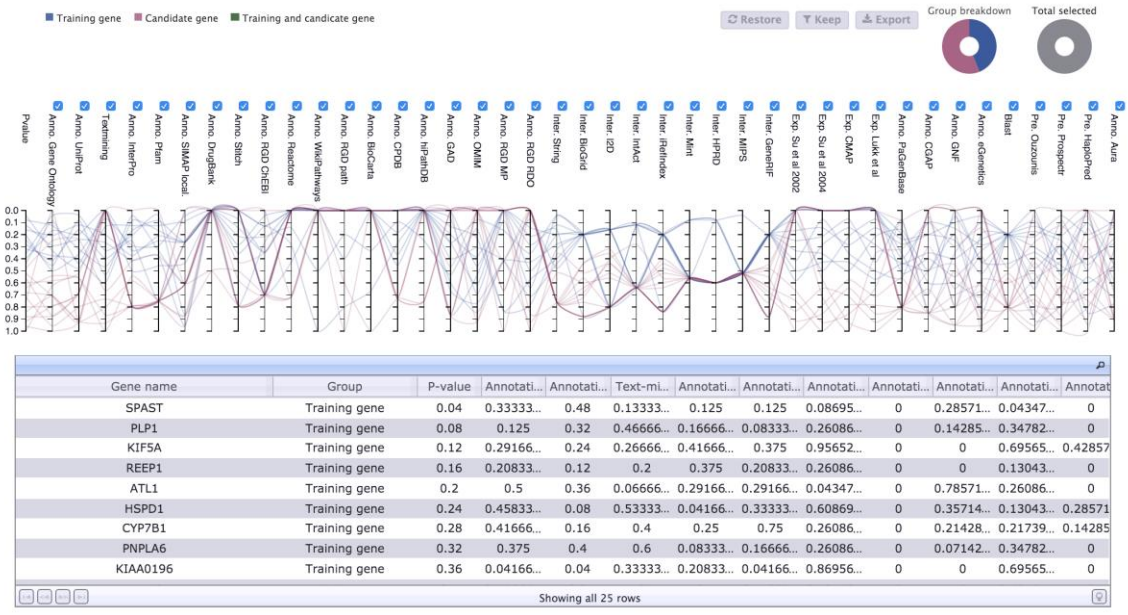


Figure 62: The Endeavour algorithm. Users can start a prioritization by (1) selecting the species of interest, (2) defining which genes are known to be associated with the process of interest, (3) selecting the data sources to be used in the process, and (4) providing the candidate genes to prioritize. Endeavour then (A) uses the seed genes to build a model of the process of interest, (B) scores the candidate genes with this model to produce several rankings, and (C) integrate these rankings into one global ranking, which (5) is returned to the user through the web server.



Gene name	Group	P-value	Annotati...	Annotati...	Text-mi...	Annotati...	Annotati...	Annotati...	Annotati...	Annotati...	Annotati...	Annotat
SPAST	Training gene	0.04	0.33333...	0.48	0.13333...	0.125	0.125	0.08695...	0	0.28571...	0.04347...	0
PLP1	Training gene	0.08	0.125	0.32	0.46666...	0.16666...	0.08333...	0.26086...	0	0.14285...	0.34782...	0
KIF5A	Training gene	0.12	0.29166...	0.24	0.26666...	0.41666...	0.375	0.95652...	0	0	0.69565...	0.42857
REEP1	Training gene	0.16	0.20833...	0.12	0.2	0.375	0.20833...	0.26086...	0	0	0.13043...	0
ATL1	Training gene	0.2	0.5	0.36	0.06666...	0.29166...	0.29166...	0.04347...	0	0.78571...	0.26086...	0
HSPD1	Training gene	0.24	0.45833...	0.08	0.53333...	0.04166...	0.33333...	0.60869...	0	0.35714...	0.13043...	0.28571
CYP7B1	Training gene	0.28	0.41666...	0.16	0.4	0.25	0.75	0.26086...	0	0.21428...	0.21739...	0.14285
PNPLA6	Training gene	0.32	0.375	0.4	0.6	0.08333...	0.16666...	0.26086...	0	0.07142...	0.34782...	0
KIAA0196	Training gene	0.36	0.04166...	0.04	0.33333...	0.20833...	0.04166...	0.86956...	0	0	0.69565...	0

Figure 63: Endeavour result page.

The algorithm behind *Endeavour* prioritizes genes in three simple steps (see Figure 62). In the first step, it trains a model of the biological process of interest, using the seed genes provided by the user. The model contains one sub-model per user-selected data source and is trained using simple statistics. For instance, for annotation-based data sources (*e.g.*, Gene Ontology) only the features over-represented within the set of seed genes are kept in the sub-model. In the second step, the candidate genes are scored using the model built in the first step. More precisely, for a given data source, the scores are computed using the associated sub-model, and represent how well the candidate genes fit this sub-model. For instance, for annotation-based data sources, the score of a candidate gene is the Fisher's omnibus combination of the p-values of its annotations that are present in the sub-model (*i.e.*, annotations not present in the sub-model are ignored). At the end of this step, each data source is associated with a ranking of the candidate genes, with the most promising candidate genes on top. In the third step, these rankings, which correspond to prioritizations made using different data sources, are then integrated using order statistics to obtain a single global ranking [143]. This method allows candidate genes with little data (*e.g.*, poorly annotated genes) to be fairly compared with candidate genes that have a lot more data. The algorithm then outputs a ranked list of candidate genes, with p-values that represent the significance of this ranking. In addition, the prioritization results are now displayed graphically using a parallel coordinate representation (Figure 63) so that users can easily check which data sources contributed the most to the global ranking.

Endeavour integrates 75 data sources for 6 species that can be classified into broad categories that describe what we know about genes. They are briefly described below, and a detailed list of all available databases is available in Supplementary Material 1. The "Gene and protein function" category includes resources such as Gene Ontology [144] and InterPro [145], which are usually organized as annotations between ontologies (that describe function in a broad sense) and biological entities (*i.e.*, genes or gene products). In addition, the category "Biomolecular pathways" includes pathway databases such as Reactome [146], which are complementary to the purely ontological annotations described above. Several resources that describe gene or protein interaction networks, such as BioGrid [147] and IntAct [148], are also integrated and classified into the "Interaction networks" category. Chemical data sets are also integrated in *Endeavour* within the "Chemical information" category. These contain annotations between gene or gene products and other entities such as drugs. One example is the DrugBank database [149]. The "Phenotypic information" category gathers all resources that collect associations between genes or gene products and phenotypes or diseases, possibly using model organism data. Examples are the OMIM database for human Mendelian disorders [150], and the Rat Disease Ontology from the Rat Genome Database [151]. Expression data are split into two categories depending on whether complete expression profiles are available (category "Expression profiles"), such as large expression data sets, or whether the data have already been summarized into annotations, such as within PaGenBase [152] (category "Expression ontologies"). The last category "Sequence-based features" contains all resources that are based on gene, transcript, or protein sequences. This includes protein sequence similarities computed using BLAST or predicted miRNA regulation using transcript sequences [153, 154].

The web server is based on a three-tier architecture, developed using Microsoft .Net technology and Microsoft SQL database. The core *Endeavour* library is written in Perl and

the data is stored in a MySQL database. We have introduced a parallel coordinate representation to visualize our multidimensional numerical results. This gives the users the opportunity to easily recognize patterns in the data and identify possible correlation among the different sources. Uploaded data and prioritization results are kept private and not viewable by other users, and are in any case deleted after 30 days. The web site is free and open to all and there is no login requirement. Two examples derived from the literature are available for users who simply want to try out *Endeavour*. In addition, a manual describes how to run a simple prioritization and a help page contains hints on how to solve the more frequent issues.

6.4. Evaluation results

For the first validation, we have performed a leave-one-out cross-validation on "gold standard" gene sets. These sets contain genes already associated with phenotypes or biological processes and have been extracted from references databases, such as OMIM and Gene Ontology. These sets are mostly derived from studies on Mendelian disorders that represent the majority of our current knowledge. However, we have also included a benchmark data set extracted from the Genetic Association Database, which focuses on complex disorders, since these are the focus of most of the studies nowadays. Briefly, for each iteration of the cross-validation, the prioritization is done by training *Endeavour* with all genes from the "gold standard" set except one (hereafter called the left-out gene), and the aim is to rank this left-out gene together with 99 randomly selected genes. We expect this left-out gene to rank towards the top since it comes from the "gold standard" set. All data sources except the ones that contain explicitly the association we want to predict are used in the process (*i.e.*, OMIM is not used for any phenotype-based "gold standard"). In the end, an Area Under the ROC curve (AUC) is computed for each "gold standard" set, and AUC values are averaged over the different sets to derive a global estimate. These are then compared to control AUC values obtained with randomly built gene sets. A summary of the results is presented in Table 10. The full results, including ROC curves, are available in Supplementary Materials 2 and 3. We observe that performance varies from 88.34% (for human phenotypes) to 94.93% (for worm gene function), which indicates that Endeavour methodology is performing as expected. In addition, our benchmark reveals that the performance on complex disorders (GAD, 88.96%) is similar to the performance on pure Mendelian disorders (OMIM, 93.41%). Based on the GAD benchmark, we also observe that the average running time for the prioritization of 100 human genes is 311 seconds (using 34 data sources).

However, cross-validation is often not the best method to estimate actual performance (because leakage of information across multiple sources can lead to overly optimistic results), and time-stamped strategies are often preferable [155].

Table 10: Results of the leave-one-out cross-validation on "gold standard" gene sets. Results of the leave-one-out cross-validation on "gold standard" gene sets. For each benchmark (row), the columns contain the number of gene sets, the number of genes, the AUC, and the control AUC respectively.

Species	Source	Nb sets	Nb genes	AUC	Ctrl AUC
H. sapiens	HPO	1,553	19,386	91.72%	49.79%
	OMIM	29	611	93.41%	48.43%
	GO	4,526	55,950	92.26%	50.04%
M. musculus	RGD-RDO	672	8,413	88.61%	49.59%
	GO	4,379	53,105	90,46%	49.75%
R. norvegicus	RGD-RDO	652	7,997	90.55%	49.21%
	GO	4,140	49,895	88.68%	49.24%
D. melanogaster	FlyBase	1,612	17,395	91.38%	50.12%
	GO	2,371	28,834	89.88%	49.93%
C. elegans	WormBase	225	2,304	94.93%	50.87%
	GO	1,400	17,060	92.05%	49.95%
D. rerio	Zfin	135	1,662	88.70%	49.37%
	GO	1,856	22,476	90.76%	49.16%

We have therefore defined a time-stamped benchmark that resembles the CAFA strategy [140]. The principle is that predictions have been made in January 2013 and only benchmarked in December 2015 to avoid data contamination (*i.e.*, data used for both training and testing, even if indirectly). Our benchmark relies on annotations between human genes and phenotypic terms from the Human Phenotype Ontology (HPO). More precisely, *Endeavour* was first used to make predictions for a wide range of HPO terms, using the whole genome and only data from January 2013 or before. We have then collected from HPO the 3,854 phenotype-gene associations that have been discovered between January 2013 (build 44) and December 2015 (build 102), and used these associations to benchmark the *Endeavour* predictions from the first step. The AUC over the 3,854 novel associations is 81.51%, which indicates that *Endeavour* is able to efficiently prioritize disease candidate genes in a realistic setup (details are in Supplementary Material 4). However, this AUC value is lower than the estimate obtained by the cross-validation, therefore confirming the suspicion that cross-validation on a "gold standard" most likely provides an overestimation of the actual performance.

6.5. Conclusion

Endeavour is a web server that performs gene prioritization for six species (*H. sapiens*, *M. musculus*, *R. norvegicus*, *D. melanogaster*, *C. elegans* and *D. rerio*). It is available through a web-based user interface. *Endeavour* has been validated using a standard cross-validation scheme, as well as a time-stamped benchmark. Altogether, we provide a server that supports researchers in deciphering the molecular basis underlying diseases and phenotypes.

Chapter 7

Conclusions

During the course of the past few years, we have been facing floods of data brought by state-of-the-art technologies in all industries with a direct effect on our lives. In particular, the emergence of NGS has revolutionized every aspect of life sciences, posing important computational and ethical questions, such as

- Which data should be stored and when?
- How will the data be stored and what are the requirements to efficiently store it?
- What is the best way to analyze and use the data?

Therefore, it is important to describe clearly the data in the first place and further think of its future uses and its place in the data ecosystem as a whole. For instance, large genomic data cannot be used for genetic diagnosis without a thorough understanding of the origin, nature, and implications of that particular data. To this end, sophisticated methods are needed for storing, retrieving and analyzing the data. Furthermore, the privacy of the patient data should always be taken into account. It is easy when a hospital only has to grant access to a couple of data scientists, but when we want to create fairly open systems accessible to a large and diverse group of researchers, securing the data becomes a real challenge. Moreover, because of the diversity in data structures produced in healthcare by the vast variety of sources and machines (*e.g.*, different sequencers), it is important to set up a well-organized data storage system and further analyze this data with robust algorithmic workflows.

In this thesis, we have explored multiple aspects of NGS data analysis to identify genomic variations across the human genome. We have also considered other omics applications, such as gene prioritization and drug-target interaction prediction. We also proposed different models to design and develop applications with complex data structures and procedural workflows.

Over the past decade, improvements in genome sequencing methods and technologies resulted in increasing sequencing speed and reduced cost. However, the use of these sequencing technologies is often challenged by errors and biases associated with the computational methods used for data analysis. In particular, the use of naïve methods to identify single nucleotides changes in the human genome and identification of genotypes can influence downstream analysis. These factors motivated us to develop a bioinformatics pipeline by making an ensemble of different tools. We proposed an optimal variant calling pipeline. Inputs and outputs of every step, as well as suggested tools, were explored. We explained that based on the goal of a specific study different tools and parameters can be applied. Specifically, these computational tools and their input parameters can be tuned based on the nature of the data and the scientific question of a specific study (*e.g.*, the input parameters of an algorithmic pipeline should be tailored if the genomic data is derived from a single cell or a lot of cells, from genome or exome, *etc*.) To this end, we performed a

comparison between three most famous variant callers and demonstrated how choosing a right tool can affect the output information. As described in the Material and Methods chapter, producing and analyzing information in a single genomic center is just the beginning. We need to have access to other databases to refine and interpret the final results. This allows us to create and publish clear and trustworthy results. Importantly, we also showed that it is critical to control and define the access level of users on the data, with the primary goal of minimizing the risk of misuse of the data by unauthorized users. Another important driver for data protection is the local and international regulations on electronic data. These regulations often define what information must be retained, for how long and under what conditions. These rules and regulations are different in different countries. For example, some of the rules established by the European Union (at the time of writing this thesis) limit data sharing between the United States and European countries. The main target of this rule is to balance individual's privacy with national security obligations. It means that big companies, such Google, Facebook, Amazon and many others may not be allowed to transfer sensitive personal data to outside of the European Union. Policy makers from both sides are working on a protocol to enhance this rule so that data can still be used. However, the privacy of people must be strictly protected. Therefore, there is a need for an acceptable solution to protect the data. To address this, we designed an access control list application with a user-friendly interface, which allows us to hide all the complexity that we are facing in managing this access controls. *Bio-ACL* is an open source Java application embedded in *NGS-Logistics*, and which controls user access to datasets and samples shared through the system.

Developing a software project is a complex task. Managing complexity is at the heart of every software project. As described, complexity affects the timeline of a project and as a consequence its budget. We have learned that several issues can cause the complexity, such as displaying features differently for a specific audience; interrelating elements where changes in one part of the system cause changes in the other; integrating with external data sources and/or software. Einstein is often quoted as having said that "everything should be made as simple as possible, but no simpler". Therefore we have also learned that software complexity is not something that can and should be avoided at any cost, but rather that it may make a piece of software unique and useful. Thus a piece of software of well-controlled complexity can become popular and help your project be successful. In Chapter 2, we elaborated on the most significant complexities faced in this work, and proposed ways to handle them. As a result, we designed and developed successfully several complex applications (Table 11).

Table 11: System use report.

Tool name	Project Start Date	Google analytics: Unique site visit (IP)	Users	Requests
NGS-Logistics	January 2014	7,530	125	2,656
GALAHAD	January 2015	4,440	10	54
Endeavour	January 2016	2,835	372	1,360

The success of every project is highly dependent on a good understanding of application's performance (*i.e.*, developers should be well aware of the complexity of their data). The complex nature of data is indicative of the level of difficulty, which a developer will face when trying to translate it into the structure of the application. We noted that the data might be considered complex when it is "big" and "coming from disparate sources". Dealing with

larger amounts of data poses challenges regarding computational resources needed to process these massive datasets. In addition, multiple data sources can often mean messy data, or simply multiple datasets that follow different internal logics or structures. Data must, therefore, be transformed, or consolidated to ensure that all sources "speak the same language". In this thesis, we explained how we should recognize data complexity and overcome the accompanying challenges. As a result, we demonstrated how to combine data from different centers in *NGS-Logistics*. In addition, we showed that by combining data with different structures we could help researchers identify relations between protein targets and drugs in *GALAHAD*. Furthermore, we described an approach to prioritize genes by integrating data from different sources in *ENDEAVOUR* and proposed an integrated model to store and analyze large amounts of heterogeneous data, which would be challenging when tackled by conventional techniques. On top of all these complex data analytics applications, we designed and developed user-friendly interfaces to hide the complexity behind these structures. We divided user interfaces into two groups: The Input and Output groups. In the input interface, users set the parameters and upload their data. Input forms are designed in a way that allows the users to submit their request with a limited number of inputs or choices. On the computation side, we used parallelizing methods to reduce the computational time. We also showed how to use relational databases to store data and allow users to check the results interactively. Regarding outputs, we used different visualization techniques to present the results in the most useful way possible.

We explained that our ability to conveniently store, exchange, and analyze NGS data are some of the main challenges that come together in bioinformatics. Organizations and institutes need to plan carefully and take the necessary actions to tackle these challenges. Moreover, the analysis of NGS data also raises important challenges as to how to reconcile the federated analysis of personal genomic data and confidentiality of the data to protect the privacy of individuals. In many situations, the analysis of data from a single study alone will be far less powerful than if it can be correlated with other studies. In particular, when investigating a mutation of interest, it is extremely useful to obtain data about other patients or controls sharing similar mutations. However, genome data is sensitive personal data. Therefore, confidentiality of this data must be guaranteed at all times and only duly authorized researchers should access such personal data. To address all challenges described above, we developed the *NGS-Logistics* platform, which fulfills all requirements for a successful application that can process data inclusively and comprehensively from multiple sources while guaranteeing privacy and security. *NGS-Logistics* has been installed in KU Leuven, UZ Leuven, CMG Antwerp, CHU Liege, UZ Brussels, UCL, and UZ Ghent. Therefore, it has made it possible to create a Belgian Human Genome Network and through *NGS-Logistics* users have access to most of the samples sequenced within Belgium. Since *NGS-Logistics* has been developed as open source project, collaborators in Japan also use our code and methods to create their genomic data sharing network. There are related initiatives around the world, such as the Global Alliance for Genomic Health (GA4GH). They have setup an international project, the Beacon project, for genomic data sharing, and convinced prominent institutes such as EMBL and NIH to join this initiative. However, currently, the data sharing enabled by the Beacon project is rather limited, as they only provide information about the existence or absence of a certain mutation in different centers (while *NGS-Logistics* provide detailed statistics and allows to trace back relevant samples). In addition, we believe that there are important limitations to their

current mechanism, such as cruder quality control (by handling unnormalized VCF files rather than handling BAM files), SNPs being queried from a pregenerated list, and users being unable to query SNPs according to their own parameters and filters. The most significant limitation is that each center uses their own pipeline with different tools and versions, and there is no guaranty that the aggregated results can be fit together. We have tackled all these limitations in *NGS-Logistics* and our current implementation is still significantly ahead of what is available through the Beacon project.

Any piece of computer software always has its own advantages and disadvantages. Our tools are no exception and we are aware that more work can be done to develop more advanced tools. Since the heart of this thesis focused on analyzing NGS data and sharing this data, we discuss here some of the weaknesses of the *NGS-Logistics* and make suggestions for future work:

- Currently, *NGS-Logistics* call variants with a GATK pipeline using BAM files. This means that if a center has *n* samples, we run GATK *n* times. To call variants, GATK needs to load the reference genome and if we need some annotations, it also needs to load dbSNP into memory as well. If the center has more samples, the turnaround time for processing is larger. To solve this issue, we will change our pipeline and use the multi-sample calling option of GATK. In this way, we will only run GATK once and we expect processing time to be faster.
- As explained before, to obtain a standardized result across all the centers, we currently call variants from the BAM files at every query. Variant calling is time consuming and users must thus wait to obtain their results. We will add to the next version of *NGS-Logistics* a primary option that allows users to query pre-built VCF files. This option will allow users to query large numbers of samples across multiple centers in a matter of seconds (at the cost of reduced flexibility in variant calling).
- Another limitation of the current version is the lack of metadata. Currently, we only offer the users variants genotypes. By adding more metadata, in particular phenotypes, we can support richer, more informative queries.
- For the moment, we only allow registered users to use the system. By implementing and applying privacy-preserving algorithm to the system, we will be able to answer some high-level queries without the need for user registration.
- Installation of the *NGS-Logistics* query manager at different centers can sometimes be challenging because of differences in hardware and network infrastructure. By using the Docker system, we have the possibility to easily install the package at any location and minimize installation efforts.

In conclusion, it is important to make wise technological choices at the right moment for the right reasons. When it comes to software development, by combining different software engineering techniques, we can support the development of cost-effective software systems that in return provide a valuable and reliable platform for any kind of application.

Bibliography

1. Rojas R, Hashagen U: **The first computers : history and architectures**. Cambridge, Mass.: MIT Press; 2000.
2. Glyn Williams DPD-G: **Informatics the same old problems, just bigger**. *Drug Discovery World* 2013, **15**:28-32.
3. Bourne PE, Bonazzi V, Dunn M, Green ED, Guyer M, Komatsoulis G, Larkin J, Russell B: **The NIH Big Data to Knowledge (BD2K) initiative**. *J Am Med Inform Assoc* 2015, **22**(6):1114.
4. Nadal-Burgues N, Bonet E: **Limitations of project management when managing research: an empirical study at the Barcelona Supercomputing Center**. *The International Journal of Management* 2012, **1**(2):1-20.
5. Zhu H: **Software design methodology: From principles to architectural styles**: Butterworth-Heinemann; 2005.
6. Patterson DA, Hennessy JL: **Computer organization and design: the hardware/software interface**: Newnes; 2013.
7. Einav L, Levin JD: **The data revolution and economic analysis**. In.: National Bureau of Economic Research; 2013.
8. De Mauro A, Greco M, Grimaldi M: **What is big data? A consensual definition and a review of key research topics**. In: *AIP Conference Proceedings: 2015*. 97-104.
9. Voelkerding KV, Dames SA, Durtschi JD: **Next-generation sequencing: from basic research to diagnostics**. *Clin Chem* 2009, **55**(4):641-658.
10. Morozova O, Marra MA: **Applications of next-generation sequencing technologies in functional genomics**. *Genomics* 2008, **92**(5):255-264.
11. Wandelt S, Rheinländer A, Bux M, Thalheim L, Haldemann B, Leser U: **Data management challenges in next generation sequencing**. *Datenbank-Spektrum* 2012, **12**(3):161-171.
12. Richter BG, Sexton DP: **Managing and analyzing next-generation sequence data**. *PLoS Comput Biol* 2009, **5**(6):e1000369.
13. Primrose SB, Twyman R: **Principles of gene manipulation and genomics**: John Wiley & Sons; 2013.
14. Kim S, Tang H, Mardis ER: **Genome sequencing technology and algorithms**: Artech House, Inc.; 2007.
15. Janitz M: **Next-generation genome sequencing: towards personalized medicine**: John Wiley & Sons; 2011.
16. Metzker ML: **Sequencing technologies - the next generation**. *Nat Rev Genet* 2010, **11**(1):31-46.
17. **DNA Sequencing Costs: Data from the NHGRI Genome Sequencing Program (GSP)** [http://www.genome.gov/sequencingcosts/]
18. **Next Generation Genomics: World Map of High-throughput Sequencers** [http://omicsmaps.com/]

19. Genomes Project C, Abecasis GR, Auton A, Brooks LD, DePristo MA, Durbin RM, Handsaker RE, Kang HM, Marth GT, McVean GA: **An integrated map of genetic variation from 1,092 human genomes**. *Nature* 2012, **491**(7422):56-65.
20. **The 100,000 Genomes Project** [http://www.genomicsengland.co.uk/the-100000-genomes-project/]
21. Boomsma DI, Wijmenga C, Slagboom EP, Swertz MA, Karssen LC, Abdellaoui A, Ye K, Guryev V, Vermaat M, van Dijk F *et al*: **The Genome of the Netherlands: design, and project goals**. *Eur J Hum Genet* 2014, **22**(2):221-227.
22. **Human genome: Genomes by the thousand**. *Nature* 2010, **467**(7319):1026-1027.
23. Buchholz W: **Planning a computer system: Project Stretch**. 1962.
24. Mersha TB, Abebe T: **Self-reported race/ethnicity in the age of genomic research: its potential impact on understanding health disparities**. *Hum Genomics* 2015, **9**:1.
25. Sachidanandam R, Weissman D, Schmidt SC, Kakol JM, Stein LD, Marth G, Sherry S, Mullikin JC, Mortimore BJ, Willey DL: **A map of human genome sequence variation containing 1.42 million single nucleotide polymorphisms**. *Nature* 2001, **409**(6822):928-933.
26. Cargill M, Altshuler D, Ireland J, Sklar P, Ardlie K, Patil N, Lane CR, Lim EP, Kalyanaraman N, Nemesh J: **Characterization of single-nucleotide polymorphisms in coding regions of human genes**. *Nature genetics* 1999, **22**(3):231-238.
27. Ramensky V, Bork P, Sunyaev S: **Human non - synonymous SNPs: server and survey**. *Nucleic acids research* 2002, **30**(17):3894-3900.
28. Ng PC, Henikoff S: **SIFT: Predicting amino acid changes that affect protein function**. *Nucleic acids research* 2003, **31**(13):3812-3814.
29. Ardeshirdavani A, Souche E, Dehaspe L, Van Houdt J, Vermeesch JR, Moreau Y: **NGS-Logistics: federated analysis of NGS sequence variants across multiple locations**. *Genome Med* 2014, **6**(9):71.
30. Curcin V, Ghanem M: **Scientific workflow systems-can one size fit all?** In: *Biomedical Engineering Conference, 2008 CIBEC 2008 Cairo International: 2008*. IEEE: 1-9.
31. Ruparelia NB: **Software development lifecycle models**. *ACM SIGSOFT Software Engineering Notes* 2010, **35**(3):8-13.
32. Food U, Administration D: **Guidance for Industry: Electronic Source Data in Clinical Investigations**. *Washington, DC: US Department of Health and Human Services* 2013.
33. Kahn SD: **On the future of genomic data**. *Science* 2011, **331**(6018):728-729.
34. Boudreau T, Tulach J, Wielenga G: **Rich client programming: plugging into the netbeans™ platform**: Prentice Hall Press; 2007.
35. Weber-Lehmann J, Schilling E, Gradl G, Richter DC, Wiehler J, Rolf B: **Finding the needle in the haystack: differentiating "identical" twins in paternity testing and forensics by ultra-deep next generation sequencing**. *Forensic Sci Int Genet* 2014, **9**:42-46.
36. **Choosing the Right NGS Sequencing Instrument for Your Study** [https://genohub.com/ngs-instrument-guide/]

37. Consultation GF: **Coverage and Read Depth Recommendations by Sequencing Application**.
38. **An Introduction to Next-Generation Sequensing Technology** [http://www.illumina.com/content/dam/illumina-marketing/documents/products/illumina_sequencing_introduction.pdf]
39. Ziller MJ, Hansen KD, Meissner A, Aryee MJ: **Coverage recommendations for methylation analysis by whole-genome bisulfite sequencing**. *Nat Methods* 2015, **12**(3):230-232, 231 p following 232.
40. Cock PJ, Fields CJ, Goto N, Heuer ML, Rice PM: **The Sanger FASTQ file format for sequences with quality scores, and the Solexa/Illumina FASTQ variants**. *Nucleic Acids Res* 2010, **38**(6):1767-1771.
41. Ewing B, Hillier L, Wendl MC, Green P: **Base-calling of automated sequencer traces usingPhred. I. Accuracy assessment**. *Genome research* 1998, **8**(3):175-185.
42. **Guide to Storage and Computation Requirements** [http://www.strand-ngs.com/support/ngs-data-storage-requirements]
43. **GATK Best Practices** [https://www.broadinstitute.org/gatk/guide/bp_step.php?p=3]
44. Vahdat A, Al-Fares M, Farrington N, Mysore RN, Porter G, Radhakrishnan S: **Scale-out networking in the data center**. *Ieee Micro* 2010(4):29-41.
45. Rehm HL, Bale SJ, Bayrak-Toydemir P, Berg JS, Brown KK, Deignan JL, Friez MJ, Funke BH, Hegde MR, Lyon E *et al*: **ACMG clinical laboratory standards for next-generation sequencing**. *Genet Med* 2013, **15**(9):733-747.
46. Del Fabbro C, Scalabrin S, Morgante M, Giorgi FM: **An extensive evaluation of read trimming effects on Illumina NGS data analysis**. *PLoS One* 2013, **8**(12):e85024.
47. Andrews S: **FastQC: A quality control tool for high throughput sequence data**. *Reference Source* 2010.
48. Schmieder R, Edwards R: **Quality control and preprocessing of metagenomic datasets**. *Bioinformatics* 2011, **27**(6):863-864.
49. Gordon A, Hannon G: **Fastx-toolkit. FASTQ/A short-reads pre-processing tools**. *Unpublished Available online at:* http://hannonlab/ *cshl edu/fastx_toolkit* 2010.
50. **FastQC, Good and Bad Illumina Data** [http://www.bioinformatics.babraham.ac.uk/projects/fastqc/]
51. Baker M: **De novo genome assembly: what every biologist should know**. *Nature methods* 2012, **9**(4):333.
52. Martin JA, Wang Z: **Next-generation transcriptome assembly**. *Nature Reviews Genetics* 2011, **12**(10):671-682.
53. Flicek P, Birney E: **Sense from sequence reads: methods for alignment and assembly**. *Nature methods* 2009, **6**:S6-S12.
54. Langmead B, Salzberg SL: **Fast gapped-read alignment with Bowtie 2**. *Nature methods* 2012, **9**(4):357-359.
55. Li H, Durbin R: **Fast and accurate short read alignment with Burrows–Wheeler transform**. *Bioinformatics* 2009, **25**(14):1754-1760.

56. Wang W, Wei Z, Lam T-W, Wang J: **Next generation sequencing has lower sequence coverage and poorer SNP-detection capability in the regulatory regions**. *Scientific reports* 2011, **1**.
57. Group SBFSW: **Sequence alignment/map format specification**. *Available at ht tps://github com/samtools/hts-specs* 2013.
58. Schneider V, Church D: **Genome reference consortium**. 2013.
59. Mazza T, Castellana S: **Multi-Sided compression performance assessment of ABI SOLiD WES data**. *Algorithms* 2013, **6**(2):309-318.
60. Bonfield JK, Mahoney MV: **Compression of FASTQ and SAM format sequencing data**. *PLoS One* 2013, **8**(3):e59190.
61. Li H, Handsaker B, Wysoker A, Fennell T, Ruan J, Homer N, Marth G, Abecasis G, Durbin R: **The sequence alignment/map format and SAMtools**. *Bioinformatics* 2009, **25**(16):2078-2079.
62. Matthijs G, Souche E, Alders M, Corveleyn A, Eck S, Feenstra I, Race V, Sistermans E, Sturm M, Weiss M: **Guidelines for diagnostic next-generation sequencing**. *European Journal of Human Genetics* 2016, **24**(1):2-5.
63. van Oeveren J, Janssen A: **Mining SNPs from DNA sequence data; computational approaches to SNP discovery and analysis**. *Methods Mol Biol* 2009, **578**:73-91.
64. Li H: **Toward better understanding of artifacts in variant calling from high-coverage samples**. *Bioinformatics* 2014, **30**(20):2843-2851.
65. Auwera GA, Carneiro MO, Hartl C, Poplin R, del Angel G, Levy - Moonshine A, Jordan T, Shakir K, Roazen D, Thibault J: **From FastQ data to high - confidence variant calls: the genome analysis toolkit best practices pipeline**. *Current Protocols in Bioinformatics* 2013:11.10. 11-11.10. 33.
66. McKenna A, Hanna M, Banks E, Sivachenko A, Cibulskis K, Kernytsky A, Garimella K, Altshuler D, Gabriel S, Daly M: **The Genome Analysis Toolkit: a MapReduce framework for analyzing next-generation DNA sequencing data**. *Genome research* 2010, **20**(9):1297-1303.
67. Li H, Ruan J, Durbin R: **Mapping short DNA sequencing reads and calling variants using mapping quality scores**. *Genome research* 2008, **18**(11):1851-1858.
68. Garrison E, Marth G: **Haplotype-based variant detection from short-read sequencing**. *arXiv preprint arXiv:12073907* 2012.
69. Wigginton JE, Cutler DJ, Abecasis GR: **A note on exact tests of Hardy-Weinberg equilibrium**. *The American Journal of Human Genetics* 2005, **76**(5):887-893.
70. Slatkin M: **Linkage disequilibrium—understanding the evolutionary past and mapping the medical future**. *Nature Reviews Genetics* 2008, **9**(6):477-485.
71. Quinlan AR, Hall IM: **BEDTools: a flexible suite of utilities for comparing genomic features**. *Bioinformatics* 2010, **26**(6):841-842.
72. Lex A, Gehlenborg N, Strobelt H, Vuillemot R, Pfister H: **UpSet: visualization of intersecting sets**. *Visualization and Computer Graphics, IEEE Transactions on* 2014, **20**(12):1983-1992.
73. Li MJ, Wang J: **Current trend of annotating single nucleotide variation in humans–A case study on SNVrap**. *Methods* 2015, **79**:32-40.

74. Wang Z, Gerstein M, Snyder M: **RNA-Seq: a revolutionary tool for transcriptomics**. *Nature Reviews Genetics* 2009, **10**(1):57-63.
75. Sauna ZE, Kimchi-Sarfaty C: **Understanding the contribution of synonymous mutations to human disease**. *Nature Reviews Genetics* 2011, **12**(10):683-691.
76. Li MJ, Yan B, Sham PC, Wang J: **Exploring the function of genetic variants in the non-coding genomic regions: approaches for identifying human regulatory variants affecting gene expression**. *Briefings in bioinformatics* 2014:bbu018.
77. Faber K, Glatting K-H, Mueller PJ, Risch A, Hotz-Wagenblatt A: **Genome-wide prediction of splice-modifying SNPs in human genes using a new analysis pipeline called AASsites**. *BMC bioinformatics* 2011, **12**(4):1.
78. Wu J, Jiang R: **Prediction of deleterious nonsynonymous single-nucleotide polymorphism for human diseases**. *The Scientific World Journal* 2013, **2013**.
79. International HapMap C, Altshuler DM, Gibbs RA, Peltonen L, Altshuler DM, Gibbs RA, Peltonen L, Dermitzakis E, Schaffner SF, Yu F *et al*: **Integrating common and rare genetic variation in diverse human populations**. *Nature* 2010, **467**(7311):52-58.
80. Richards S, Aziz N, Bale S, Bick D, Das S, Gastier-Foster J, Grody WW, Hegde M, Lyon E, Spector E *et al*: **Standards and guidelines for the interpretation of sequence variants: a joint consensus recommendation of the American College of Medical Genetics and Genomics and the Association for Molecular Pathology**. *Genet Med* 2015, **17**(5):405-424.
81. Elmagarmid AK, Rusinkiewicz M, Sheth A: **Management of heterogeneous and autonomous database systems**: Morgan Kaufmann; 1999.
82. Debbarma MK, Debbarma S, Debbarma N, Chakma K, Jamatia A: **A review and analysis of software complexity metrics in structural testing**. *International Journal of Computer and Communication Engineering* 2013, **2**(2):129.
83. Halstead MH: **Elements of software science**, vol. 7: Elsevier New York; 1977.
84. Sharma A, Kushwaha DS: **Complexity measure based on requirement engineering document and its validation**. In: *Computer and Communication Technology (ICCCT), 2010 International Conference on: 2010*. IEEE: 608-615.
85. Card DN, Glass RL: **Measuring software design quality**: Prentice-Hall, Inc.; 1990.
86. Tegarden DP, Sheetz SD, Monarchi DE: **A software complexity model of object-oriented systems**. *Decision Support Systems* 1995, **13**(3):241-262.
87. Laenen G, Thorrez L, Bornigen D, Moreau Y: **Finding the targets of a drug by integration of gene expression data with a protein interaction network**. *Mol Biosyst* 2013, **9**(7):1676-1685.
88. Laenen G, Ardeshirdavani A, Moreau Y, Thorrez L: **Galahad: a web server for drug effect analysis from gene expression**. *Nucleic Acids Res* 2015, **43**(W1):W208-212.
89. Laney D: **3D data management: Controlling data volume, velocity and variety**. *META Group Research Note* 2001, **6**:70.
90. Taylor RC: **An overview of the Hadoop/MapReduce/HBase framework and its current applications in bioinformatics**. *BMC bioinformatics* 2010, **11**(Suppl 12):S1.

91. Niemenmaa M, Kallio A, Schumacher A, Klemelä P, Korpelainen E, Heljanko K: **Hadoop-BAM: directly manipulating next generation sequencing data in the cloud**. *Bioinformatics* 2012, **28**(6):876-877.
92. Nguyen T, Shi W, Ruden D: **CloudAligner: A fast and full-featured MapReduce based tool for sequence mapping**. *BMC research notes* 2011, **4**(1):171.
93. D O'Connor B, Merriman B, Nelson SF: **SeqWare Query Engine: storing and searching sequence data in the cloud**. *BMC bioinformatics* 2010, **11**(12):1.
94. Hadoop A: **Welcome to apache hadoop**. *Hämtat från* http://hadoop/ *apache org* 2014.
95. White T: **Hadoop: The definitive guide**: " O'Reilly Media, Inc."; 2012.
96. **Protection of personal data** [http://europa.eu/legislation_summaries/information_society/data_protection/l14012_en.htm]
97. **Data Protection Directive** [http://en.wikipedia.org/wiki/Data_Protection_Directive]
98. Fujita PA, Rhead B, Zweig AS, Hinrichs AS, Karolchik D, Cline MS, Goldman M, Barber GP, Clawson H, Coelho A *et al*: **The UCSC Genome Browser database: update 2011**. *Nucleic Acids Res* 2011, **39**(Database issue):D876-882.
99. Danecek P, Auton A, Abecasis G, Albers CA, Banks E, DePristo MA, Handsaker RE, Lunter G, Marth GT, Sherry ST *et al*: **The variant call format and VCFtools**. *Bioinformatics* 2011, **27**(15):2156-2158.
100. Van Houdt JK, Nowakowska BA, Sousa SB, van Schaik BD, Seuntjens E, Avonce N, Sifrim A, Abdul-Rahman OA, van den Boogaard MJ, Bottani A *et al*: **Heterozygous missense mutations in SMARCA2 cause Nicolaides-Baraitser syndrome**. *Nat Genet* 2012, **44**(4):445-449, S441.
101. Tryka KA, Hao L, Sturcke A, Jin Y, Wang ZY, Ziyabari L, Lee M, Popova N, Sharopova N, Kimura M *et al*: **NCBI's Database of Genotypes and Phenotypes: dbGaP**. *Nucleic Acids Res* 2014, **42**(Database issue):D975-979.
102. Bunnage ME: **Getting pharmaceutical R&D back on target**. *Nat Chem Biol* 2011, **7**(6):335-339.
103. Arrowsmith J: **A decade of change**. *Nat Rev Drug Discov* 2012, **11**(1):17-18.
104. Arrowsmith J, Harrison R: **Drug repositioning: the business case and current strategies to repurpose shelved candidates and marketed drugs**. *Drug Repositioning: Bringing New Life to Shelved Assets and Existing Drugs John Wiley & Sons, Inc, Hoboken, NJ* 2012:9-31.
105. Sirota M, Dudley JT, Kim J, Chiang AP, Morgan AA, Sweet-Cordero A, Sage J, Butte AJ: **Discovery and preclinical validation of drug indications using compendia of public gene expression data**. *Sci Transl Med* 2011, **3**(96):96ra77.
106. Iskar M, Campillos M, Kuhn M, Jensen LJ, van Noort V, Bork P: **Drug-induced regulation of target expression**. *PLoS Comput Biol* 2010, **6**(9).
107. Hurle MR, Yang L, Xie Q, Rajpal DK, Sanseau P, Agarwal P: **Computational drug repositioning: from data to therapeutics**. *Clin Pharmacol Ther* 2013, **93**(4):335-341.
108. Qu XA, Rajpal DK: **Applications of Connectivity Map in drug discovery and development**. *Drug Discov Today* 2012, **17**(23-24):1289-1298.

109. Montero-Melendez T, Perretti M: **Connections in pharmacology: innovation serving translational medicine**. *Drug Discov Today* 2014, **19**(7):820-823.
110. Lamb J: **The Connectivity Map: a new tool for biomedical research**. *Nat Rev Cancer* 2007, **7**(1):54-60.
111. Lamb J, Crawford ED, Peck D, Modell JW, Blat IC, Wrobel MJ, Lerner J, Brunet JP, Subramanian A, Ross KN *et al*: **The Connectivity Map: using gene-expression signatures to connect small molecules, genes, and disease**. *Science* 2006, **313**(5795):1929-1935.
112. Irizarry RA, Bolstad BM, Collin F, Cope LM, Hobbs B, Speed TP: **Summaries of Affymetrix GeneChip probe level data**. *Nucleic Acids Res* 2003, **31**(4):e15.
113. Gautier L, Cope L, Bolstad BM, Irizarry RA: **affy--analysis of Affymetrix GeneChip data at the probe level**. *Bioinformatics* 2004, **20**(3):307-315.
114. Carvalho BS, Irizarry RA: **A framework for oligonucleotide microarray preprocessing**. *Bioinformatics* 2010, **26**(19):2363-2367.
115. Wilson CL, Miller CJ: **Simpleaffy: a BioConductor package for Affymetrix Quality Control and data analysis**. *Bioinformatics* 2005, **21**(18):3683-3685.
116. Bolstad B, Collin F, Brettschneider J, Simpson K, Cope L, Irizarry R, Speed T: **Quality assessment of Affymetrix GeneChip data**. In: *Bioinformatics and computational biology solutions using R and bioconductor*. Springer; 2005: 33-47.
117. Ritchie ME, Phipson B, Wu D, Hu Y, Law CW, Shi W, Smyth GK: **limma powers differential expression analyses for RNA-sequencing and microarray studies**. *Nucleic Acids Res* 2015, **43**(7):e47.
118. Benjamini Y, Hochberg Y: **Controlling the false discovery rate: a practical and powerful approach to multiple testing**. *Journal of the Royal Statistical Society Series B (Methodological)* 1995:289-300.
119. Szklarczyk D, Franceschini A, Kuhn M, Simonovic M, Roth A, Minguez P, Doerks T, Stark M, Muller J, Bork P *et al*: **The STRING database in 2011: functional interaction networks of proteins, globally integrated and scored**. *Nucleic Acids Res* 2011, **39**(Database issue):D561-568.
120. Bento AP, Gaulton A, Hersey A, Bellis LJ, Chambers J, Davies M, Kruger FA, Light Y, Mak L, McGlinchey S *et al*: **The ChEMBL bioactivity database: an update**. *Nucleic Acids Res* 2014, **42**(Database issue):D1083-1090.
121. Croft D, Mundo AF, Haw R, Milacic M, Weiser J, Wu G, Caudy M, Garapati P, Gillespie M, Kamdar MR *et al*: **The Reactome pathway knowledgebase**. *Nucleic Acids Res* 2014, **42**(Database issue):D472-477.
122. Kohler S, Doelken SC, Mungall CJ, Bauer S, Firth HV, Bailleul-Forestier I, Black GC, Brown DL, Brudno M, Campbell J *et al*: **The Human Phenotype Ontology project: linking molecular biology and disease through phenotype data**. *Nucleic Acids Res* 2014, **42**(Database issue):D966-974.
123. Highnam G, Wang JJ, Kusler D, Zook J, Vijayan V, Leibovich N, Mittelman D: **An analytical framework for optimizing variant discovery from personal genomes**. *Nat Commun* 2015, **6**:6275.
124. Botstein D, Risch N: **Discovering genotypes underlying human phenotypes: past successes for mendelian disease, future approaches for complex disease**. *Nat Genet* 2003, **33 Suppl**:228-237.

125. Karchin R: **Next generation tools for the annotation of human SNPs**. *Brief Bioinform* 2009, **10**(1):35-52.
126. Stenson PD, Ball EV, Mort M, Phillips AD, Shaw K, Cooper DN: **The Human Gene Mutation Database (HGMD) and its exploitation in the fields of personalized genomics and molecular evolution**. *Curr Protoc Bioinformatics* 2012, **Chapter 1**:Unit1 13.
127. Coordinators NR: **Database resources of the National Center for Biotechnology Information**. *Nucleic Acids Res* 2015, **43**(Database issue):D6-17.
128. Pavlopoulos GA, Oulas A, Iacucci E, Sifrim A, Moreau Y, Schneider R, Aerts J, Iliopoulos I: **Unraveling genomic variation from next generation sequencing data**. *BioData Min* 2013, **6**(1):13.
129. Sims D, Sudbery I, Ilott NE, Heger A, Ponting CP: **Sequencing depth and coverage: key considerations in genomic analyses**. *Nat Rev Genet* 2014, **15**(2):121-132.
130. Li Y, Sidore C, Kang HM, Boehnke M, Abecasis GR: **Low-coverage sequencing: implications for design of complex trait association studies**. *Genome Res* 2011, **21**(6):940-951.
131. Piro RM, Di Cunto F: **Computational approaches to disease-gene prediction: rationale, classification and successes**. *FEBS J* 2012, **279**(5):678-696.
132. Moreau Y, Tranchevent LC: **Computational tools for prioritizing candidate genes: boosting disease gene discovery**. *Nat Rev Genet* 2012, **13**(8):523-536.
133. Goh KI, Cusick ME, Valle D, Childs B, Vidal M, Barabasi AL: **The human disease network**. *Proc Natl Acad Sci U S A* 2007, **104**(21):8685-8690.
134. Smith NG, Eyre-Walker A: **Human disease genes: patterns and predictions**. *Gene* 2003, **318**:169-175.
135. Chen J, Bardes EE, Aronow BJ, Jegga AG: **ToppGene Suite for gene list enrichment analysis and candidate gene prioritization**. *Nucleic Acids Res* 2009, **37**(Web Server issue):W305-311.
136. Seelow D, Schwarz JM, Schuelke M: **GeneDistiller--distilling candidate genes from linkage intervals**. *PLoS One* 2008, **3**(12):e3874.
137. Fontaine JF, Priller F, Barbosa-Silva A, Andrade-Navarro MA: **Genie: literature-based gene prioritization at multi genomic scale**. *Nucleic Acids Res* 2011, **39**(Web Server issue):W455-461.
138. Pers TH, Dworzynski P, Thomas CE, Lage K, Brunak S: **MetaRanker 2.0: a web server for prioritization of genetic variation data**. *Nucleic Acids Res* 2013, **41**(Web Server issue):W104-108.
139. Tranchevent LC, Barriot R, Yu S, Van Vooren S, Van Loo P, Coessens B, De Moor B, Aerts S, Moreau Y: **ENDEAVOUR update: a web resource for gene prioritization in multiple species**. *Nucleic Acids Res* 2008, **36**(Web Server issue):W377-384.
140. Radivojac P, Clark WT, Oron TR, Schnoes AM, Wittkop T, Sokolov A, Graim K, Funk C, Verspoor K, Ben-Hur A *et al*: **A large-scale evaluation of computational protein function prediction**. *Nat Methods* 2013, **10**(3):221-227.
141. Yu L, Wynn J, Cheung YH, Shen Y, Mychaliska GB, Crombleholme TM, Azarow KS, Lim FY, Chung DH, Potoka D *et al*: **Variants in GATA4 are a rare cause of**

familial and sporadic congenital diaphragmatic hernia. *Hum Genet* 2013, **132**(3):285-292.

142. Zielinski D, Markus B, Sheikh M, Gymrek M, Chu C, Zaks M, Srinivasan B, Hoffman JD, Aizenbud D, Erlich Y: **OTX2 duplication is implicated in hemifacial microsomia**. *PLoS One* 2014, **9**(5):e96788.
143. Aerts S, Lambrechts D, Maity S, Van Loo P, Coessens B, De Smet F, Tranchevent LC, De Moor B, Marynen P, Hassan B *et al*: **Gene prioritization through genomic data fusion**. *Nat Biotechnol* 2006, **24**(5):537-544.
144. Gene Ontology C: **Gene Ontology Consortium: going forward**. *Nucleic Acids Res* 2015, **43**(Database issue):D1049-1056.
145. Mitchell A, Chang HY, Daugherty L, Fraser M, Hunter S, Lopez R, McAnulla C, McMenamin C, Nuka G, Pesseat S *et al*: **The InterPro protein families database: the classification resource after 15 years**. *Nucleic Acids Res* 2015, **43**(Database issue):D213-221.
146. Fabregat A, Sidiropoulos K, Garapati P, Gillespie M, Hausmann K, Haw R, Jassal B, Jupe S, Korninger F, McKay S *et al*: **The Reactome pathway Knowledgebase**. *Nucleic Acids Res* 2016, **44**(D1):D481-487.
147. Chatr-Aryamontri A, Breitkreutz BJ, Oughtred R, Boucher L, Heinicke S, Chen D, Stark C, Breitkreutz A, Kolas N, O'Donnell L *et al*: **The BioGRID interaction database: 2015 update**. *Nucleic Acids Res* 2015, **43**(Database issue):D470-478.
148. Orchard S, Ammari M, Aranda B, Breuza L, Briganti L, Broackes-Carter F, Campbell NH, Chavali G, Chen C, del-Toro N *et al*: **The MIntAct project--IntAct as a common curation platform for 11 molecular interaction databases**. *Nucleic Acids Res* 2014, **42**(Database issue):D358-363.
149. Knox C, Law V, Jewison T, Liu P, Ly S, Frolkis A, Pon A, Banco K, Mak C, Neveu V *et al*: **DrugBank 3.0: a comprehensive resource for 'omics' research on drugs**. *Nucleic Acids Res* 2011, **39**(Database issue):D1035-1041.
150. Hamosh A, Scott AF, Amberger J, Valle D, McKusick VA: **Online Mendelian Inheritance in Man (OMIM)**. *Hum Mutat* 2000, **15**(1):57-61.
151. Shimoyama M, De Pons J, Hayman GT, Laulederkind SJ, Liu W, Nigam R, Petri V, Smith JR, Tutaj M, Wang SJ *et al*: **The Rat Genome Database 2015: genomic, phenotypic and environmental variations and disease**. *Nucleic Acids Res* 2015, **43**(Database issue):D743-750.
152. Pan JB, Hu SC, Shi D, Cai MC, Li YB, Zou Q, Ji ZL: **PaGenBase: a pattern gene database for the global and dynamic understanding of gene function**. *PLoS One* 2013, **8**(12):e80747.
153. Agarwal V, Bell GW, Nam JW, Bartel DP: **Predicting effective microRNA target sites in mammalian mRNAs**. *Elife* 2015, **4**.
154. Hausser J, Berninger P, Rodak C, Jantscher Y, Wirth S, Zavolan M: **MirZ: an integrated microRNA expression atlas and target prediction resource**. *Nucleic Acids Res* 2009, **37**(Web Server issue):W266-272.
155. Myers CL, Barrett DR, Hibbs MA, Huttenhower C, Troyanskaya OG: **Finding function: evaluation methods for functional genomic data**. *BMC Genomics* 2006, **7**:187.

Appendix A: Glossary

Descriptions in this glossary are borrowed from the NIH glossary of genetic terms: https://www.genome.gov/glossary/

ACGT	Is an acronym for the four types of bases found in a DNA molecule: adenine (A), cytosine (C), guanine (G), and thymine (T). A DNA molecule consists of two strands wound around each other, with each strand held together by bonds between the bases. Adenine pairs with thymine, and cytosine pairs with guanine. The sequence of bases in a portion of a DNA molecule, called a gene, carries the instructions needed to assemble a protein.
Acquired genetic disorders	Some diseases are caused by mutations that are inherited from the parents and are present in an individual at birth, like sickle cell disease. Other diseases are caused by acquired mutations in a gene or group of genes that occur during a person's life.
Allele	An allele is one of two or more versions of a gene. An individual inherits two alleles for each gene, one from each parent. If the two alleles are the same, the individual is homozygous for that gene. If the alleles are different, the individual is heterozygous. Though the term allele was originally used to describe variation among genes, it now also refers to variation among non-coding DNA sequences.
Allele depth	Ad is the unfiltered allele depth, i.e. The number of reads that support each of the reported alleles. All reads at the position (including reads that did not pass the variant caller's filters) are included in this number, except reads that were considered uninformative. Reads are considered uninformative when they do not provide enough statistical evidence to support one allele over another.
Allele frequency	The proportion of individuals in a population who have inherited a specific gene mutation or variant.
Allelic heterogeneity	Different mutations in the same gene at the same chromosomal locus that cause a single phenotype.
Amino acids	Amino acids are a set of 20 different molecules used to build proteins. Proteins consist of one or more chains of amino acids called polypeptides. The sequence of the amino acid chain causes the polypeptide to fold into a shape that is biologically active. The amino acid sequences of proteins are encoded in the genes.
Annotation	The process to predict the effect or function of an individual gene or SNP by using related annotational tools.
Assembly	When the genome of a species is to be sequenced, the chromosomes from many cells are broken at random positions into small fragments, which are sequenced, and reassembled into long sequences (contigs). Contigs may be assembled into longer sequences called scaffolds and sometimes, if the depth of sequencing is high enough, there may be enough information to assemble most of the scaffolds into chromosomes. The resulting collection of sequences after assembly is called a genome assembly.
Autosomal	Refers to any of the chromosomes other than the sex-determining chromosomes (i.e., the X and Y) or the genes on these chromosomes.
Autosomal dominant	Describes a trait or disorder in which the phenotype is expressed in those who have inherited only one copy of a particular gene mutation (heterozygotes);

	specifically refers to a gene on one of the 22 pairs of autosomes (non-sex chromosomes).
Autosomal recessive	Describes a trait or disorder requiring the presence of two copies of a gene mutation at a particular locus in order to express observable phenotype; specifically refers to genes on one of the 22 pairs of autosomes (non-sex chromosomes).
Base pair	A base pair is two chemical bases bonded to one another forming a "rung of the dna ladder." the dna molecule consists of two strands that wind around each other like a twisted ladder. Each strand has a backbone made of alternating sugar (deoxyribose) and phosphate groups. Attached to each sugar is one of four bases--adenine (A), cytosine (C), guanine (G), or thymine (T). The two strands are held together by hydrogen bonds between the bases, with adenine forming a base pair with thymine, and cytosine forming a base pair with guanine.
Bioinformatics	Bioinformatics is a subdiscipline of biology and computer science concerned with the acquisition, storage, analysis, and dissemination of biological data, most often DNA and amino acid sequences. Bioinformatics uses computer programs for a variety of applications, including determining gene and protein functions, establishing evolutionary relationships, and predicting the three-dimensional shapes of proteins.
Candidate gene	A candidate gene is a gene whose chromosomal location is associated with a particular disease or other phenotype. Because of its location, the gene is suspected of causing the disease or other phenotype.
Chromosome	A chromosome is an organized package of DNA found in the nucleus of the cell. Different organisms have different numbers of chromosomes. Humans have 23 pairs of chromosomes--22 pairs of numbered chromosomes, called autosomes, and one pair of sex chromosomes, x and y. Each parent contributes one chromosome to each pair so that offspring get half of their chromosomes from their mother and half from their father.
Clinical significance	This attribute of a SNP is generally imported from ClinVar, either assigned by the submitter or by dbSNP or OMIM. The most common value is probable-pathogenic; the other possibilities are pathogenic, non-pathogenic, probable-non-pathogenic, drug-response, histocompatibility, untested and other.
Complex disorder	Common medical problems such as heart disease, diabetes, and obesity do not have a single genetic cause—they are likely associated with the effects of multiple genes in combination with lifestyle and environmental factors. Conditions caused by many contributing factors are called complex or multifactorial disorders.
Coverage	Sequencing coverage describes the average number of reads that align to, or "cover," known reference bases. The NGS (next-generation sequencing) coverage level often determines whether variant discovery can be made with a certain degree of confidence at particular base positions. Sequencing coverage requirements vary by application, as noted below. At higher levels of coverage, each base is covered by a greater number of aligned sequence reads, so base calls can be made with a higher degree of confidence.
De novo mutation	A DNA variation that is either germline or somatic but is not readily found in either parent. Germline de novo mutations can be mutations arising in a parental gamete (sperm or egg), which lead post-fertilization to being present in all cells of the offspring. Somatic de novo mutations arise in a somatic cell, and often are therefore found in only a percentage of the cells of that type; they are not passed on to offspring.

Deletion	Deletion is a type of mutation involving the loss of genetic material. It can be small, involving a single missing DNA base pair, or large, involving a piece of a chromosome.
Depth of coverage	DP is the filtered depth, at the sample level. This gives you the number of filtered reads that support each of the reported alleles. You can check the variant caller's documentation to see which filters are applied by default. Only reads that passed the variant caller's filters are included in this number. However, unlike the ad calculation, uninformative reads are included in DP.
Diploid	Diploid is a cell or organism that has paired chromosomes, one from each parent. In humans, cells other than human sex cells, are diploid and have 23 pairs of chromosomes. Human sex cells (egg and sperm cells) contain a single set of chromosomes and are known as haploid.
DNA	DNA is the chemical name for the molecule that carries genetic instructions in all living things. The DNA molecule consists of two strands that wind around one another to form a shape known as a double helix. Each strand has a backbone made of alternating sugar (deoxyribose) and phosphate groups. Attached to each sugar is one of four bases--adenine (A), cytosine (C), guanine (G), and thymine (T). The two strands are held together by bonds between the bases; adenine bonds with thymine, and cytosine bonds with guanine. The sequence of the bases along the backbones serves as instructions for assembling protein and RNA molecules.
DNA sequencing	Is a laboratory technique used to determine the exact sequence of bases (A, C, G, and T) in a DNA molecule.
Duplication	Duplication is a type of mutation that involves the production of one or more copies of a gene or region of a chromosome. Gene and chromosome duplications occur in all organisms, though they are especially prominent among plants. Gene duplication is an important mechanism by which evolution occurs.
Exon	An exon is the portion of a gene that codes for amino acids. In the cells of plants and animals, most gene sequences are broken up by one or more DNA sequences called introns. The parts of the gene sequence that are expressed in the protein are called exons, because they are expressed, while the parts of the gene sequence that are not expressed in the protein are called introns, because they come in between--or interfere with--the exons.
GC content	In molecular biology and genetics, GC-content (or guanine-cytosine content) is the percentage of nitrogenous bases on a DNA molecule that are either guanine or cytosine (from a possibility of four different ones, also including adenine and thymine).
Gene	The gene is the basic physical unit of inheritance. Genes are passed from parents to offspring and contain the information needed to specify traits. Genes are arranged, one after another, on structures called chromosomes. A chromosome contains a single, long DNA molecule, only a portion of which corresponds to a single gene. Humans have approximately 20,000 genes arranged on their chromosomes.
Gene expression	Is the process by which the information encoded in a gene is used to direct the assembly of a protein molecule. The cell reads the sequence of the gene in groups of three bases. Each group of three bases (codon) corresponds to one of 20 different amino acids used to build the protein.
Genetic variation	Genetic variation refers to diversity in gene frequencies. Genetic variation can refer to differences between individuals or to differences between populations. Mutation is the ultimate source of genetic variation, but mechanisms such as sexual reproduction and genetic drift contribute to it as well.

Genome	The genome is the entire set of genetic instructions found in a cell. In humans, the genome consists of 23 pairs of chromosomes, found in the nucleus, as well as a small chromosome found in the cells' mitochondria. Each set of 23 chromosomes contains approximately 3.1 billion bases of DNA sequence.
Genomics	Genomics refers to the study of the entire genome of an organism whereas genetics refers to the study of a particular gene.
Haploid	Haploid is the quality of a cell or organism having a single set of chromosomes. Organisms that reproduce asexually are haploid. Sexually reproducing organisms are diploid (having two sets of chromosomes, one from each parent). In humans, only their egg and sperm cells are haploid.
Haplotype	A haplotype is a set of DNA variations, or polymorphisms, that tend to be inherited together. A haplotype can refer to a combination of alleles or to a set of single nucleotide polymorphisms (SNPs) found on the same chromosome. Information about haplotypes is being collected by the international HapMap project and is used to investigate the influence of genes on disease.
Hemizygous	When used in a phrase like, "the patient was hemizygous for the mutation", this indicates that the person was carrying only one allele (in total) for this SNP, usually because the corresponding area of the other (autosomal) chromosome has been deleted. Hemizygosity also can refer to x chromosome alleles in males.
Heritability	The percentage of a trait estimated to be due to the variations you've inherited, as opposed to the contribution of non-genetic influences such as your environment or diet. For example, the heritability of bipolar disorder is estimated to be 70%, which means that genetic variations account for 70% of the overall odds of developing the disorder and non-genetic factors account for 30%.
Heterozygous	Heterozygous refers to having inherited different forms of a particular gene from each parent. A heterozygous genotype stands in contrast to a homozygous genotype, where an individual inherits identical forms of a particular gene from each parent.
Homozygous	Homozygous is a genetic condition where an individual inherits the same alleles for a particular gene from both parents.
Human genome project	The human genome project was an international project that mapped and sequenced the entire human genome. Completed in April 2003, data from the project are freely available to researchers and others interested in genetics and human health.
Identical twins	Identical twins are also known as monozygotic twins. They result from the fertilization of a single egg that splits in two. Identical twins share all of their genes and are always of the same sex. In contrast, fraternal, or dizygotic, twins result from the fertilization of two separate eggs during the same pregnancy. They share half of their genes, just like any other siblings. Fraternal twins can be of the same or different sexes.
InDel	While many SNPs are swaps between the four DNA bases (A, C, G and T as abbreviated), extra DNA bases may be present (called "insertions"), or a few DNA bases may be missing ("deletions"). The term in/del refers to an insertion or deletion of this type.
Inherited	An inherited trait is one that is genetically determined. Inherited traits are passed from parent to offspring according to the rules of Mendelian genetics. Most traits are not strictly determined by genes, but rather are influenced by both genes and environment.
Insertion	Insertion is a type of mutation involving the addition of genetic material. An insertion mutation can be small, involving a single extra DNA base pair, or large, involving a piece of a chromosome.

Intron	An intron is a portion of a gene that does not code for amino acids. In the cells of plants and animals, most gene sequences are broken up by one or more introns. The parts of the gene sequence that are expressed in the protein are called exons, because they are expressed, while the parts of the gene sequence that are not expressed in the protein are called introns, because they come in between the exons.
Mendelian disorder	Mendelian inheritance refers to patterns of inheritance that are characteristic of organisms that reproduce sexually.
Missense mutations	A missense mutation is when the change of a single base pair causes the substitution of a different amino acid in the resulting protein. This amino acid substitution may have no effect, or it may render the protein nonfunctional.
Mutation	A mutation is a change in a DNA sequence. Mutations can result from DNA copying mistakes made during cell division, exposure to ionizing radiation, exposure to chemicals called mutagens, or infection by viruses. Germ line mutations occur in the eggs and sperm and can be passed on to offspring, while somatic mutations occur in body cells and are not passed on.
Next generation sequencing	Next-generation sequencing (NGS), also known as high-throughput sequencing, is the catch-all term used to describe a number of different modern sequencing technologies. These recent technologies allow us to sequence DNA and RNA much more quickly and cheaply than the previously used sanger sequencing, and as such have revolutionized the study of genomics and molecular biology.
Non-coding DNA	Non-coding DNA sequences do not code for amino acids. Most non-coding DNA lies between genes on the chromosome and has no known function. Other non-coding DNA, called introns, is found within genes. Some non-coding DNA plays a role in the regulation of gene expression.
Non-synonymous mutation	A nonsynonymous substitution is a nucleotide mutation that alters the amino acid sequence of a protein. It is contrasted with synonymous substitution which do not alter amino acid sequences. As nonsynonymous substitutions result in a biological change in the organism, they are subject to natural selection.
Nonsense mutations	A nonsense mutation is the substitution of a single base pair that leads to the appearance of a stop codon where previously there was a codon specifying an amino acid. The presence of this premature stop codon results in the production of a shortened, and likely nonfunctional, protein.
Paired-end sequence	The term 'paired ends' refers to the two ends of the same DNA molecule. So you can sequence one end, then turn it around and sequence the other end. The two sequences you get are 'paired end reads'.
Phenotype	A phenotype is an individual's observable traits, such as height, eye color, and blood type. The genetic contribution to the phenotype is called the genotype. Some traits are largely determined by the genotype, while other traits are largely determined by environmental factors.
Phred quality score	A Phred quality score is a measure of the quality of the identification of the nucleobases generated by automated DNA sequencing. It was originally developed for Phred base calling to help in the automation of DNA sequencing in the human genome project.
Point mutation	A point mutation is when a single base pair is altered. Point mutations can have one of three effects. First, the base substitution can be a silent mutation where the altered codon corresponds to the same amino acid. Second, the base substitution can be a missense mutation where the altered codon corresponds to a different amino acid. Or third, the base substitution can be a nonsense mutation where the altered codon corresponds to a stop signal.

Polymorphism	Polymorphism involves one of two or more variants of a particular DNA sequence. The most common type of polymorphism involves variation at a single base pair. Polymorphisms can also be much larger in size and involve long stretches of DNA. Called a single nucleotide polymorphism, or SNP, scientists are studying how SNPs in the human genome correlate with disease, drug response, and other phenotypes.
Population genomics	Population genomics is the application of genomic technologies to understand populations of organisms. In humans, population genomics typically refers to applying technology in the quest to understand how genes contribute to our health and well-being.
Position	The simple answer is that this is the location of a SNP on the chromosome.
Protein	Proteins are an important class of molecules found in all living cells. A protein is composed of one or more long chains of amino acids, the sequence of which corresponds to the DNA sequence of the gene that encodes it. Proteins play a variety of roles in the cell, including structural (cytoskeleton), mechanical (muscle), biochemical (enzymes), and cell signaling (hormones). Proteins are also an essential part of diet.
Raw data	Raw data refers to any data object that hasn't undergone thorough processing, either manually or through automated computer software. Raw data may be gathered from various processes and it resources. Raw data is also known as source data, primary data or atomic data.
Read	A read refers to the sequence of a cluster that is obtained after the end of the sequencing process which is ultimately the sequence of a section of a unique fragment.
Recessive	Recessive is a quality found in the relationship between two versions of a gene. Individuals receive one version of a gene, called an allele, from each parent. If the alleles are different, the dominant allele will be expressed, while the effect of the other allele, called recessive, is masked. In the case of a recessive genetic disorder, an individual must inherit two copies of the mutated allele in order for the disease to be present.
Reference genome	A reference genome (also known as a reference assembly) is a digital nucleic acid sequence database, assembled by scientists as a representative example of a species' set of genes.
rs#	All this reflects is that the SNP in question has been officially registered and given an (rs) identifier by dbSNP, the largest public database of SNPs, maintained by the national institutes of health.
Sex chromosome	A sex chromosome is a type of chromosome that participates in sex determination. Humans and most other mammals have two sex chromosomes, the x and the y. Females have two x chromosomes in their cells, while males have both x and a y chromosomes in their cells. Egg cells all contain an x chromosome, while sperm cells contain an x or y chromosome. This arrangement means that it is the male that determines the sex of the offspring when fertilization occurs.
Single nucleotide polymorphism	Single nucleotide polymorphisms (SNPs) are a type of polymorphism involving variation of a single base pair. Scientists are studying how single nucleotide polymorphisms, or SNPs (pronounced "snips"), in the human genome correlate with disease, drug response, and other phenotypes.
Stop codon	A stop codon is a trinucleotide sequence within a messenger RAN (mRNA) molecule that signals a halt to protein synthesis. The genetic code describes the relationship between the sequence of DNA bases (a, c, g, and t) in a gene and the corresponding protein sequence that it encodes. The cell reads the sequence of the gene in groups of three bases. Of the 64 possible combinations of three bases,

	61 specify an amino acid, while the remaining three combinations are stop codons.
Suffix array	In computer science, a suffix array is a sorted array of all suffixes of a string. It is a data structure used, among others, in full text indices, data compression algorithms and within the field of bioinformatics.
Synonymous mutation	Synonymous substitution do not alter amino acid sequences.
Target sequencing	Targeted resequencing is a method that aims to identify causative mutations within discrete genomic loci in populations by isolating and sequencing genomic regions of interest from a sample library. It is one of the fastest growing applications for next-generation sequencing technologies.
Transcriptomics	The study of transcriptomes and their functions.

Curriculum vitae

Amin Ardeshirdavani

Date and place of birth: 28 September, 1979 – Shiraz, Iran

Academic qualifications

Year	*Qualification*	*Program*	*Institution*
2011	*M.Sc.*	*Bioinformatics*	*KU Leuven, Leuven, Belgium*
2009	*B.Sc.*	*Software Engineering*	*University of Sadra, Tehran, Iran*

Non-Academic qualifications

Year	*Program*	*Institution*
2004	*Aptech Certified Computer Professional (ACCP)*	*APTECH*
2003	*International Computer Driving License (ICDL)*	*UNESCO*

Academic projects

Master thesis

- Protein Kinase 3D Database (KU Leuven)

Bachelor projects

- Development of a student management application (University of Sadra)
- Development of online sales application (University of Sadra)

Conference talks

- NGS'14: An Official Conference of the International Society for Computational Biology, 2-4 June 2014, Barcelona Spain
- SCS'14 – ISMB 2014: Student Council Symposium 10, Intelligent Systems in Molecular Biology, Boston, USA.
- BBC'2014: 9th Benelux Bioinformatics Conference
- BeSHG 2015: Belgian Society of Human Genetics conference
- ISMB-ECCB 2015: 23rd annual international conference on Intelligent Systems in Molecular Biology – 14th European Conference on Computational Biology
- iMinds SuperMinds 2015: Human Genomics Data Sharing (video: https://vimeo.com/144356748)
- ISMB 2016: Intelligent Systems for Molecular Biology, Orlando, USA.

Awards

- BeSHG 2015: Belgian Society of Human Genetics. Best selected presentation for *NGS-Logistics: Federated analysis of NGS sequence variants across multiple locations*.

Professional experience

- **2011–2016: KU Leuven (Leuven, Belgium)**
  PhD Student
  1. **Master thesis supervision**
     Analyzing large collections of NGS data based on the integration of Apache Hadoop, HBase, and Hive. Jia Yu, September 2014
  2. **Teaching assistance**
     - Bioinformatics and Systems Biology: Sequence, Structure and Evolution
     - Omics Techniques and Data Analysis
     - Integrated Bioinformatics Project
     - Biological Databases
  3. Member of the Data Access Committee of KU Leuven
  4. Member of the GA4GH workgroups on data sharing and data security
  5. Member of the ELIXIR Belgium
- **2010: KU Leuven** (Leuven, Belgium) **- Student Job**
  Developing an application to store student information at the department of service for students with disabilities
- **2006 – 2009: Zamyad Co.** (Tehran, Iran). (http://zamyadco.ir/)
  Designing, developing, and maintaining the following tools:
  1. Production control and planning.
  2. Engineering system (backbone of the supply chain system)
  3. User access control system
  4. Internet sale system
  5. Quality control system
  6. Website of the company
  7. Management reporting system

  * Some parts of 1 & 5 developed based on Barcode & RFID technology and Windows mobile
- **2005 – 2006: Saipa Special Service Co.** (Tehran, Iran) (http://saipacorp.com)
  Server and database maintenance administrator
- **2000 – 2005: IKD Co.** (Tehran, Iran) (http://www.ikd-co.com)
  Hardware and network maintenance administrator

List of publications

International Journal Papers:

- *eXtasy: Variant Prioritization by Genomic Data Fusion*. Sifrim A., Dusan P., Tranchevent L-C., **Ardershirdavani A.**, Sakai R., Konings P., Vermeesch J., Aerts J., De Moor B., Moreau Y. **Nature Methods**. 2013 Nov.
- *NGS-Logistics: Federated analysis of NGS sequence variants across multiple locations.* **Ardeshirdavani A.**, Souche E., Dehaspe L., Van Houdt J., Vermeesch J., Moreau Y.. **Genome Medicine** 2014 Sep.
- *Effective diagnosis of genetic disease by computational phenotype analysis of the disease-associated genome.* Zemojte T., Köhler S., Mackenroth L., Jäger M., Hecht J., Krawitz P., Graul-Neumann L., Doelken S., Ehmke N., Spielmann M., Christine Øien N., R.Schweiger M., Krüger U., Frommer G., Fischer B., Kornak U., **Ardeshirdavani A.**, Moreau Y., E. Lewis S., Haende M., Smedley D., Horn D., Mundlos S., Robinson P.. **Science Translational Medicine** 2014 Sep.
- *NGS-Logistics: data infrastructure for efficient analysis of NGS sequence variants across multiple centers.* **Ardeshirdavani A.**, Souche E., Dehaspe L., Van Houdt J., Vermeesch J., Moreau Y.. **BMC Bioinformatics** 2015 Jan.
- *Galahad: a web server for gene expression data analysis in support of drug development*. Laenen G.**, **Ardeshirdavani** A.**, Moreau Y., Thorrez L.. **Nucleic Acids Research** 2015 Jul
- *Beegle: A Generic Tool for Disease-Gene Annotation based on Literature Mining*. ElShal S., Tranchevent L., Sifrim A., Adreshirdavani A., Davis J., Moreau Y.. **Nucleic Acids Research** 2015 Jul.
- *Candidate gene prioritization with Endeavour.* Tranchevent L.**, **Ardeshirdavani A. ****, ElShal S., Alcaide D., Aerts J., Auboeuf D., and Moreau Y.. **Nucleic Acids Research** 2016 Apr.

Manuscripts in preparation:

- *Big Data solutions for Variant discovery from low coverage sequencing data, by integration of Hadoop, HBase and Hive.* **Ardeshirdavani A.,** Souche E., Yu J., Oldenhof M., Moreau Y..
- *HiVA: Haplarithm inference of Variant Alleles.* Zamani Esteki M. **, **Ardeshirdavani A.** **, Alcaide D. **, Aerts J., Vermeesch J., Moreau Y., Voet T.

** These authors contributed equally to this work as first authors.